Universidad Politécnica de Cartagena

industriales
etsii UPCT

PROYECTO FIN DE CARRERA
K-JUNIOR. COMUNICACIÓN Y VISIÓN

Titulación: Ingeniería Técnica Industrial
Especialidad: Electrónica Industrial 2012
Autor: Francisco Manuel López Pérez
DNI:48505185-W
Director de proyecto: D. José Luis Muñoz Lozano
Departamento de ingeniería de Sistemas y Automática

Cartagena, 24 de Julio de 2012

INDICE

CAPITULO I: Introducción del proyecto

1.- Motivación

La verdad, es que han existido muchas motivaciones con respecto a la elección de este tipo de proyecto.
La primera de todas, la que también me llevó a elegir la carrera de electrónica, es que desde siempre me han fascinado los robots. Cuando era pequeño me gustaba desmontar los robots que me compraban para ver cómo funcionaban, ¿cómo era posible que aquello se moviese?, me preguntaba. Después vinieron otros, como el de radio control, un robot de unos 90cm de alto con muchas funciones (movimiento, luces, sonido), esto aumentaba mi curiosidad. Después llegaron otros y otros que incrementaron esa atracción que sentía por la robótica.

La segunda motivación es que, aunque en alguna asignatura se dan nociones básicas sobre robótica móvil, mayoritariamente se ha destacado la parte de programación frente a la de construcción. A lo largo de la carrera, he echado en falta alguna asignatura donde a través de la práctica pudiésemos construir o programar alguno ya construido.

La tercera, es porque me gustó la idea de poder utilizar lo que había aprendido en las asignaturas de programación, ya que en nuestra carrera no sólo se dan los conocimientos de cómo hacer un buen diseño de circuitos y qué debemos tener en cuenta en nuestra fabricación; sino que también nos enseñan a cómo programar los micro-controladores y otros sistemas de control, que nos serán necesarios en el desarrollo de nuestros futuros proyectos.

La cuarta es que, por diversos motivos, deseaba algo diferente para el proyecto fin de carrera, algo diferente a lo que los demás compañeros han realizado.
Como he dicho anteriormente, mi inquietud sobre la robótica me ha acompañado desde siempre y este proyecto me ha permitido reflejar esa pasión por esa parte de la

electrónica, pudiendo aportar mi pequeño toque personal a la vez que obtener un mayor aprendizaje sobre este campo.

Como quinta motivación, puedo decir que, se me quita una espinita clavada al no poder haber cursado a lo largo de la carrera, por circunstancias, las asignaturas de robótica y visión artificial. Este proyecto me ha permitido aprender una pequeña base sobre robótica.

Por último, y tras la explicación de las diversas motivaciones que me han llevado a realizar este proyecto, quería agradecer a D. José Luis Muñoz Lozano su guía, dedicación, paciencia y colaboración para que este proyecto haya llegado a su término.

2.- Explicación del proyecto

Este proyecto trata de un estudio sobre el robot llamado K-Junior, fabricado por la empresa K-Team, la cual, se ha especializado en la fabricación de diversos tipos de robots de diferentes niveles. Todos ellos están centrados en el campo de la educación, y primeros acercamientos a la robótica, pudiendo ser utilizados por cualquier estudiante, desde estudiantes de bachiller a universitarios e incluso personas que no hayan estudiado nada sobre el tema. Se puede decir que este proyecto, es una continuación de otro encargado de la parte de control del movimiento del robot, a través de los sensores infra-rojos instalados en el frontal y parte trasera de éste; evitando, de este modo, los obstáculos que se le presentaran al robot mientras que éste se mueve libremente por un laberinto.

A continuación veremos varias partes del robot, tanto de la comunicación a través del bluetooth como del puerto USB al ordenador, hasta una comunicación más sencilla través de un infrarrojo en la parte superior del robot, que se utilizará para poder mandar datos cortos entre los diferentes robots, sin la necesidad de la participación ni del ordenador ni de un intermediario; además de poder controlar el robot a través del infrarrojo por un mando a distancia.

A este estudio, le sumaremos otro sobre una de las extensiones perteneciente a la caja básica de este robot, la cámara en línea, que funciona como un sensor óptico, formado por tres cadenas de valores que dependerán de la luminosidad del ambiente y de las características del material que está enfrente. Esta parte del estudio la describiremos más detalladamente en los siguientes apartados.

3.- Objetivos

Aquí veremos los principales objetivos que va a tratar nuestro proyecto fin de carrera. Desde un punto de vista técnico mostraremos a grandes rasgos lo que debe realizar nuestro robot:

1. **Estudio, comunicación y control a través de la comunicación por infrarrojos.** Con esto lo que deseamos es mostrar, tanto la comunicación entre dos robots como el control del robot por un mando a distancia, utilizando la comunicación de infrarrojos que incorpora el robot, a través de un sensor en su parte superior.

2. **Estudio y comunicación del robot con otro a través del bluetooth.**
   En este caso nuestro objetivo será realizar una red entre los diferentes robots, sin la necesidad de la participación del ordenador o de un usuario. Utilizando para ello el manual del fabricante de bluetooth y su firmware.

3. **Control del robot a través de la cámara.**
   Se tratará de realizar un programa donde podamos controlar el movimiento del robot a través de la cámara que incorpora éste en la caja básica, además de realizar un estudio de los materiales que se utilizarán.

Estos tres son los objetivos que se intentara cumplir a lo largo de nuestro proyecto y que tendremos en cuenta a la hora de elegir nuestra aplicación para nuestro proyecto.

4.- Estructura de la memoria

En este apartado mostraremos lo que podremos encontrar en las diferentes partes de esta memoria. Para empezar, podemos ver que la memoria está dividida en capítulos de una temática diferente, siendo así más fácil de entender por alguien que no haya trabajado en él y facilitando también la búsqueda de lo que se desea leer.

El primer capítulo, nos muestra el porqué hemos elegido este proyecto en la parte de motivación, una explicación de lo que vamos a realizar en el proyecto en la parte llamada de la misma forma (explicación del proyecto) y los diferentes objetivos que puede tener nuestro proyecto.

El segundo capítulo es una explicación sobre los temas de los que trata el proyecto. Este capítulo estará dividido en cada uno de los temas que tratamos desde la robótica móvil, que es la primera pregunta. El siguiente tema es la visión artificial y su aplicación en el robot. Y el tercer tema, será el de las comunicaciones en la robótica con respecto al K-Junior.

El tercer capítulo es un estudio del K-Junior, el robot que vamos a utilizar. Como en el anterior se dividirá en tres preguntas. Una primera sobre la información dada en el CD del fabricante. Otra pregunta sobre la información encontrada en internet. Y una última será una descripción detallada tanto del hardware como del software que utiliza el robot.

El cuarto capítulo se encarga de la búsqueda de la aplicación que podrá tener el robot en nuestro proyecto, es decir, lo que al final realizará. Esta vez está dividido en 4 bloques casi independientes. En cada bloque mostraremos los diferentes programas y pruebas realizadas y las diferentes aplicaciones posibles. También tendremos la elección de la aplicación.

El quinto capítulo, se trata de la descripción y realización de la aplicación que hemos elegido. También contiene todos los estudios que hemos hecho para llegar a la

aplicación, dividiendo cada estudio en una pregunta, el porqué de la aplicación y cómo debemos llevar a cabo la realización de ésta.

El último capítulo será nuestra conclusión donde expondremos lo que nos ha parecido el proyecto, los diferentes problemas que hemos encontrado y las diferentes soluciones además de mostrar una comparación con la siguiente versión del robot.

CAPITULO II: INTRODUCCION A LOS CAMPOS TRATADOS EN EL PROYECTO

1.- Robótica móvil

Como su nombre indica uno de los campos que vamos a tratar en este proyecto es la robótica móvil. Esta rama se especializa en dar movilidad a los robots, ya que, aunque muchos en la industria se utilizan en cadenas de montaje sin la necesidad de que tengan que desplazarse, hay otros que se utilizan para diferentes tareas como exploración, limpieza o cargas y que necesitarán de movilidad para poder realizar la tarea deseada.

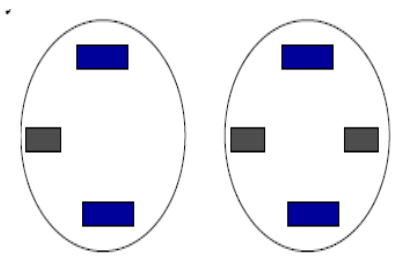
La primera gran distinción dentro de la rama de la robótica móvil, viene dada por su sistema locomotor, sacando de aquí dos grandes tipos:

LOCOMOCIÓN POR RUEDAS

Este sistema de locomoción es el más sencillo de los utilizados. Como su nombre indica, el sistema locomotor del robot estará basado en la utilización de un sistema de ruedas para poder moverse por el entorno. Este sistema da una mayor velocidad en terrenos llanos en comparación con el otro sistema.

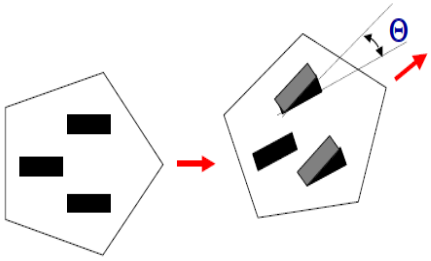
El sistema por ruedas también se puede dividir dependiendo del sistema de control y la cantidad de ruedas que contenga en:

- **Sistema diferencial:** Es el sistema más sencillo. Trata del control de dos motores diferentes uno para cada una de las ruedas (en el dibujo, los de color azul). Este control diferencial, le da al robot un centro de rotación instantáneo dentro de éste. Por lo que es capaz de girar sobre sí mismo. Los mayores problemas que pueden tener estos sistemas son, en primer lugar, la



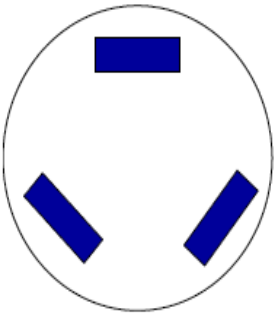
necesidad de que tengan los dos motores la misma velocidad para que puedan moverse en línea recta. El segundo es la estabilidad ya que el sistema de dos ruedas necesitará de otros puntos de apoyo, para ello utilizaremos unas ruedas multidireccionales (una o dos dependiendo de la estabilidad que deseemos), o como en nuestro caso, unos pequeños puntos de apoyo.

- **Sistema síncrono:** Se puede decir que es una caso particular del caso diferencial pues también utiliza dos ruedas, pero esta vez puede utilizar el mismo motor ya que las ruedas girarán y se moverán siempre hacia el mismo lado y con la misma velocidad, dando así una mayor simplicidad a la hora de construir y programar el robot, pero perdiendo la posibilidad de poder girar sobre sí mismo. Como en el anterior suele también utilizar una tercera rueda para estabilizarlo.



- **Sistema de triciclo o de coche:** Este sistema se caracteriza por el control de las tres o cuatro ruedas que puede tener el robot. Donde las dos ruedas traseras se utilizarán de tracción dando la velocidad al robot y la o las ruedas delanteras serán las que den la dirección al robot. Como en el caso anterior no podrá girar sobre sí mismo.

- **Sistema omnidireccional:** Se basa en la utilización de tres ruedas puestas de forma que cada rueda se encontrase en los vértices de un triángulo. Se caracteriza por tener menor dificultad al programarlo que el diferencial y una mayor estabilidad. También podrá girar sobre sí mismo a diferencia del síncrono, triciclo o del coche. Para movimientos rectilíneos utilizará dos ruedas a la vez mientras que para los giros utilizará las tres.



LOCOMOCIÓN POR PATAS

Es el otro tipo de locomoción el cual se caracteriza por la utilización de extremidades a diferencia de las ruedas. También se caracteriza por una mayor dificultad a la hora de la construcción, por la necesidad de un mayor control y una mayor cantidad de motores necesarios para su funcionamiento.

Hay una gran variedad de tipos dependiendo de la cantidad de extremidades que se tengan, por ejemplo: bípedos, cuadrúpedos y hexápodos entre otros. Los primeros antes nombrados son los de mayor complejidad por la necesidad de estabilidad y monitorización, mientras los hexápodos tienen una mayor complejidad a la hora de control por la mayor cantidad de motores. Pero éstos, presentan una mayor estabilidad dada por la gran cantidad de extremidades, haciéndolo estable sin la necesidad de control.

La mayor diferencia que existe entre este tipo de locomoción por patas con respecto al otro, es la posibilidad de moverse en terrenos más abruptos de lo que se podría mover con un sistema de ruedas, además de una mayor maniobrabilidad.

2.-Visión artificial

Otra parte importante de la robótica, es la toma de datos a través de los sensores de un robot. Con su uso, el robot puede saber lo que le rodea, ya sea para evitarlo, como los robots preparados para seguir laberintos o los limpiadores, o para seguirlo, como los robots que siguen una línea negra trazada, o los que interactúan directamente con lo que le rodea, como por ejemplo, todas las máquinas que pertenecen a las cadenas de montaje en una fábrica.

Uno de los sistemas de toma de datos y control de autómatas más importante, es el de visión artificial, el cual, se puede definir como "*un campo del control de autómatas que, mediante la utilización de las técnicas adecuadas, permite la obtención del procesamiento y análisis de cualquier tipo de información especial obtenida a través de las imágenes digitales. Se suele utilizar para procesos repetitivos donde el producto u objeto no se puede estudiar con otros métodos diferentes o que se necesite que no haya un contacto físico o reducir el tiempo de ciclo en procesos automatizados*". En nuestro caso, de los cuatro procesos que se tiene lugar, en la visión artificial utilizaremos tres: captación, procesado e interpretación de los resultados. Esto, será para analizar lo que se tiene enfrente, viendo la forma y si esta en blanco o en negro, para que el robot pueda parar a una distancia elegida por el usuario.

3.- Comunicación

La comunicación es un tema importante en la robótica, aunque no muy utilizado.
Tomando una de las posibles definiciones de robótica como *"la rama de la tecnología que estudia el diseño y construcción de maquinas, cuyo uso suele ser en tareas repetitivas o peligrosas para las personas",* según esto, podemos decir que esta rama no utiliza demasiado la comunicación, sino la programación de las máquinas con el proceso que deben realizar. Pero en el caso de ciertos procesos (sistemas de seguridad o producción de cierto objetos…) el robot necesitará comunicar cierta información, que

pasará desde este mismo terminal, hasta una centralita la cual se encargará de decidir qué hacer. Normalmente, esta centralita estará controlada por un ser humano.

En un principio esta comunicación se hacía a través de cables, con lo cual, se tenía que conectar todos los robots a las centralitas a través de ellos, produciendo así una limitación de no solo el número de autómatas que se podían conectar y controlar en una misma centralita, sino también la distancia máxima que podrían estar los robots con respecto a la centralita. Esto causaría molestias, ya que, al cruzarse los cables podrían confundir al operario y hacer que su manejo fuese mucho más difícil, no solo para cualquier operario de mantenimiento, que ya tuviese una idea de cómo se han conectado, sino también para cualquiera que no hubiese sido contratado durante el tiempo de la instalación.

Después apareció la conexión inalámbrica a través de pequeñas redes de bluetooth, cuya mejor característica consistía en la desaparición del componente cable, el cual es un aumento de las posibilidades de conexión. Aunque su distancia máxima no es muy grande, se puede utilizar en lugares donde la introducción de un cable hubiese podido ser un gran problema. El mayor problema que tiene este tipo de comunicación es que al tratarse de una comunicación tipo maestro y esclavo, aunque guarde todas las direcciones de los diferentes autómatas, solo se podrá conectar con uno por vez, por lo que se utilizará en sistemas donde su control no sea de forma continua sino discreta.

Por último, puede decirse que apareció la comunicación por WIFI la cual al ser totalmente inalámbrica y usar como base internet, no tiene limitación de distancia máxima ya que se puede controlar desde cualquier ordenador que tenga los programas necesarios y una conexión a internet. Esto permite conectarlo y controlarlo a grandes distancias. El único problema es la necesidad de un router con wifi cercano para que el autómata pueda conectarse a él, encareciendo así el producto final.

CAPITULO III: Estudio sobre el robot

Como su nombre indica, nuestra segunda etapa va a ser un estudio bastante detallado del robot (K-Junior). Este estudio se verá reflejado en dos partes esenciales, siendo éstas, la búsqueda de información sobre el robot y la descripción de los componentes tanto del hardware, como del software del K-Junior.

La siguiente parte de la etapa será mostrarnos una descripción detallada del K-Junior, ésta se podrá dividir en dos partes. La primera parte, será la del hardware donde se describirán todos los componentes que forman parte del K-Junior (el micro-controlador, memoria, puertos…entre otros). La segunda parte consistirá en una descripción de la parte software, donde explicaremos cuál es el programa que se usa para su programación, qué compilador y qué cargador utilizaremos, además de mostrar una pequeña descripción de las diferentes bibliotecas y programas de prueba que nos dará el proveedor para poder trabajar con el robot sin que nosotros tengamos que partir de cero.

1.- Análisis de la información suministrada por el fabricante

La búsqueda de información se realizará de dos maneras, ya sea a través del CD que el fabricante nos da con todas sus especificaciones o también, a través de internet donde, no solo se buscará información sobre el K-Junior, sino también sobre las posibles aplicaciones que podrá tener.

1.1.- Descripción del CD

En este CD, el distribuidor introduce tres carpetas, dos manuales, un glosario de las diferentes funciones y un esquemático con el circuito de conexión del hardware perteneciente al robot. La única molestia que se puede encontrar en este CD es que todo viene en inglés, y sin manual traducido, lo cual hace que se necesite una base de inglés. Aunque las palabras que se utilizan no son demasiado complicadas, si que pueden ser

un poco técnicas. Si se tiene alguna duda, siempre se puede utilizar los diferentes traductores gratuitos que existen por internet para facilitarnos la traducción, siempre comprobando si su funcionamiento es bueno y con ello ahorrarnos algo de tiempo.
Ahora pasaremos a describir cada parte de este CD de forma detallada.

1.1.1.- Kjos código fuente

Es la primera carpeta que nos viene en el CD, como su nombre nos indica en inglés, se trata del código fuente que nos proporciona el fabricante. Este código son bibliotecas que nos ayuda y facilita el trabajo de programar, ya que prepara e inicializa los diferentes puertos, sensores y tipos de comunicación del robot, además de darnos las diferentes funciones con las que se puede controlar todas las partes del K-Junior, ya sean los motores, los leds, e incluso la manera de comunicarse con él a través del ordenador o entre los robots.

Dentro también podemos encontrar otra carpeta llamada lib que trata de las librerías de las diferentes extensiones, además de los dos incluidos (cámara en línea y sensor de ultrasonidos), que se pueden comprar en la misma tienda. En nuestro caso, nos vendrá bien para utilizar las de ultrasonido y la de cámara en línea, ya que así podremos ahorrarnos el tiempo de programación necesario para estas dos piezas además de que comprobaremos cómo funciona y con ello resolver alguna duda si nuestra programación no resulta efectiva.

Además, se incluye una biblioteca llamada constants, donde están las definiciones de todas las direcciones de los diferentes pins que se tiene en la programación, dándoles nombres a cada uno dependiendo de las diferentes funciones que tienen. Otra, es la llamada variables que define las diferentes variables que crea el fabricante para la programación del robot desde unas variables para la velocidad de los motores, como otra para la comunicación por infrarrojos. También viene con una hoja de CCS preparada ya para trabajar en ella donde se incluye una pequeña cabecera siempre necesaria para trabajar con el robot y que utilizaremos para la creación de nuestros diferentes programas.

1.1.2.- K-Junior USB drivers

En esta carpeta se encuentra todas las bibliotecas y el código que se necesita para poder comunicarse el robot con el ordenador a través del USB. Con esto el robot gana rapidez en el paso de información desde el ordenador al robot y hace su uso más sencillo.

1.1.3.-TinyBootltloader

Esta carpeta contiene el programa que vamos a utilizar para cargar nuestros diferentes programas creados en el ordenador y pasarlos al robot. Lo primero que vemos, es que dentro tiene dos carpetas una de información y otra con el código fuente de PIC. Después nos encontramos dos archivos, uno de ellos con código fuente. Por último, encontramos el ejecutable, el cual, es lo que utilizaremos para la carga de los programas. Este ejecutable tiene varios tipos de variables, que ya explicaremos en la parte de la descripción del software, además con este programa no hará falta ningún adaptador más que el cable USB, que sería necesario si no se utilizase el programa cargador con el firmware del fabricante instalado en el robot.

1.1.4.- Esquemático

Es el siguiente archivo que se encuentra en el CD, es un PDF con tres páginas, las cuales contienen el circuito esquemático, que expondremos en la parte de descripción. Con esto veremos las diferentes conexiones que tiene el robot, y también las diferentes terminaciones que tiene en las salidas de hardware extensible.

1.1.5.- KJCS Funions Glosary

Este archivo es bastante interesante, ya que es un glosario con todas las funciones que el fabricante ha hecho para que su manejo sea rápido y sencillo. Si nos surge alguna duda de la utilización de las funciones podremos verlas en este glosario.

1.1.6.-KJOS Manual

Se trata de un documento Word, es un manual de las funciones. Como todo manual tiene un índice que nos muestra, de manera general, donde podemos encontrar la función que necesitamos, así como una pequeña explicación de su funcionamiento, cómo programarlo y un ejemplo de éste.

1.1.7.- K-Junior manual de uso

Este pdf es el último archivo que se encuentra en el CD, se trata de un manual completo del robot. En sus primeras páginas, además de un índice, encontraremos el apartado "***la introducción*** *al robot*", donde se puede destacar la parte llamada "precauciones de seguridad" que vendría a ser un pliego de condiciones donde se muestra las causas que podrían deteriorar e incluso romper nuestro robot.

El siguiente apartado, "***desempaquetado e inspección***" (página 9) como su nombre indica trata de ver si el contenido del paquete que hemos comprado es el correcto y si funciona. En su primera parte (contenido del paquete) nos muestra una pequeña lista de comprobación con el contenido de lo que tendría que venirnos al abrir el paquete. Su siguiente parte "primera puesta en marcha", se trata de un pequeño programa, ya introducido en el robot que nos servirá para ver si se encuentra en buen estado o si por el contrario es defectuoso, para ello primeramente pondremos en marcha el robot dando le a on/off y después en la parte de abajo del robot pondremos el primer interruptor a 1 o en marcha para cargar el programa. Se trata de un programa muy sencillo de evitar obstáculos, donde dejaremos al robot en un terreno y el deberá esquivarlos, a través de sus sensores instalados. Para ponerlo en marcha, se pondrán los interruptores como se indica en el dibujo y haciendo caso a las siguientes indicaciones reflejadas en el manual.
En el tercer aparatado "***El robot y sus accesorios***" (página 10) se verá más extensamente descrito en la segunda parte de la etapa 1 ya que se trata de la descripción física de todos los componentes y accesorios del robot que vamos a tratar. Este apartado en sí está dividido en tres partes. La primera parte "descripción global" que se trata de

dos dibujos uno de la parte superior y el otro de la inferior del robot y la señalización de todos los componentes del robot. En la segunda parte de este apartado *"El K-Junior robot"* es la parte donde se describe todo el hardware del robot, desde el micro-controlador, hasta los diferentes tipos de conexión que tiene y la batería. La tercera y última parte de este apartado se llama "*Accesorios*", nos muestra todo lo que el fabricante ha hecho que se puede conectar al robot, y una pequeña descripción, que realizaremos en la segunda parte de este mismo apartado.

En el cuarto apartado, llamado "***modos de funcionamiento***" (página 21) se trata de una descripción de los diferentes modos de prueba que vienen dados por el fabricante ya instalados. Lo primero que se nos dice es, que cuando programemos el robot, su programa será borrado, pero que éste se encuentra en la primera carpeta del CD. Además nos facilita una dirección, donde podemos ponernos en contacto con el fabricante para pedirle más información para solucionar nuestros problemas. Después, viene explicado el modo de control remoto, el cual, como su nombre indica, inicializa el robot de forma que está esperando las instrucciones desde los sensores IR o a través del puerto serie y no hará nada hasta no haberla recibido. Esta configuración, es también válida tanto con el puerto USB, como con el bluetooth, vía IR o través del webots-K-Junior. El siguiente modo, es el de evitar el obstáculo y evitar caer, como su nombre indica hace que el robot evite obstáculo y, si se encuentra en una mesa o en un lugar elevado, tampoco caiga de éste ya que se parará cuando se acerque al borde. También viene incluido el modo seguidor de línea, que lo que hace es buscar una línea negra y seguirla. En este y el anterior programa se debe llevar cuidado con la las luces fuertes, como las de las lámparas o el sol ya que puede dar problemas con los sensores. Otro modo, es el modo baile, donde el robot empezará a girar. En este modo el robot irá dibujando una especie de hipocicloide. El modo 5, es el modo de "estudio de ejecución", donde podremos cargar los programas previamente compilados en la memoria con el Bot-Studio. En los siguientes apartados de la pregunta encontraremos explicaciones sobre como conectar el ordenador a través del USB o bluetooth. Como penúltima parte de este apartado, es el modo control a través de un mando a distancia normal. En el manual, nos dice que se utiliza los mismos infra-rojos y codificación que son utilizados en muchos mandos a distancia de televisión, además, nombra los

diferentes botones que están programados con los distintos movimientos del robot. El último modo, es el de comunicación a través del modo IR emisor, que será de comunicación a corto alcance, como ya hemos indicado. Es en este modo donde se pueden comunicar los robots entre ellos. Se suele utilizar cuando no hay modo bluetooth, o simplemente se quiere enviar paquetes pequeños de información.
En este modo se puede controlar a varios, a través de uno de ellos. Para esto, tenemos que poner a cada uno de los robots una dirección de puerto diferente, para saber cuál es el que está mandado datos y cuál es el que los está recibiendo. Para evitar que la información se pierda, uno de los robots (el que recibe) se "desactiva" su modo de comunicación por infrarrojos mientras esté recibiendo el dato. Otra cosa que va a tener cuenta, es que la distancia entre los dos robots es directamente proporcional a la altura que se encuentra el techo, el material y el color del que está hecho, por lo que para esta aplicación también se deberá tener en cuenta esto, ya que con esto, podremos calcular la distancia máxima y mínima con la que se pueden comunicar entre sí los robots.

El quinto apartado se llama "**reprogramación del K-Junior**" (página 30) y como su nombre indica nos guía en la programación del robot. Lo primero que nos dice, es que tiene un programa cargador instalado (TinyBootloader) que nos sirve para cargar el código que hemos compilado al PIC. También nos dice, que podremos cambiar el fireware del robot, pero que no se hacen responsables de los daños que esto pueda acarrear a la memoria ni al robot. Lo siguiente en exponernos es que, el cargador se encuentra en la parte inferior de la memoria y que si esta parte de código es utilizada, se necesitará un programador del PIC, ya que se borrará el cargador de la memoria. En caso de problemas, se puede mandar otra vez al fabricante, pero esta reparación no está dentro de la garantía. Además, este apartado se divide en 3 preguntas más. La primera de ellas, trata sobre el "cargador" (el cargador de programas al robot) llamado Tiny bootloader, donde viene explicando para que sirve. Después, nos muestra paso a paso las diferentes etapas que tenemos que seguir para hacer funcionar nuestros programas. La segunda pregunta, encabezada por los nombres de compiladores CCS C, nos muestra, que podemos compilar nuestros programas a través del compilador llamado CCS, el cual, se trata de un programa compilador especializado para micro controladores de PIC, tal como el que tiene el K-Junior instalado. Este compilador al ser

de la familia de C (C++, visual C…) responderá a todos los comandos utilizados en anteriores C pero con el aliciente de tener unos comandos y bibliotecas especializadas en el funcionamiento de los microcontroladores de PIC. También nos dice que los códigos fuente entregados por el fabricante, están compilados a través de este programa y que aunque en un principio están protegidos contra modificaciones, se podrán cambiar para ser utilizados por el usuario con el uso de éste u otro compilador parecido. Nos dice que tienen una pequeña muestra gratuita del programa por si no lo encontramos y nos hace falta para la programación. En la última parte de este apartado, nos explica que se puede utilizar cualquier otra herramienta que PIC nos permita (ejemplo Mplab), mientras que se genere el archivo en hexadecimal, necesario para el cargador. El único problema sería tener que volver a crear todas las funciones básicas que nos da el fabricante. Además, nos avisan de que hay que tener experiencia en programar PIC, que no saben si servirá otro programador y recomiendan el uso del CCS.

La ultima parte del manual son los "**protocolos de comunicación**" (página 32), donde en este apartado nos dice qué teclas del teclado podemos utilizar para comunicar el robot con el ordenador, a través del USB o del bluetooth. Nos dice, que el código está hecho para admitir datos en ASCII y, por lo tanto, se utilizarán para nombrar las funciones a través de las teclas de letras y números, éstos serán complementarios a las funciones. Estas funciones vendrán descritas en la parte de software de nuestro proyecto.

2.- Búsqueda de información y aplicaciones del K-Junior.

La búsqueda de información ha sido un poco difícil, ya que, realmente no hay demasiada información sobre el K-Junior.

Sobre la información encontrada, ha sido bastante productiva. Por lo menos hemos conseguido tres pdf. El primero que nos encontramos, es un archivo pdf, que se llama *generación K-Junior*, el cual muestra los componentes, no solo los que vienen en el manual, sino otros como dos dispositivos conectores para poder programar el K-Junior. Este archivo también viene con una tabla con las especificaciones más "importantes" que se puede tomar (ejemplo: velocidad del robot, radio del bluetooth). El siguiente

archivo, es llamado "*primeros pasos con el K-Junior. Guía para escribir, compilar y cargar el código al K-Junior*" y, como su nombre indica, es una guía sobre la creación de tus primeros programas del K-Junior, con el cual podemos realizar un acercamiento a la manera de programar el robot, es más, nos explica del archivo que vamos a utilizar, sólo necesitaremos modificar el cuerpo de la programación (el main), ya que tiene todas las cabeceras necesaria ya escritas y todas las variables y constantes ya programadas. Nos recomienda que solo programadores experimentados cambien esas cabeceras. En estas páginas, también podemos encontrar, un enlace que nos unirá a una página de internet, la que nos mostrará más material como el ejemplo de evitar y reconocer los obstáculos, entre otros.

De la búsqueda de aplicaciones, lo único que hemos encontrado, es solo una pequeña muestra, que sería el código del programa para detectar obstáculos y desniveles, el cual nos puede venir bien como ejemplo de cómo hacer el cuerpo futuro de nuestros programas.

Por último, también hemos encontrado un manual sobre la cámara en línea que vamos a utilizar para la aplicación del proyecto y que explicaremos detalladamente en la parte de búsqueda de aplicaciones.

3.- Descripción del K-Junior

En este apartado complementario al anterior, tratamos la descripción tanto física (hardware) como de comportamiento y funcionamiento (software) del robot y, cómo y con qué programarlo. Por tanto, vamos a dividir este apartado en dos grandes sub-apartados, que como sus nombres indican trataran de mostrar todas sus características de una forma detallada.

3.1.- Hardware.

Esta parte tratará con todo detalle lo que podemos estudiar en las partes físicas del robot, tanto de sus componentes por separado, como en conjunto mostrando sus

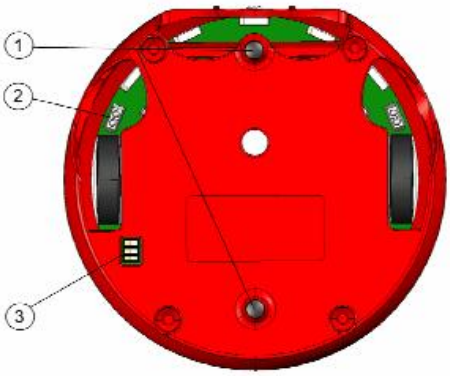
esquemáticos. También incluirán una descripción de sus accesorios; las más detalladas serán las que vienen incluidas en el robot.

3.1.1.- Descripción general

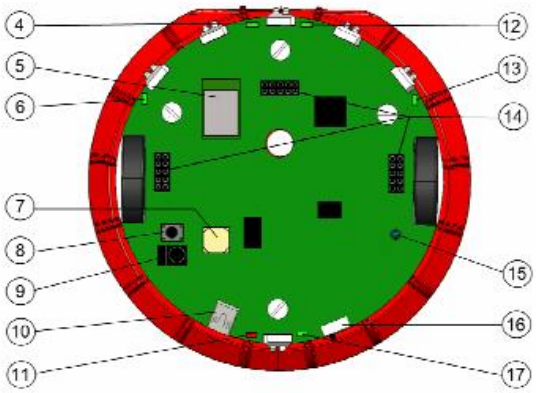
En primer lugar realizaremos una pequeña descripción general del robot, lo que podremos ver a simple vista (ejemplo sensores, interruptores, conectores, ruedas, leds entre otros).

Para ello, ponemos dos dibujos del robot, uno por la zona superior y otro por la zona inferior. A través de flechas y números, pondremos nombre y función a lo que le corresponde a cada cosa.

Zona inferior



Zona superior



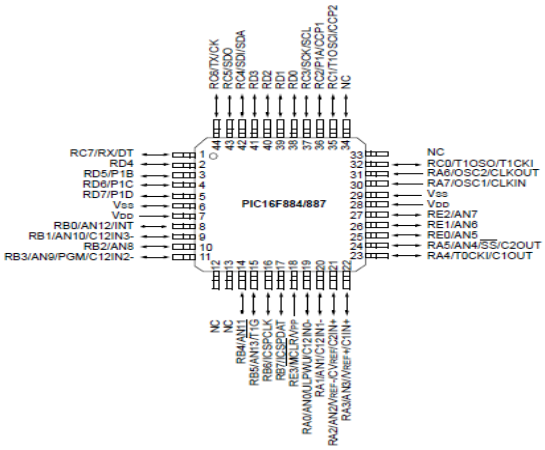
1. **Contactos de tierra**: son unas pequeñas esferas que están delante y detrás del robot que se utilizan para que el robot no arrastre.
2. **Sensores de proximidad IR**: son los que se utilizan para detectar obstáculos e incluso desniveles o límites en el terreno.
3. **Interruptores de modo de selección**: son los interruptores que utilizamos para elegir el modo de funcionamiento, se encuentran en la zona inferior y se necesita de un pequeño destornillador o cualquier otro elemento para acceder a ellos.
4. **Led verde frontal**.
5. **Módulo bluetooth**: es el módulo que necesita el robot para realizar la conexión del robot al ordenador a través del bluetooth.
6. **Led verde izquierdo**.

7. **Zumbador**: es el instrumento con el que el robot produce sonido y dependiendo de la intensidad de corriente que le llegue hará un sonido u otro.
8. **Botón de reset**, utilizado tanto para producir un pequeño apagado como para la carga del programa.
9. **Receptor remoto de IR**: módulo con el que el robot recibe información de infra rojos ya sea de un mando a distancia como de otro robot.
10. **Conector de Mini USB**: módulo con el que conectar el cable USB.
11. **Led indicador de carga**: es un led que nos indica que está conectado al USB, se pone de color rojo y carga las baterías del robot.
12. ,13. **Led's verdes.**
14. **Conectores de extensiones**: es donde irán conectados todos los accesorios y circuitos que deseemos ponerle al robot. Se encuentra en la parte superior del robot.
15. **Led emisor IR**: se trata de un led especial que se utiliza para mandar datos a través de infra rojos a otros dispositivos como otros robots del mismo tipo.
16. **Botón on/off.**
17. **Led puesta en marcha**: led que marca que el robot está en funcionamiento.

3.1.2.- Microcontrolador PIC 16F887

El microcontrolador, se trata del centro de control del robot, donde cargaremos nuestros programas. Este circuito integrado está compuesto por un procesador, una memoria y unos periféricos ya sean de entrada digitales como de entradas analógicas.

El microcontrolador que se utiliza en el robot K-Junior, se trata del PIC de 16F887 de 44 patillas. Éste, se caracteriza por ser un micro-controlador de 14 bits, el cual solo tiene 35 palabras para programarlo. Esto hace que sea fácil de programar. Es uno de los más potentes microcontroladores de 14 bits que tiene PIC, que es una empresa



especializada en la creación de tanto micro-controladores como micro-procesadores. Además, tiene un oscilador interno funcionando como reloj, que va desde 8Mhz a 32Khz, en nuestro caso utilizaremos el de 8Mhz. La tensión con la que puede funcionar va desde una tensión muy baja, los 2 voltios a 5.5, el K-Junior queda en una tensión intermedia, aún así bastante baja, de 3.3 voltios. Otras caracteristicas que nos pueden venir bien saber, es el tipo de memoria que se utiliza par el almacenaje, son del tipo flash, teniendo 368 bytes de RAM y 256 de ROM. Además tiene 14 canales con 10 bits de convertidores analógico-digital, 4 puertos de 8 bits, 2 comparadores internos, 3 temporizadores, dos de ellos de 8 bits (TIMER0 y TIMER2) y uno de 16 (TIMER1) que puede funcionar como temporizador o como contador, entre otros componentes.

3.1.3.- Led's

LED es la abreviatura en lengua inglesa para Light Emitting Diode, y son diodos que se caracterizan, como su nombre indica, en que cuando esta polarizado el material semiconductor que hay en su interior, el cual se encuentra separado en unos milimetros, produce un arco de luz que es visible. Este pequeño arco, es lo que vemos como luz. Una de sus características es que no produce calor como sería en el caso de una bombilla normal. El color que adquiera la luz emitida por este dispositivo dependerá de los materiales semiconductores utilizados en la fabricación de éste.

Son 6 los led's que van ensamblados al robot, 5 son programables y uno se utiliza como indicador de la bateria. Automáticamente, en la puesta en marcha del robot el led 4 parpadeará mientras que los demás están apagados. También tiene un modo de comprobación del funcionamiento de los leds programables, con un programa que los hace parpadear automáticamente. Por último, solo indicar que el led de carga se pondrá rojo cuando se conecte el cable USB, y en cuanto el robot esté cargado el led se apagará.

3.1.4.- Sirena

Es un módulo piezoeléctrico incluido en el robot. Estos piezoeléctricos se caracterizan porque cuando pasa una pequeña corriente por estos materiales cambian su tamaño en unos pocos milímetros por lo que pasando una corriente alterna puede hacer que el material vibre y como su nombre indica, funcione como un tipo de sirena cuya característica principal es, que trabaja en un ancho de frecuencia que será desde los 20 Hz a los 2 KHz. Ésto se controla a través de una función, donde, dependiendo la cifra que introduzcamos, recorrerá las frecuencias (con el valor 0 la sirena se apaga mientras que con el valor 1 emite un sonido de 20 Hz y con el valor 100 los 2 KHz). Con ello, se puede poner simples melodías al robot, siempre teniendo las notas dentro de la frecuencia que pueda producir esta sirena. Para saber qué tono producirá por cada valor podemos utilizar la siguiente fórmula:

$$\text{Frecuencia de sonido} = \frac{2000}{101 - \text{valor dado}}$$

3.1.5.- Módulo de control remoto y comunicación por IR

Se trata del módulo con el que se puede, no solo controlar el robot con un mando de televisión cualquiera, sino con el que podemos hacer una comunicación entre dos diferentes robots, con ello se puede enviar pequeñas cantidades de información entre ellos, utilizando el mismo tipo de código, el RC5, el mismo que utilizan los mandos a distancia. Nosotros seremos los que elijamos los datos y las direcciones que queramos enviar o recibir de los diferentes robots.

Por la forma de utilización el IR tiene dos formas de comunicación a través de éste:

Control remoto de IR a través de un mando

Como ya hemos dicho antes, al utilizar el código RC5 se puede utilizar los mandos a distancia que utilicen este protocolo (los cuales suelen ser los de televisiones de tubo, los otros, como por ejemplo de DVD o de televisiones planas, tienen diferentes códigos

y no servirán), aquí mostramos los botones y códigos instalados por defecto por el fabricante:

- Botón 1: Se mueve hacia la izquierda en un círculo grande.
- Botón 2: Mover hacia delante.
- Botón 3: Mover hacia la derecha en un círculo grande.
- Botón 4: Gira sobre sí mismo (sentido antihorario).
- Botón 5: Detener.
- Botón 6: Gira sobre sí mismo (a la derecha).
- Botón 7: Igual que el 1, pero se mueve hacia atrás.
- Botón 8: Se mueve hacia atrás.
- Botón 9: Igual que el 3, pero se mueve hacia atrás.

Control de comunicación IR, control de comunicación a corto rango

Es la otra parte de la comunicación en el IR, se puede utilizar cuando falle el sistema de bluetooth o para mandar bytes de datos pequeños. Su primer protocolo es el mismo que con el mando, pero se puede cambiar haciendo nuestros propios programas. Hay que ponerle diferentes números a los puertos de cada robot para que se puedan diferenciar entre ellos y que puedan mandarse los datos entre ellos, para eso hay que tener en cuenta que mientras que manda datos, no puede recibirlos, y viceversa, ya que es la misma función la que se encarga tanto de la recepción de datos como del envío de los mismos.

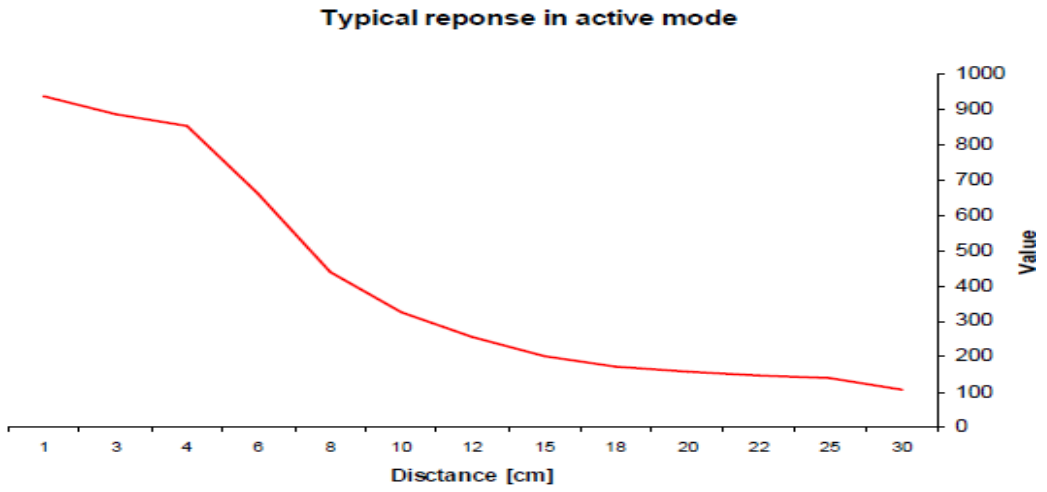
Por último, hay que tener en cuenta que los datos se mandan a través del emisor IR colocado en la parte superior del robot, haciendo que los datos salgan hacia el techo, por lo tanto, se debe tener en cuenta el tipo de techo que es, tanto el material, como el color, la textura y la altura a la que está, para saber la distancia máxima a la que van a poder comunicarse entre los robots. Esta función tiene un área cónica cuyo centro será el del robot que envía datos.

3.1.6.- Interruptores

El robot viene con 4 micro interruptores. Uno de ellos está situado en la parte superior, el cual, se trata del botón on/off y sirve para poner en marcha al robot. Luego, en la parte de abajo, tiene tres interruptores en serie que sirven para poder ir controlando los diferentes modos de funcionamiento del robot que vienen predeterminados, dependiendo del interruptor que pongamos a nivel alto. Se podrá utilizar además en nuestros programas como forma de elección o modo de introducción de una variable etc.

3.1.7.- Sensores infra-rojos

El robot tiene diez sensores infra-rojos instalados repartidos de la siguiente manera, cinco en los laterales encargados de tomar información de alrededor del robot y cuatro en el parte de abajo del robot para no solo utilizarlos para seguir una línea sino también para hacer que el robot no se caiga si se encuentra en una superficie elevada. El último se encuentra en la parte superior y es el encargado de la comunicación por infrarrojos. Cada sensor ésta compuesto de dos partes. Un Led emisor que manda una señal infra-roja. Un fototransistor el cual dependiendo de si recibe la señal se activará con una intensidad mayor o menor. También se sabe que estos sensores pueden funcionar de dos maneras diferentes. Una de forma pasiva, en la cual se mide a través del fototransistor la cantidad de infra-rojos que hay en el ambiente. Otra de forma activa donde, el emisor enviará una señal infra-roja, ésta se perderá si no hay un obstáculo cerca, pero si el objeto ésta suficientemente cerca el haz rebotará y será recibido por el fototransistor con lo que el robot, sabrá que hay un obstáculo y su localización para poder esquivarlo.
Hay que tener en cuenta, que para que el robot lo detecte a través del infra-rojo será mejor cuanto más claro, ya que el porcentaje de señal que será devuelta será mayor, también dependiendo del material del que esté hecho. Con los sensores de abajo se puede seguir una línea o detectar los bordes de donde esté. También hay que tener en cuenta que no se puede poner en zonas donde le pueda dar mucho el sol ya que esto puede llevar a los sensores a error.



3.1.8.- Módulo de comunicación Bluetooth

El robot tiene un módulo bluetooth llamado WT12 de bluegiga, el cual, se trata de un módulo de bluetooth 2.0 + EDR. Mientras esté alimentado, está esperando información de la comunicación con el ordenador u otros módulos de bluetooth. En esta comunicación se podrá ver desde el ordenador un puerto serie llamado "K-Junior 1234" el cual el número de serie era el mismo que el del nombre. Una vez establecida la comunicación, se puede controlar el robot a una distancia de hasta 20 metros, a través de los comandos de protocolo serie. Se podrá conectar hasta siete robots a la vez, pero solo controlar uno por vez. También se pueden conectar entres ellos haciendo una red sin necesidad de un ordenador. Esto se hará a través del iWarp que es un firmware instalado en el módulo WT12, pero que en el manual sólo pone la página en la que debemos buscar si deseamos realizar la red de robots.

Control de bluetooth

1. A través del puerto USB, configurar el K-Junior en modo de control remoto.
2. Ver a través del led si el robot está o no cargado y si no lo está cargarlo.
3. Conectar el bluetooth del ordenador y buscar señales de bluetooth.

4. Ahora debe aparecer la señal del robot como si fuera una PDA con solo el servicio de puerto serie, con el nombre y el número ya descrito antes.
5. Se establece ahora la conexión con el robot. Nos pedirá que introduzcamos un código el cual será “0000”
6. Su equipo le asignará ahora un puerto COM.
7. Ahora abre una terminal para el puerto COM
8. Vuelva al K-Junior y pulsa el botón de reinicio
9. Con esto ya se puede enviar los mensajes al robot utilizando un programa UART como es el de Hiperterminal.

3.1.9.- Módulo de Comunicación USB

El robot lleva integrado un chip de adaptación de comunicación serie a USB, el cual se llama RS232, necesario para la utilización del cable USB ya que el micro no podría comunicarse directamente con este tipo de puerto. Lo describiremos mas adelante en la etapa de elección de aplicación en la parte de comunicación por USB.

Además de poder conectar el robot al ordenador a través de un mini-USB, también sirve para cargar la batería del robot. Se utiliza más para la actualización del software y carga de programas, no para su funcionamiento en situ, ya que su limitado tamaño dificulta el campo de funcionamiento del robot en sí.

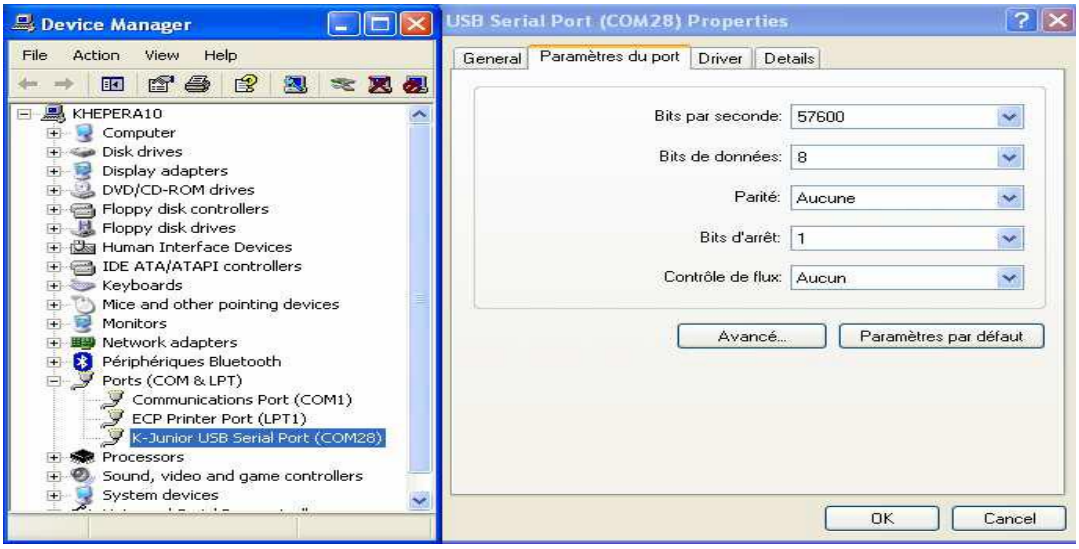
Ahora pasaremos a describir los siguientes pasos que hay que dar para poder realizar la conexión del robot con el ordenador.

Control de comunicación USB

1. Conectar el cable USB-mini-USB al ordenador y al robot.
2. Seleccionar el modo remoto del robot.
3. El robot será visto por el ordenador y le pedirá un controlador. Se deberá seleccionar el que se suministra en el Cd llamado K-JuniorUSBdrivers.zip, en

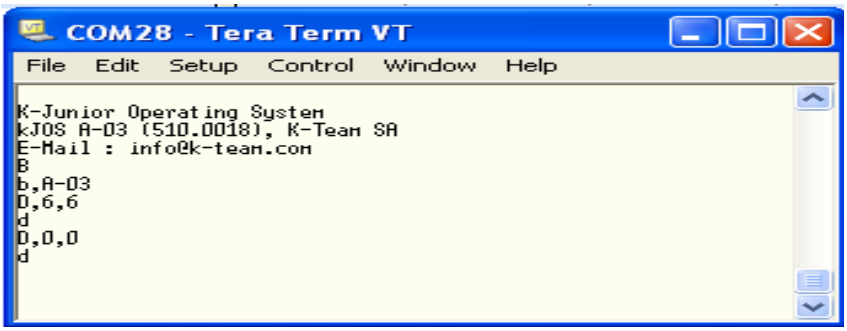
nuestro caso la carpeta de K-JuniorUSBdrivers y hay buscar los drivers correctos.

4. Ahora será visto el robot como un puerto COM.
5. Tenemos que dar los valores de velocidad correctos al puerto a través del controlador
6. Abrir el administrador de dispositivos hardware, y en la K-Junior dispositivo, seleccionar propiedades, y en la configuración de puertos configuramos los parámetros, como se ve en la imagen.



7. También se necesita un programa de UART como Hiperterminal, para poder comunicarse el robot. Este programa antes nombrado nos servirá a nosotros para nuestras aplicaciones.
8. Vemos cual es el puerto COM correcto para no tener problemas (esto se puede ver en el administrador de dispositivos hardware) y luego tenemos que configurar el hiperterminal:
    - 57600 bytes por segundos.
    - 8 Bits de datos.
    - 1 bit final.
    - Sin control de flujo.
9. Pulse el botón de reset.

10. Si se ha configurado bien el K-Junior, saldrá una ventana con un mensaje.



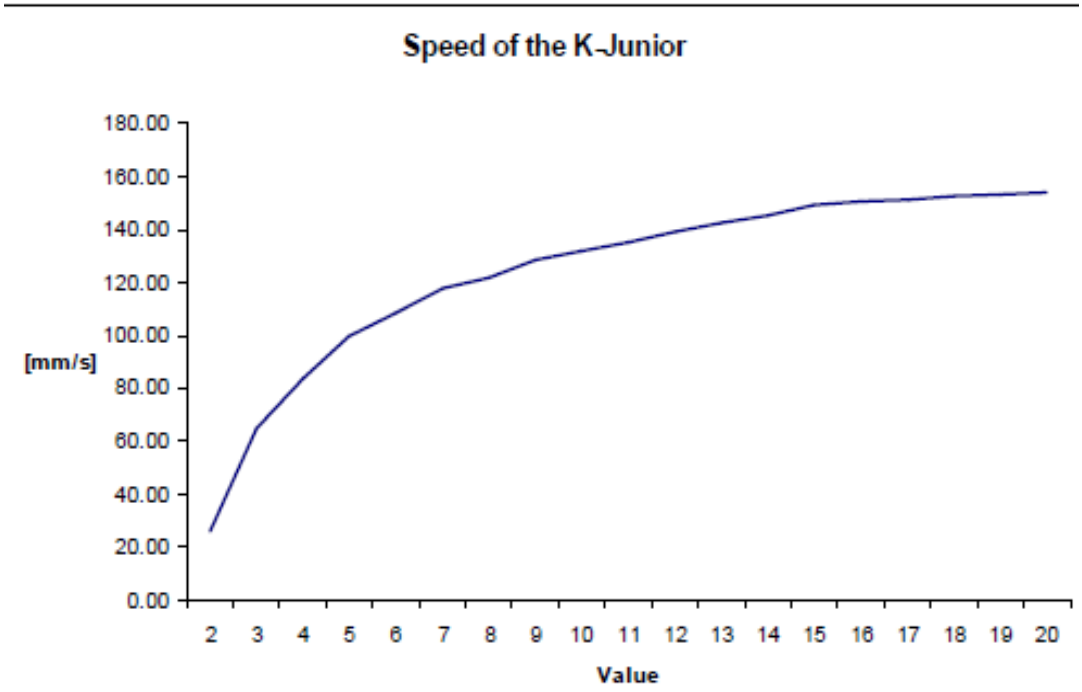
11. Ahora se podría probar mandando los protocolos de comunicación.

3.1.10.- Batería

Se trata de una batería de li-pol que nos da unos 3,7 V y unos 1250mAh. Esto le da una autonomía al robot de unas 4 horas de duración y se carga a través del cable mini-USB. Se verá si el robot está cargado y si el led rojo se apaga. Como se indica en el manual, si el robot o la batería se rompieran, no se deberían tirar simplemente a la basura sino que se deberían dejar en sitios especializados, ya que la batería contiene partes muy contaminantes y perjudiciales para el medio ambiente.

3.1.11.- Mecanismo accionamiento

Se trata del mecanismo que activa los motores que tiene el robot para moverse. Es del tipo diferencial, por lo que es mejor que otros sistemas, ya que, con este sistema lo que se hace es controlar la velocidad relativa de los dos motores por separado, y con ello, podemos hacer que gire el robot sobre sí mismo. Tiene una velocidad máxima de 0.16m/s



3.1.12.-Accesorios

En este apartado nombramos a los accesorios, que podemos definirlos como complementos que vienen con el robot y que no van conectados fijos a los puertos del robot. Además, explicaremos los que hagan falta:

- Cable USB-miniUSB.
- Rotulador: se da un rotulador que encajará en el hueco que tiene el robot y sirve para marcar toda la trayectoria del robot.
- Lápiz para configurar los interruptores: pequeño trozo de plástico en forma de lápiz que se utiliza para poder tocar los interruptores internos del robot, se debe tener cuidado para no romper los interruptores o el propio lápiz.

3.1.13.- Extensiones

Son un tipo de accesorios, los cuales se conectan a los puertos que se encuentran en la parte superior del robot. Dependiendo de que tipo de extensión sea, y como se comunique con el robot, se deberá conectar en uno de los tres conectores:

- El conector frontal se utiliza para la comunicación I^2C y la alimentación de estas extensiones.
- El conector izquierdo se utiliza para la posible ampliación de memoria del robot. Siendo en este conector donde se colocaría la nueva memoria RAM.
- El conector de la derecha se utiliza para una fijación mecánica (como en la de la pinza.)

También nos dice que podremos encontrar más información de los conectores en los esquemáticos de los apéndices B y C.

En la caja nos vienen dos de serie, que serán los que explicaremos a continuación, los otros simplemente los nombraremos.

- **Hemlincam**: Se trata de un módulo óptico que se utiliza para que se pueda ver, en blanco y negro los perfiles de algunas formas que haya enfrente del robot. La cámara nos muestra una cadena de 102 pixeles, divididos en tres zonas una izquierda, una central y otra derecha. Cada pixel es capaz de diferenciar una cantidad de 256 niveles de grises, desde 0 (negro) hasta 255 (blanco). Además, el módulo de la cámara lleva un PIC integrado utilizado para la representación visual.
- **HemUltrasoncisensor**: Se trata de un módulo de ultrasonido el cual nos sirve para poder localizar obstáculos con una distancia de hasta 6 metros con una precisión de 1 cm de error.

Otras de las extensiones son:

- **HemgenIO**: Es una placa que nos sirve para aumentar en número de pines para utilizar y por tanto nos aumenta la cantidad de hardware que se puede conectar.
- **HemTextoSpeech**: Módulo que se utiliza para convertir un texto en sonido y palabras. Esta placa transforma cualquier palabra escrita en ASCII a su equivalente en sonido.
- **Hemgripper**: Se trata de una garra que se puede acoplar. Está formada por dos motores, ésta tiene un sistema de alimentación independiente, ya que para los

motores se necesita una tensión que la alimentación que el robot no puede proporcionar. El peso máximo es de unos 30g y una anchura desde 1 cm a 7 centímetros. Se tendrá en cuenta que la autonomía del robot bajará una a dos horas.

- **Hemwirelescam**: Es otro tipo de micro-cámara, la cual se diferencia primero en que es a color, segundo, que tiene instalado un bluetooth para poder utilizarlo con la televisión, ya que también te viene un módulo para conectarlo a la televisión y poder controlarlo.

3.2.- Software

En este apartado, veremos mucho más detallado, cómo y con qué programar, así como cargar los programas con el cargador que nos da la empresa fabricante del robot. Esta parte, como en la anterior, se dividirá en 4 partes.

3.2.1.- CCS

Es el lenguaje de programación que nos recomienda el fabricante, ya que, siendo un lenguaje que viene del lenguaje C, C++, es muy fácil de manejar, con instrucciones sencillas. Es mejor que el ensamblador, porque con una sola de esas instrucciones de CCS, equivaldría a muchas líneas de código en el ensamblador. Como en los lenguajes C, es lo que se diría un lenguaje intermedio, porque como hemos dicho tiene comandos de tipo palabra, más complejo que el ensamblador, pero también puede operar al mismo nivel simple que operaría el ensamblador.

El fabricante además nos dice que este C está especializado en los micro-controladores de la marca PIC. Por un lado su ensamblador, el PCW, está preparado para las diferentes familias (12, 14, 16, 18F….), teniendo todas las bibliotecas con las funciones, interrupciones y patillas de cada micro. Por tanto, para hacer nuestras aplicaciones del robot, tendremos que tener cierta familiaridad con el lenguaje CCS. Aunque es un lenguaje sencillo y fácil de encontrar en sus versiones de prueba, encontrar uno completo y que nos sirva puede llevarnos algo de tiempo, además como todos los

lenguajes necesitaremos saber cómo funcionan sus librerías y las instrucciones especiales.

Lo primero que debemos hacer, es copiar en la carpeta donde vayamos a trabajar, la carpeta de kjossource, donde están todas las bibliotecas y códigos que vamos a necesitar. Segundo paso, abrir el programa de CCS, que se encuentra en la carpeta PIC. Dentro ya del programa, damos a la barra de proyecto, donde abrimos el archivo de proyecto llamado Kjos.pjt, en el cual utilizaremos de base para nuestros programas. El cuarto paso, es darle a view y hay que darle a una pestaña llamada program files, el cual nos mostrará todos los componentes de nuestro proyecto. En el quinto paso, se abre el llamado Kjos.c, es donde escribiremos nuestro programa. Cuando hayamos terminado de escribir nuestro programa, buscamos la barra Compiler, donde lo primero que hacemos es pinchar en tarjeta de chip, deberemos elegir la opción de PCM 14 bits, luego pulsaremos el botón "lookup part", donde en su primera opción buscaremos el micro del robot, el PIC 16F887. En el último paso, después de haber seguido todos los demás pasos, solo habrá que darle al botón de "build all" dentro de la barra "compiler", el cual servirá para compilar el archivo y si no hay ningún error se crearán dos archivos. Sólo nos interesará el archivo Kjos.hex, que será el que se cargará en el robot.

2.2.2.- Tiny bootloader

Se trata de un firmware dado por el fabricante, donde parte de este programa está cargado ya en el robot y se encarga de que podamos cargar el código que nosotros creamos en el ordenador al robot. Como se ha dicho ya varias veces, se debe llevar cuidado en qué lugar se guardan nuestros programas en la memoria del robot, ya que podemos borrarlo, y con ello, no se podrá volver a programar a través del USB sino que se necesitará un cargador de PIC. Este tipo de cargador nos costará unos 200 euros o así. Otra opción sería mandándolo otra vez a la empresa donde lo compramos y que lo reprogramen, claro está, pagando una tasa que nos dirá el fabricante, además de la pérdida de tiempo correspondiente. Como es normal en estos casos, la empresa no se hace responsable de este tipo de problemas por un mal uso de sus productos.

Modo de uso

1. Conectar el K-Junior con el mini-USB al ordenador.
2. Encender el K-Junior.
3. Ejecutar el tiny Bootloader.
4. Configurar la velocidad en baudios a 19200.
5. Seleccionar el COM correcto y si no aparece, darle al botón de search.
6. Cargar nuestro programa (debe ser el terminado en .hex).
7. Mantenga pulsado el botón de reset del K-Junior.
8. Haz clik en el botón Write Flash soltar el botón de reset del K-Junior.
9. Después de haberse cargado, nuestra aplicación empezará automáticamente.

2.2.3.- Protocolo de comunicación

En este apartado, nombramos un protocolo creado por el fabricante, el cual, se puede utilizar cuando el robot se encuentra conectado al ordenador a través del USB o a través del bluetooth para controlarlo de una manera rápida y sencilla. Este protocolo se trata de simples letras y, cuando sea necesario, utilizar un número separado por una coma. El fabricante ha configurado el protocolo, de manera que cada letra tiene una función. Para terminar, después de cada instrucción, debe introducirse un salto de línea para que se mande al robot y que éste realice las diferentes funciones que dependerán de cada letra.
Las instrucciones se pondrán en un anexo contiguo con una pequeña explicación de cada función.

2.2.4.-Otros compiladores

Aunque en el manual se recomienda la utilización del compilador de CCS, no es el único programador que puede utilizarse. Por ejemplo, el de Visual C, que es otro tipo de programador de C, pero más especializados en la utilización del ordenador, ya que tiene funciones específicas para pantalla que no tiene el CCS. Otro posible lenguaje de programación es el que suministra la propia empresa de Microchip (MPlab), que es el básico para la utilización de micros del fabricante PIC. Pero, aun así, el compilador recomendado para trabajar en el robot será el de CCS, ya que lo utiliza el mismo fabricante y, como ya hemos dicho, es fácil de manejar.

CAPITULO IV: Análisis de los módulos del robot

Esta es la parte central del proyecto, donde se decidirá qué aplicación es la que va a realizarse en el proyecto de una manera más concreta. En este apartado como en el anterior se podrá dividir en dos partes.

Las aplicaciones que vamos a elegir para su estudio son los diferentes tipos de comunicación, empezando por la comunicación IR, la cual, es una comunicación que puede realizar entre los robot. Esto hace que sea muy interesante, aunque sea de pequeños datos y de una distancia que aunque dependerá de la altura nunca podrá ser mayor que la que sería la del bluetooth. Otra cosa importante de esta comunicación, es que se pueda controlar el robot a través del mando a distancia.

Comunicación por IR

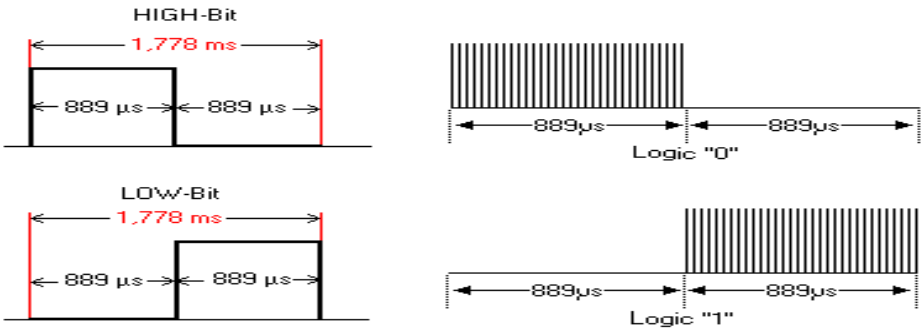
Lo primero a tener en cuenta en esta comunicación, es la forma de mandar y recibir la información a través del sensor de infrarrojos que tiene en la parte superior del robot. Para ello realizaremos un estudio sobre la codificación RC5, el cual, es el tipo de codificación que se ha elegido para la comunicación. Como ya se ha dicho también se trata del protocolo estándar que se había utilizado en la comunicación a través de mandos a distancia de las televisiones de tubo como se mostrará en las pruebas hechas.

1.-Codificación RC5

Se trata de un modo de codificación muy sencillo. La información que se envía está formada por 14 bits los cuales se dividen en, tres de cabecera para comprobar que funciona bien, luego tiene cinco bit para la dirección del aparato, en nuestro caso el del robot, y por último seis bit para mandar el dato que se desea mandar.

La codificación en sí, trata de enviar un 1 a través de unos trenes de pulsos. El primer bit que manda será un 0 durante unos 889 microsegundos y luego se pondrá a enviar un

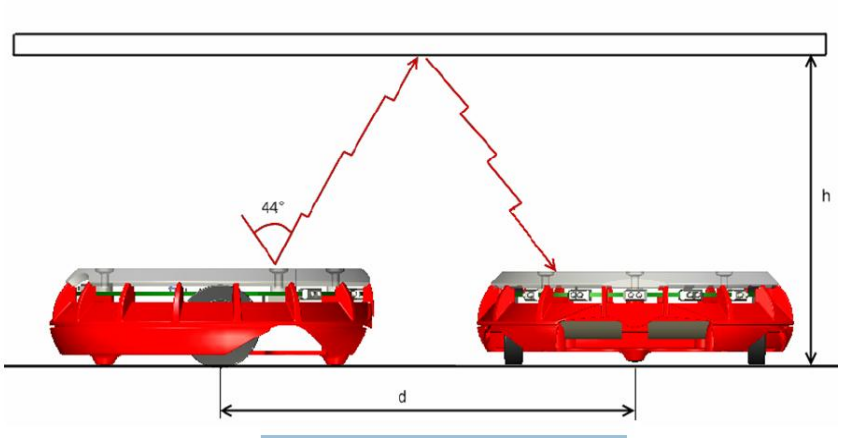
tren que cambia de 1 a 0 durante el mismo tiempo que el anterior. Además si el bit cambia a mitad del tiempo de 0 a 1 seguirá mandando un 1. Si al contrario, se quiere mandar un 0, primero mandara el tren de pulsos de 1 a 0 durante 889 microsegundos y luego se mandará un 0 continuo otros 889 microsegundos, también pasa como en el caso anterior, si en mitad del tiempo, el bit pasa de 1 a 0 entonces se tomará como un 0. Como se ve, es una codificación muy sencilla, la cual, se puede utilizar para mandar unos bytes no muy largos, donde el más largo es el byte de dato que tiene 6 bits, a corta distancia como ya se ha dicho.



Como se muestra en el dibujo ésta es la codificación de una forma gráfica.

2.- Aspectos a tener en cuenta

La primera condición se presenta en la situación del receptor-emisor del robot, como hemos dicho antes, se encuentra en la parte superior del robot, y por tanto la distancia mayor a la que el robot podrá comunicarse, dependerá de la altura del techo, color, material del que está compuesto, y por último la forma que tenga, ya que no devolverá la señal con la misma intensidad en un techo liso o un techo rugoso o de color más claro o más oscuro. Esto hará que la señal devuelta pueda ser vista a mayor o menor distancia incluso que con ciertos techos no se refleje y no se pueda utilizar.



También deberemos tener en cuenta el estado de la batería, ya que, una batería casi gastada puede producir que la señal mandada o recibida no sea la correcta, pues el nivel de señal dependerá también del estado de la batería. Las pruebas se realizarán siempre o casi siempre con las baterías llenas.

Después, para realizar el trabajo de programación que nos corresponde debemos tener en cuenta la biblioteca y la hoja de C que nos proporciona el fabricante con todas las funciones y los programas de control de los sensores ya realizados. Por ello, cogeremos las funciones que nosotros necesitemos y las explicaremos aquí.

-KJunior_init (): Es las primera de todas las funciones a tener en cuenta a la hora de trabajar con el robot. Se trata de una rutina de inicialización que se explica en la hoja de C K-Junior. Esta rutina se encarga de inicializar y configurar todos los puertos que tiene el micro del robot, además de configurar el convertidor analógico digital con un oscilador interno el cual también lo configura (a 8 MHz). Inicializa todos los valores de los sensores a 0, además de su mapa de pines correspondiente de donde se introduce un valor del sensor, dándole así un nombre sencillo y entendible por cualquiera, a través de la función que va a realizar, en vez del nombre de un pin. Por último, inicializando las interrupciones que necesitaremos para poder empezar a funcionar.

-KJunior_config_auto_refresh_sensors (1 o 0): Función que sirve para decidir si los sensores toman el valor automáticamente (1) o de una forma manual (0). Si no se configura, en principio será de manera automática.

-KJunior_config_auto_refresh_tv_remote (1 o 0): Este sirve para configurar el receptor de televisión y comunicación para que se refresque de manera automática (1) o de manera manual (0). En principio, si no se configura se hará de forma automática.

-KJunior_flag_tv_data_refreshed (): Es Flag el cual se activará solo si el robot ha recibido un nuevo dato a través de la comunicación de televisión. Esto, nos puede servir para hacer nuestro proyecto, como por ejemplo utilizarlo en una pregunta para que haga algo cuando tenga un nuevo dato. Para poner el flag en 0 se tendrá que utilizar la función de KJunior_flag_tv_data_reset ()

-KJunior_flag_tv_data_emitting (): Este flag se activará solo si está activo el emisor de televisión, por lo que podemos utilizarlo para hacer otra pregunta en nuestro programa, como que mientras se esté emitiendo, se enciendan unos leds. Esta función necesita la misma función anterior para poder poner a 0 el flag de dato emitido.

-KJunior_flag_tv_data_reset (void): La función ya nombrada, se utiliza para poner en 0 todos los flags utilizados en la comunicación.

-KJunior_get_switch_state(): Se utiliza para sacar el estado de los interruptores que tiene el robot incorporado.

-KJunior_get_tv_data (): Esta función se utiliza para sacar el último valor que se haya introducido en la variable data.

-KJunior_get_tv_addr (): Esta función es como la anterior pero se diferencia en que en vez del data se saca la variable addr.

-KJunior_send_tv_value (addr, data): Es la función utilizada para la transmisión de datos entre los robots por infrarrojos. En ella, se ve como en principio se introduce los valores en dos variables para enviarlas (TV_Data_send, TV_addr_send). Después, pone el flag tv_data_emiting en 1 y espera mientras que esté recibiendo información del otro

robot. Cuando se termina, prepara la interrupción del TIMER2 para que salte a los 886 microsegundos, desconectando a su vez la función de interrupción externa. Además pone a 0 el contador de tv y el flag del contador que más adelante veremos para qué sirven.

Cuando salta la interrupción del TIMER2 se introduce en la interrupción llamada RC5_Decoding_Interrupt

- TV_Remote_Interrupt(void): Este es el nombre de la interrupción exterior, que se activará cuando se active el sensor de comunicación IR, o como en nuestro caso se desactive, entrará en la interrupción. En ella, lo primero que hace, es preguntar si el flag auto_refresh_tv_remote está a 1, si es así, pone el tv_counter a 0, mete en tv_table[0] el valor contrario del pin IR_in, después desactiva la interrupción externa y activa la del TIMER2 poniendo su contador a 0 y activando su interrupción, por último pone el flag tv_Data_pending a uno.

-RC5_Decoding_Interrupt(void): Es la interrupción que nos da el fabricante para la codificación y decodificación del código RC5, el cual como hemos dicho antes, es el mismo que utilizan los mandos a distancia de muchas televisiones. Además, utiliza la misma tanto para la toma de datos como para enviarlos.

Lo primero que hace es crear un flag llamado RC5_bit que se utilizará más adelante, en la parte de envío de datos. Después, pregunta si tiene un dato por recibir, viendo si esta a uno el flag llamado TV_Data_pending. Si es así, prepara el TIMER2 para que salte a los 1,778 milisegundos aproximadamente, lo que se tarda en enviar un período completo de la señal, o mejor dicho lo que se tarda en mandar un 0 o un 1 con la codificación RC5. Luego incrementa el valor del contador_tv y pregunta si es menor de 13, si es así, introduce el valor que tenga en la patilla llamada Ir_in, que será la que esté conectada con el sensor y lo guardara en el arrays llamado TV_table. Si el contador es mayor de 13, entonces comprueba si los valores de los dos primeros bit del arrys de tv_table [] que se ha recibido son correctos (iguales a 1), entonces, si son correcto se introducirán los valores, en tv_addr y en tv_data, recibidos. Debemos de

tener en cuenta que los valores se introducen bit a bit, de forma que el valor de tv_addr corresponderá a los bits del 3 al 7 (4 bits) del arrays tv_table []. Lo mismo pasa con el nuevo valor de tv_data, que serán desde el 8 al 13 (5 bits). Luego, comprueba si su flag tv_control está activo (a 1). Si es así, pasa a un menú de selección llamado tv_control, el cual, moverá el robot dependiendo del valor que se le haya dado. Lo malo es, que si se activa de esta manera como se explicará en las próximas hojas, no serviría ponerle un control a través de los sensores que tiene alrededor, por lo que si no se lleva cuidado, el robot puede darse algún golpe fuerte. Si el flag de tv_control está a 0 seguirá con el programa, el cual, desconectará la interrupción del TIMER2 y conecta la externa o ext. También para el contador del TIMER2 y conectar el TIMER1 y su interrupción. Luego, se resetea el intcon para evitar posibles interrupciones falsas y pone a cero el flag_tv_pending para mostrar que ya ha recibido el dato.

Si no está pendiente de dato y está emitiendo dato, o mejor dicho si el falg tv_Data_emiting esta a 1, entonces prepara para que TIMER2 salte a los 886 microsegundos, que corresponde a cada una de las dos partes del periodo total necesario para enviar un dato. Después, mientras que el flag tv_data_loop se mantenga a 1 hará:
Lo primero, poner ese flag a cero, también pone a 0 el contador del TIMER2. Luego encadenara una serie de preguntas, empezando por si el tv_counter es menor de 3, si es así, entonces pone el flag RC5_bit a 1, sino pregunta si el tv_counter es mayor de 13 si es así entonces, pone el flag tv_emiting a 0, activa la interrupción exterior y la del TIMER1, desconectando la del TIMER2 y limpia el flag de interrupciones del puerto b, el flag tv_counter_flflag a 0 y el flag RC5_bit a 1.Sino, sigue preguntando el tv_counter es mayor o igual a 8 y menor de 13, entonces hace otra pregunta más, la cual, es que si el bit correspondiente al dato tv_data_send esta a 1, que será el mismo que el del counter menos 8 (va desde el bit más alto al más bajo), entonces pone el RC5_bit a uno y, sino, lo pone a 0.Sino, si el bit correspondiente del tv_addr_send al counter menos 3 está como el anterior, va del bit de mayor peso al de menor, está en 1 entonces pondrá en 1 el flag RC5_bit sino, estará en 0. Lo siguiente que hace es preparar la codificación Manchester teniendo en cuenta que si RC5_bit es 0 lo que se quiere mandar es un 0 y viceversa, y para ello volverá a encadenar una serie de preguntas empezando por si el tv_counter_flag y el RC5_bit están a 0, si esto sucede, es

porque se quiere mandar un 0, que corresponde a un tren de pulsos durante 886 milisegundos más un 0 otros 886 milisegundos. Para ello lo primero que hace es mandarlo a una pequeña rutina llamada KJunior_IR_Send32x(), el cual se encarga de el tren de pulsos que pondrá a uno la patilla IR_out durante 13 milisegundos y a 0 durante 8 milisegundos y esto lo repite unas 32 veces que sería lo que dura el tren de pulsos. Después, vuelve a la interrupción, inceremenata el contador pone el flag tv_counter_flag a 0, mantiene el flag tv_data_loop a 1 pone a 0 el TIMER2 y resetea su flag de interrupción. Sino entra en la anterior, y los dos flag anteriores están a uno hará lo mismo que el anterior pero esta vez, para mandar un 1, sino entra en ninguna anterior entonces al tv_counter se le suma el valor del flag tv_counter_flag y luego ese mismo flag toma su valor contrario, por último pone a cero la patilla IR_out. Con esto ya se tiene toda la codificación de manera que si se manda un cero se pone RC5_bit 0 y el flag tv_counter_flag también está a 0 por lo que entra en la primera pregunta. En primer lugar, manda un tren de pulsos y cuando sale manda 0 el resto del tiempo. Si manda uno pondrá un 1 en RC5_bit por lo que no se podrá meter en la primera pregunta y al tener el otro flag a 0 pasa a la tercera parte donde el tv_counter no aumentaría pero pondría en uno el flag y mandaría 0 durante los 886 milisegundos. Al siguiente paso se meterá en la segunda pregunta ya que tiene el RC5_bit y el flag a 1 con lo que mandará el tren de pulsos.

Por último, cuando sale del bucle desactiva la interrupción del TIMER2 activa la del ext y resetea el flag encargado de las interrupciones del puerto b.

3.-Funciones propias:

Aunque el fabricante nos haya dado las funciones necesarias para la codificación en RC5, nosotros vamos a realizar unas funciones propias, las cuáles, unas se utilizarán para la misma codificación, y otras para la simplificación de la utilización del robot. Además para algunas funciones debemos crear unas variables que ahora nombraremos:

-int16 F,FL,L,FR,R,D,GL,GFL,GR,GFR : éstas son las variables que se utilizan para la toma de datos de los sensores de proximidad. Son enteros de 16 bits ya que suelen ser valores un poco largos.

-int1 flag = 0, led_lf,led_lr,led_l, led_r, receptor=0, emitir: se tratan de unos bits, los cuales, se utilizan como marcas para ciertas preguntas en las funciones que vamos a crear, también se utilizan los de led para poner a 1 o a 0 los valores de los leds que incorpora el robot.

-int8 tr[13]: es el arrays que se utiliza para almacenar los valores que se reciban a través del IR de televisión, después de decodificar el RC5. Tiene 13 bites ya que son trece los valores que son mandados a través de la codificación de RC5.

-int8 dato, puerto, dato_env, puerto_env: éstas son las variables que se utilizarán para la comunicación de datos, el de dato para tomar el dato mandado. El de puerto para tomar el valor de la dirección enviada. Los de dato_env y puerto_ env son las variables que se utilizarán para enviar los datos

A continuación, mostraremos las funciones que hemos ido programando:

-Void refresh (): hemos introducido aquí todas las variables de los sensores justamente cuando se están metiendo los datos de los sensores. Esto simplemente se hace igualando la variable con la función de obtener el dato del sensor, que corresponda a la variable elegida.

Ejemplo de una de las funciones: F= (KJunior_get_proximity (Front)), así con todas las otras funciones.

-Void led_com(int, int, int, int,): se trata de una función simple que, con una frase podemos conseguir conectar o desconectar los 4 leds y no tendremos que gastar más líneas de código.

-Void led_out (): se trata como la anterior de una función utilizada para disminuir las líneas de código y poner todos los leds a 0.

-Void comunic(void): es la que utilizaremos con las subrutinas que nos proporciona. Se trata de una función creada, la cual desconecta la función de control_tv dada por el fabricante, luego utiliza la función KJunior_send_tv_value(int8,int8), con la que mandamos la dirección y el dato a través del IR. Después simplemente preguntamos si el falg TV_Data_Available es igual a uno a través de su función antes nombrada. Si es así, conectará los motores para que se mueva hacia delante. Si no es así, simplemente se quedará parado, desconectando si son necesarios los motores, esperando. Por ello se le añadió una señalización visual que sería que se encendiesen los led de la izquierda y luego los de la derecha para luego apagarse.

-Void controltv (): función creada para poder realizar el control del robot a través de un mando a distancia normal. Es una ampliación, a la que nos da el fabricante. En este caso ponemos a 0 la función KJunior_config_tv_remote_control (0) la cual se encarga del control remoto hecho por el fabricante. Después, se saca el último valor del dato que se ha recibido por IRC a través de la función KJunior_get_tv_data (). Después, utilizamos un swith case con la variable de elección TV_DATA. Dependiendo del valor que tiene, tomará una dirección u otra como se indicará más adelante. La mayor diferencia es la introducción de los sensores laterales para el control de posición, ya que, en el que nos da el fabricante no se utilizan, ya lo explicaremos un poco más adelante. Con esta introducción de los valores de los sensores a través de la función refresh(), se pregunta si los valores de los sensores son muy altos, y con ello si tienen un obstáculo en su camino. Si hay un obstáculo, lo hemos programado para que gire hasta que no encuentre ningún obstáculo y cuando esto haya pasado, se pare a la espera de un nuevo dato que se transmita por el infra rojo que se utiliza para la comunicación.

Las siguientes interrupciones son las que se crean para la comunicación por IR, las más importantes son:

-Void pimer_interrupt(void): se trata de una interrupción exterior, la cual, será activada cuando RB0 se ponga a nivel bajo. Lo primero que hace es pasar un valor contrario, al que haya en la patilla RB0, también llamada por el fabricante IR_IN, a la cadena de valores que hemos creado con el nombre tr[], en este caso tr[0]. Después desconectamos la interrupción externa y preparamos la del TIMER2 para que salte un poco antes de lo normal. También incluimos la función led_com() para revisar si entraba o no en la función. Esto lo hicimos al principio ya que al probarlo inicicalmente no realizaba ninguna cosa. Por lo que introdujimos que se encendiese un led de comprobación si se metía en esta función y otro led si salía de la función, pero siempre se quedaba en esta función, por lo que después de meterse, tuvimos que desconectarla y preparar la del TIMER2, la cual se encarga de la decodificación del protocolo recibido.

-Void deco_interrupts(void): se trata de la interrupción de la decodificación y toma del dato. Se conecta cuando salta el TIMER2, en ella utilizamos un bucle for el cual, estará introduciendo los valores que se mandan por el infrarojo en la cadena de valores tr[], empezando por el valor 1 hasta el último valor, el 13. Además, tiene un retraso temporal de unos 1,778 milisegundos, que corresponde a la suma de los dos semiperiodos con los que se manda el dato. Después, si los dos primeros bits (tr[0] y tr[1]) recibidos son 1, entonces introduce los valores de dato y dirección en las variables dato y puerto. Teniendo en cuenta claro está que, cada valor de tr[] debe ser introducido en su lugar moviendo los correspondiente bits (ejem dato= tr[8] << 5 ya que el valor de tr[8], equivale al 5 bits de la variable dato). A continuación, llamamos a la función select, dentro de la interrupción ya que si se pusiese fuera, dentro de la misma hoja de C, no sería llamada ya que siempre se está introduciendo en las interrupciones y por tanto nunca serán llamadas, ya que cuando se desconecta la interrupción exterior se conecta la del TIMER2, esto también afecta a la toma de datos de los sensores los cuales son tomados con la interrupción de tiempo TIMER1 y como nunca se llama nunca se toman. Es por ello que en la función de tv_control no puede tomar los valores de los sensores, ya que su función está en la misma hoja C donde están las interrupciones. Después simplemente se desconecta la interrupción del TIMER2 y se vuelve a conectar la interrupción externa a la espera de que se mande un nuevo dato.

4.- Pruebas realizadas con la comunicación IR

Aquí mostraremos las pruebas que se han realizado y las posibles soluciones y aplicaciones que se nos presentan.

1º Programa de control

Este programa no presentó problemas, ya que, se hizo a través de las funciones que nos dio el fabricante. Para ello, simplemente se creó una función específica llamada controltv (). Ésta, solo se encargaba de, en primer lugar, apagar la función de control por mando que venía dada por el fabricante. En segundo lugar y más importante, recargar el nuevo valor de la variable tv_data, que es el dato recibido a través del mando. Por último, lo mandamos a otra función llamada elec(), ésta se encargará de elegir, dependiendo del valor que tiene la variable tv_data, lo que va a realizar el robot. Esto, se encuentra dentro de un bucle infinito, consiguiendo que se esté leyendo el último valor que ha recibido, eligiendo qué va a realizar de una forma continuada hasta que se cambie de valor o se apague. De esta forma, utilizando el mismo esquema que el que nos daba el fabricante, los valores serán:

- 1: Se mueve hacia la izquierda en un círculo grande.
- 2: Mover hacia delante.
- 3: Mover hacia la derecha en un círculo grande.
- 4: Gira sobre sí mismo (sentido anti horario).
- 5: Detener.
- 6: Gira sobre sí mismo (a la derecha).
- 7: Igual que el 1, pero se mueve hacia atrás.
- 8: Se mueve hacia atrás.
- 9: Igual que el 3, pero se mueve hacia atrás.

Así se quedó en un primer momento, pero al ver que el robot chocaba en la pared, y no tomaba ninguno de los valores de los sensores de alrededor; se hizo una modificación en la función de elec() introduciendo en todos los valores necesarios, los que producían

un movimiento no estático que pudiese acarrear un choque del robot. En estas partes de la función, se introdujo una nueva función llamada refresh (), utilizada para refrescar o, mejor dicho, recargar con los nuevos valores de los sensores laterales y frontales utilizados para notar cualquier tipo de obstáculo y no tener que escribir la toma de datos de cada uno de los sensores, así ahorrando unas cuantas líneas de código. Si los datos recibidos de los sensores llegaban a un determinado valor o los sobrepasaban, el robot giraba sobre sí mismo hasta que ya no detectase ningún tipo de obstáculo y se detenía a la espera de un nuevo valor mandado por el mando a distancia. Esta parte nos supuso un poco más de tiempo en hacer un par de pruebas para ver la distancia, pues si era muy pequeño el valor elegido, supondría que giraría con el menor signo de obstáculo, mientras que si este valor era demasiado grande entonces el giro no lo haría a tiempo.

Al final se utilizó el mismo valor que el fabricante utilizó para cuando detectaba un obstáculo y ha funcionado bien. Con lo que ya tenemos el programa de control remoto del robot a través del infra-rojo de una manera sencilla y segura.

Continuamos el primer programa, ya que se nos pidió si era posible, introducir todos los botones del mando. Para ello, lo primero era comprobar cuál era el valor de cada uno de ellos. Al principio se quiso hacer un programa, el cual, mostrase ese valor a través de los leds incorporados por el robot, pero al no conocer cuales podían ser esos valores, y al pensar que, estos valores podrían ser muy altos, y no podrían mostrarse con los pocos leds que tiene el robot se cambió de opinión. Por lo que se dio otra solución, comunicar el robot con el ordenador a través del programa hiperterminal, mostrando el valor recibido por la pantalla del ordenador y así pudiendo ser utilizados. Con esto podemos mostrar los diferentes valores que nos da el mando. Después se continuó con el programa anterior de control por mando, dándole diferentes opciones. A continuación se muestra los valores con ciertas funciones ya creadas para ellos:

- 12: power: si se pulsa el robot pasará a un estado standby, es decir, hasta que no se vuelva a tocar no hará nada.
- 13: silencio: como indica corresponde al botón de silencio por lo que apaga el altavoz del robot.

- 15 i^+: muestra por pantalla el valor de los motores y el sonido emitido en frecuencia.
- 16 vol+: como se indica aumenta el volumen en 10 cada vez que se toca.
- 17 vol-: disminuye en 10 cada vez que se toca.
- 20 música: produce una nota musical, solo 3 notas por ahora.
- 21 leds: coche fantástico, los leds van encendiendo de uno a otro.
- 32ch+: aumenta la velocidad en 1.
- 33ch-: disminuye la velocidad en 1.
- 38 temporizador: apaga el sonido y empieza a dar vueltas durante 4 segundos, después vuelve al último movimiento y produce el último sonido.
- 41encuadre de pantalla: va encendiendo los leds de dos en dos cada vez que tocamos, hasta tener todos encendidos.
- 43 alto de pantalla: va haciendo el contrario que el anterior, apagando los leds de dos en dos hasta apagar los dos últimos.
- 44.
- 56 av.
- 60 teletexto.

2º Programa de control

Este programa llevó mucho más tiempo que el anterior, ya que, se empezó de cero, nos pareció una buena manera de aprender cómo se recibía o se mandaba un dato a través del infrarrojo y de forma codificada.

Así pues, lo primero que se hace es mirar el código que nos daba el fabricante para recibir un dato o enviar un dato. También se hizo una búsqueda por internet de información, sobre el tipo de codificación con la que se manda el dato a través del mando a distancia. Es el RC5, que se nombra en el manual, pero que no se explica nada más de él, ya que, como ya hemos dicho antes, el fabricante ya había hecho el programa necesario para su manejo.

Después de ver cómo funcionaba, y, ver como lo había hecho el fabricante del robot, se empezó a hacer, creando una nueva hoja C para trabajar. A continuación, creamos unas cuantas variables que son necesarias para poder tomar los valores enviados.

Lo primero que hace el programa, después de prepararlo todo, es crear una interrupción exterior, que se conecte cuando el receptor de infrarrojo recibe una primera señal. Ésta se introduce en la cadena de valores llamada tr [], en su primer valor. Luego, se prepara la interrupción de tiempo, llamada TIMER_2, para que salte después de un período completo. En cuanto pasa, se introduce en la interrupción de tiempo, en donde se introducirá todos los nuevos valores recibidos desde el mando a la cadena de valores ya mencionada. Cuando ya se han recibido todos los valores, se comprueba si la comunicación es correcta. Si es así, los valores recibidos pasan a las variables anteriormente creadas. Después, se llama a la función elec, la cual se encargará, como ya dije en la descripción del anterior programa, de la elección de la acción del robot.

En este programa se tuvo muchos más problemas que en el anterior, como por ejemplo, el cómo se recibe los datos, la programación del período de tiempo que se necesita para el envío de cada uno de los bits del dato, que sería de unos 1,778 microsegundos. Después, dio un pequeño código erróneo, que nos indicaba que tenía unas interrupciones duplicadas, ya que las mismas interrupciones, las del TIMER_2 y la exterior, estaban tanto en la página en la que estaba trabajando, como la que estaba unida a la biblioteca que nos da el programador. Por lo que tuvimos que, dentro de la propia biblioteca, comentar las partes del código que no nos hacían falta.

El código no dio error, pero, no por ello se consiguió el resultado deseado, puesto que, al principio, el robot no hacía nada. Por lo que, se introdujo en las funciones el que se encendiese uno u otro leds, dependiendo de, si estaba en una u otra interrupción. Viendo así, que se quedaba siempre en la primera interrupción. Para no perder tiempo, en cargar el código cada vez que hacia un cambio, se decidió utilizar el programa Proteus. Un programa, que se utiliza, no solo para realizar los diferentes diseños de circuitos electrónicos, sino que también viene con un potente simulador, que se utiliza a modo de pruebas con los códigos creados con ciertos lenguajes, como el que utilizamos. Ésto nos

fue muy práctico, ya que, nos facilitó observar donde tenía el fallo, a través de una de sus variantes, ya que una de las opciones de la simulación es que lo puede hacer paso por paso, viendo el código mientras se simula, y así ver donde puede estar el fallo.

Una vez descubierto donde estaba el error, solo tuvimos que solucionarlo. Después, cargamos el programa en el robot, y se comprobó que funcionaba. Así fue. Pero, lo malo, es que aunque le habíamos metido la misma función llamada elec, la cual, debería haber sido capaz de detectar los objetos que están a su alrededor, la realidad, es que esto no sucedía, ya que, esta vez el programa utiliza directamente las interrupciones de tiempo y exterior, por lo que no se puede introducir en la del TIMER_1 puesto que siempre está, o introduciéndose en la interrupción exterior, o en la del TIMER_2 y por tanto no puede tomar los nuevos datos dados por los sensores, haciendo así que no se pueda saber, si hay obstáculo o no y con ello choques.

También se comprobó que pasado cierto número de veces, que al introducirse un nuevo valor a través del infrarrojo, éste daba error e introducía un código erróneo. Por tanto, como última modificación, se introdujo un pequeño bucle, en el que, se borrasen los valores que hay en la cadena de valores tr[].

3º Programa de control

En este mini programa, es el que utilizamos para realizar la prueba de comunicación a carga máxima y carga mínima. Este programa se trata del que ya hemos descrito antes de comunicación entre robots donde, los dos robots están todo el rato mandando los datos quietos y si uno recibe un dato se pondrá en movimiento hasta que pierda el contacto con el otro robot. Con este programa podremos ver cuál es la distancia máxima que podrá tener esta comunicación en lo que sería un entorno de una casa normal (altura de 3 m techo blanco).

Las pruebas a hacer son las mismas para los dos ya que tienen el mismo tipo de sensor los dos robots.

Máxima carga:

Distancia máxima recibiendo información del mando: 6,5 metros.

Distancia mínima: ninguna.

Carga media:

Distancia máxima recibiendo información del mando: 6 metros.

Distancia mínima: ninguna.

Carga mínima:

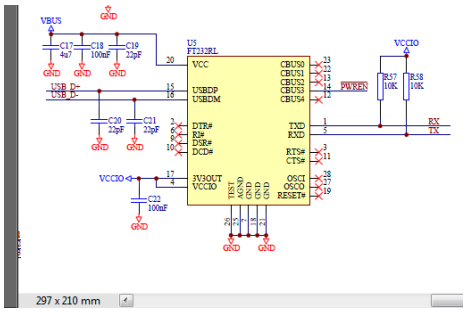
Distancia máxima recibiendo información del mando: 5,4 metros.

Distancia mínima: ninguna.

Comunicación por USB

Se trata de una comunicación sencilla, con la que podemos controlar el robot a través del ordenador y dar diferentes valores a sus variables a través de un cable USB, para lo cual hemos utilizado los diferentes comandos e interrupciones dados por el fabricante.

Una cosa que se ve es que, aunque el micro está fabricado para usar una comunicación RS232. Éstos serán los mismos parámetros que necesitará un chip llamado FT232RL, el cual, transforma el código de comunicación RS232 en el protocolo USB y la comunicación asíncrona.



Éste está conectado a las patillas RC5 la patilla RXy RC6 la patilla TX. Así cuando recibe algún comando tomado por el chip FT232RL lo convierte y lo manda por las patillas del micro haciendo activar la interrupción int_RDA, la cual se encarga de la comunicación.

Podemos usar las funciones que nos da el fabricante:

-serial-interrupts: Se trata del nombre de la interrupción INT_RDA, es una interrupción que como antes se ha dicho, se activa cuando recibe un dato en código

ASCII. Lo primero, crea una variable llamada flush, que utilizaremos para guardar el dato recibido del ordenador momentáneamente. Lo siguiente que hace es preguntar si el flag serialcomandok esta activo, si es así introduce el valor que hemos tomado a través del hiperterminal con la función getc en la variable. Si el flag serialcomandok esta a 0, entonces el valor del dato se meterá en el arrays llamado serialbuffer[] con la posición que corresponda al valor serialcounter. Si el valor introducido corresponde a los valores 13 o 10 entonces pondrá el serialcomandok a 1. Para descartar que el valor del serialcounter es bajo, se comprueba si este dato puede ser inferior al del SERIAL_BUFFER_SIZE menos uno y si es así, lo incrementará. Con esto termina la interrupción.

- **SerialCommandHandler:** Se trata de otra función que nos da el fabricante. Ésta se caracteriza en estar dividida en dos partes. La primera, es si el valor del arrays serialbuffer[] corresponde a las teclas de coma, salto de línea o retroceso. Si es así entonces simplemente pone a cero el serialcounter y el serialcomandok. Si no es así, pone un switch case, que se elegirá a través del serialbuffer[0]. Donde los datos coincidirán con su valor en ASCII. Por último, vuelve a poner los valores de serialcounter y serialcomandok a 0.

Hemos creado una función sustituyendo la de la interrupción y la de serialCommandHandler. Aunque hacían sus funciones, no nos servirán si se utiliza con la extensión cámara en línea. Estos detalles se verán mas reflejados en su parte de la búsqueda de aplicación. Solo decir, que es una aplicación corta que carga los datos introducidos por teclado a través de hiperterminal a una variable llamada F y que utilizaremos más adelante en nuestras pruebas.

Comunicación por bluetooth

Introducción

En este apartado tratamos, no solo la comunicación de bluetooth, sino también cómo configurar esta comunicación para poder conectar varios robots entre ellos, sin la

necesidad de un ordenador que esté en medio o la otra comunicación, que sería la poco fiable comunicación por infrarrojos, la cual, solo puede mandar datos cortos y a una distancia que sería proporcional a la altura del techo, pero que no sería demasiado grande, como por ejemplo, en una altura normal de una casa, la cual suele ser de 3m, sería como mucho unos 6 m de radio. En cambio, una red de bluetooth entre los robots sería de unos 10m a 20m entre cada uno de ellos, por tanto será de unos 4m más entre cada uno de los robots.

Por todas estas prestaciones, consideramos en hacer una red con los robots a través de bluetooth, como aplicación del proyecto fin de carrera. Esta aplicación sería un sistema de alarma donde un primer robot detectaría si hay o no una intrusión, si es así buscaría al segundo robot y le mandaría un mensaje a éste. El primer robot volvería a sus coordenadas iniciales, mientras el segundo robot iría hacia donde está la centralita y cuándo la encontrase, le mandaría un mensaje diciendo la zona de intrusión. Para ello, nos dice el fabricante que tendremos que acceder al firmware que viene instalado en el módulo de bluetooth del robot, llamado Iwrap que se utiliza para modificar todas las características que tiene el módulo de bluetooth, ya que para comunicarse, lo primero que tendremos que hacer es que uno de los módulos de bluetooth de los robots vea al otro que, de una manera normal, no podían verse.

Cómo funciona el bluetooth

Estandar FHSS

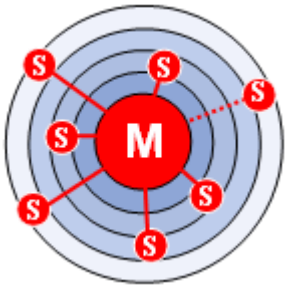
El bluetooth funciona como un WIFI a pequeña escala. Éstos utilizan la técnica de FHSS (en español Espectro ensanchado por saltos de frecuencia) es una técnica de modulación en espectro ensanchado, en el que la señal se emite sobre una serie de radiofrecuencias, aparentemente aleatorias, saltando de frecuencia en frecuencia sincrónicamente tanto el emisor como el receptor. Realmente lo que hace, es que transmite parte de la información en una frecuencia y luego salta a otra frecuencia de una manera pseudo-aleatoria, que solo conocerán tanto el emisor como el receptor, de manera que si algún otro receptor, por casualidad, está en la misma frecuencia que el

emisor solo recibirá parte del dato, como un ruido. Es una forma segura de transmisión, ya que, solo los que estén conectados de manera directa son los que tendrán lo que sería el código de frecuencias que cambiarán en cada caso.

La técnica FHSS utiliza la banda de los 2.4Ghz (desde la 2.402 hasta 2.480, ambas incluidas) para ir saltando entre ellas de manera que crea 79 canales, creando saltos de un 1Mhz. Como ya hemos dicho antes, solo el receptor y el emisor serán los que conozcan en que salto está. También, hay que tener en cuenta que salta a cada canal unas 1600 veces por segundo. Esto hace que le sea fácil evitar todas las señales de radio y no se oiga ruido si hay un aparato de este tipo cerca de una radio.

Principio de comunicación

El bluetooth basa su modo de comunicación en el modo maestro-esclavo. Donde el maestro es el primero en transmitir y va seleccionando a los diferentes esclavos que estén conectados a él. A esta red formada por 2 a 8 dispositivos (en estado activo, 255 si están en modo espera) se llama piconet. Cada dispositivo conectado a la piconet tiene una dirección lógica de 3 bits, por ello el máximo de 8 dispositivos en activo. Los que están a la espera se sincronizan pero no tienen dirección como los físicos. También se puede unir dos redes piconet a través de un dispositivo creando una red llamada "scatternet".



Ejemplo de una piconet

Se puede conectar dos dispositivos de bluetooth a través de los siguientes pasos:

1. **Modo pasivo**, en este modo el dispositivo está la espera, tomando información de la red.

2. **Fase de solicitud**, es la primera fase que se presenta para la comunicación, donde el dispositivo que hace la función de maestro envía una solicitud a todos los demás dispositivos que se encuentran en su radio de acción. Todos los dispositivos en funcionamiento que reciban la señal, mandarán su dirección de vuelta al dispositivo maestro.
3. **Paginación**: el dispositivo maestro elige una dirección y se sincroniza con el punto de acceso, dispositivo esclavo, mediante una técnica, que principalmente consiste en la sincronización de su reloj y frecuencia con el punto de acceso.
4. De esta manera se consigue un enlace entre el esclavo y el maestro, el cual, podrá pasar a la fase de descubrimiento de servicio a través del protocolo DSP (en español, **Protocolo de Descubrimiento de Servicio**).
5. **Canal de comunicación**: Cuando esta fase de descubrimiento del servicio finaliza, el dispositivo maestro está preparado para crear un canal de comunicación con el punto de acceso, mediante el protocolo *L2CAP*. Produce un puerto serie virtual.
6. Emparejamiento mediante el **PIN**, pin de seguridad que puede ir o no incluido en la conexión, sirve tanto para emparejar el maestro con el esclavo como un sistema de seguridad ya que sin este código no se podrá terminar de hacer la comunicación. Se pide desde el maestro que introduzca el código, en algunos casos, como en teléfonos móviles son los propios usuarios los que pueden introducir los códigos.

A partir de este momento queda abierto el canal de comunicación de una manera segura y se podrá tanto enviar como recibir datos.

Cómo funciona el bluetooth en el robot

Lo primero que se realiza, es la conexión con los pasos antes expuestos, donde lo primero será conectar el pendrive Bluetooth e introducir los drives de este último. Conectamos el robot K-Junior. Lo siguiente es pedir una solicitud a través del bluetooth conectado al ordenador. Para ello abrimos el panel de dispositivos e impresoras y

pinchamos en la de bluetooth, lo que nos abrirá una ventana que nos permitirá realizar el siguiente paso. Cuando pinchamos el de búsqueda de equipos pasamos automáticamente a la fase de solicitud donde se verá, si están bien instalados los drivers, del robot. Los demás pasos se realizarán automáticamente hasta llegar al último paso donde se nos pedirá el pin de conexión. Éste lo encontraremos en el manual dado en el Cd que nos ha proporcionado el fabricante (el código será 0000). Cuando se haya realizado, automáticamente se hará la conexión y nos dará automáticamente unos puertos COM virtuales. En esta comunicación, lo primero que hay que tener en cuenta, después de que ya esté listo para la comunicación, es ver el nombre del puerto COM virtual creado en la conexión entre el adaptador conectado al ordenador y el dispositivo W12 integrado al robot. Se puede ver a través de sus propiedades.

Para el envío de datos utilizamos el mismo programa que en el de la comunicación por USB, se trata del programa llamado Hiperterminal. Este programa se utiliza en muchos modos de comunicación, tanto del tipo de internet (TCP/IP), como en la comunicación con micro y otros aparatos. El programa, lo primero que hace es pedir cual va a ser el puerto y qué tipo de puerto con el que va a tener que comunicarse. Lo siguiente es la configuración del puerto si se puede, en el caso de bluetooth, ya está configurado por lo que pasa a una pantalla en blanco donde se pueden mandar los mensajes a través de código ASCII. Lo que simplifica en muchos casos la comunicación y el entendimiento entre un usuario y el dispositivo a controlar.

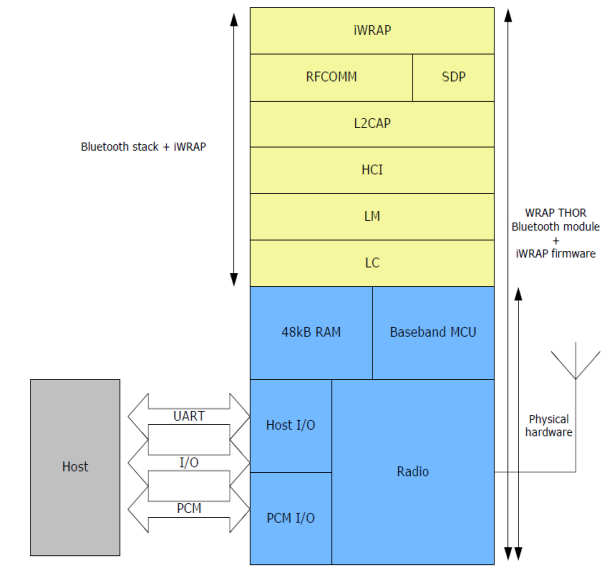
Del dispositivo del robot W12, el módulo de bluetooth, está conectado directamente a las mismas que las del FT232R. Por tanto, cuando el micro envía un dato, lo envía a través de las dos funciones y si lo reciben, lo recibirán por los dos módulos. Por lo que hay que tener cuidado de no conectar los dos a la vez, ya que esto puede llevar a un error. Pero también lo facilita ya que los dos transmiten en código ASCII y por tanto con una sola función de comunicación se hace cargo de los dos tipos de comunicaciones, simplificando un poco la programación y ahorrando muchas líneas de código.

Para ello, utilizaremos dos cosas. La primera es la función de printf la cual se utilizará para mostrar, tanto frases como datos que el micro puede mover y que el usuario desee ver. La otra es la interrupción INT_RDA, la cual salta cuando recibe un dato a través del teclado o mejor dicho a través de la comunicación con el código ASCII. Así con esta interrupción podremos manejar datos introducidos por el usuario, o acceder a menús de pantallas que de otra forma no se podrían acceder.

Estudio de Iwrap

El Iwrap se trata de un firmware, un bloque de instrucciones básicas de tipo máquina (más bajo incluso que el tipo ensamblador), son del tipo sólo lectura y se utiliza para controlar los circuitos electrónicos del aparato. En nuestro caso, las diferentes características y opciones de funcionamiento del bluetooth. Por tanto, se va ejecutar por completo en el procesador interno (RISC) del modulo W12 del robot. El Iwrap implementa la pila interna del bluetooth al completo y no requiere de un procesador central para hacerlo funcionar, ya que va incluido en el procesador del bluetooth.

El interfaz físico más comúnmente utilizado es el de mandar las ordenes por UART (transmisor receptor asíncrono universal), con comandos de ASCII. Por tanto para ello podremos utilizar el programa anteriormente mencionado, el Hiperterminal, con el que podremos mandar las instrucciones necesarias. También podremos ver a través de un comando el último comando por pantalla, para ello introducimos el comando + y nos pondrá por pantalla echo on.



En esta figura, se muestra el esquema del módulo Bluetooth, equipado con iWRAP conectando a un sistema anfitrión, como por ejemplo nuestro ordenador, mediante una interfaz UART pudiendo crear o utilizar

aplicaciones del bluetooth potentes de una forma fácil.

Si el sistema está conectado a un procesador, el software IWRAP puede ser utilizado para controlar el módulo mediante el uso de comandos basados en ASCII. Si, por el contrario, no tiene un procesador, o no hay necesidad de él. El IWRAP se puede configurar para que el módulo bluetooth sea totalmente transparente, conectándose o abriendo las conexiones automáticamente, pero en este modo, no se podrá contar con todos los comandos que tiene el IWRAP. También, se puede utilizar para otras funciones adicionales como detección de portadora, cambiar los baudios o velocidad de transmisión o conectar otros equipos de bluetooth a éstos.

En sí, el Iwrap hace que el bluetooth pueda funcionar de varios modos, estos son: modo de comandos, modo de datos, modo multiplexado y modo de audio. Se puede pasar automáticamente de uno a otro modo cuando tienen una o varias conexiones.

- **Modo de datos**: se trata del modo de funcionamiento normal donde los bluetooth mandan datos e información recibidos desde los aparatos y sistemas a los que están conectados. Ejemplo: Los datos recibidos desde un micro a un ordenador por bluetooth.

- **Modo de comandos**: es el modo predeterminado cuando se activa el iwrap. En este modo se puede introducir los comandos en ASCII para realizar las diferentes funciones que tiene el Iwarp como por ejemplo el comando Call (llamada), RESET…..Para entrar en este modo debemos de desconectar el adaptador y volverlo a conectar y tendremos un tiempo de 1 minuto para introducir una serie de comandos que será el de esc esc esc (aunque esto difiere dependiendo de la versión del Iwrap). Después, se mostrará por pantalla la palabra ready y ya se podrá escribir los comandos necesarios. Para salir, bastará con desconectarlo o volver a introducir la serie de comandos de escape.

- **Modo multiplexado**: es un modo especial donde se mete tanto el modo de datos y el modo de comando a la vez, sin necesidad de estar pasando de un modo a otro y sin tener que introducir la serie de comandos escapes. La única pega que puede tener es que al mezclar datos, comandos y eventos puede producir errores de comprensión. Para entrar

a este modo debemos introducirnos en el modo comando e introducir el comando llamado set control mux.

Pruebas realizadas

1.- Prueba realizada

En este caso utilizamos un programa base de trasmisión de datos ya programado en el robot como base y que se había probado en la comunicación a través del USB, que funcionaba. Después de seguir los pasos de instalación y conexión, activamos el programa hiperterminal con el COM perteneciente al robot. Comprobamos que el programa funciona con normalidad viendo como los datos van cambiando. Después, hacemos una desconexión del robot y lo volvemos a activar y rápidamente tecleamos el código de escape pero no conseguimos nada. Lo intentamos varias veces pero no se consigue nada.

2.- Prueba realizada

Después de haber visto un vídeo donde se configura el Iwrap, se ve cómo puede meterse en él de una manera sencilla, se decide probar con el mismo programa que utilizamos anteriormente, con un nuevo código de escape, esta vez es con la tecla del dinero (símbolo del dólar) otras tres veces. Realizamos la misma operación que siguen en el vídeo pero no se consigue nada.

3.- Prueba realizada

En esta se basa en el manual de Iwrap 3.0, el cual tiene un código de escape diferente también que será el de +++. Por lo que, como en las anteriores, utilizamos el mismo programa, con el mismo resultado. De este modo, vimos que no era cosa de la codificación sino del USB bluetooth conectado al ordenador que al no tener Iwrap (al ser del tipo anterior al W12) no reconocía ni mandaba los comandos.

Por tanto, al no haber conseguido entrar en la configuración del Iwrap y al ver que la configuración del bluetooth no nos permitiría la configuración que antes pusimos para el desarrollo de nuestra aplicación, la desechamos de nuestras posibles aplicaciones.

Ahora vamos a pasar a la última parte de esta etapa de elección de aplicación. Ésta es diferente a las otras partes de la etapa ya que se encargará del estudio de la cámara.

Cámara en línea del robot

Se trata de una extensión que viene en la caja del robot. Como su nombre indica, es una cámara que puede ser conectada al robot en la parte delantera, como sensor óptico. Esto nos da un método más seguro para tomar datos del entorno, además de que por la parte delantera ampliará la zona a analizar. Lo malo es que consume mucha cantidad de RAM, como ya veremos en la explicación más adelante.

Se empezó con un estudio de la cámara, para ello buscamos la información en internet, aunque no se encuentra demasiada información, sólo la que nos venía en la página del fabricante. Ésta, trata de un manual de uso de la cámara la que resumiremos en las siguientes páginas. También, te viene un archivo de descarga. Este archivo, contiene la biblioteca necesaria para hacerla funcionar y un ejemplo, que será explicado en la parte de programación y ejemplos.

1.- Manual de la cámara en línea

Se trata de un pequeño manual de explicación, donde podemos ver, no solo el funcionamiento, sino detalles de la cámara, como las funciones de programación, cómo se conecta la cámara y cuidados a tener en cuenta. Esto se expone en cuatro etapas que ahora vamos a explicar.

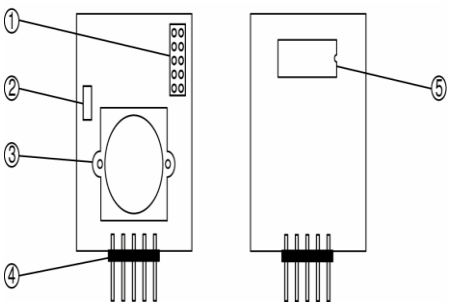
La primera se titula introducción (página 5), y empieza diciendo que con ese módulo introduce al robot en el campo de la visión artificial, y que, puede llegar a usar mucha

más información que la que solo nos es dada por los sensores de proximidad. Después, nos dice cómo utilizar el manual que se nos da y que si no nos queda clara la información que se nos ha dado, siempre podremos ponernos en contacto con el fabricante. El siguiente apartado son unas precauciones de seguridad, que realmente repiten lo que dicen en el manual principal del robot, excepto la parte de la instalación de la cámara, que como se indica deberá conectarse cuando esté apagado el robot. En la última parte de la introducción nos habla del reciclaje, y que hay partes del robot que cuando éste deje de funcionar no se deben tirar a la basura sino que deben llevarse al punto de reciclaje, como por ejemplo las baterías las cuales tienen polímeros de Litio, que pueden ser muy contaminante si se derraman o se queman sin un proceso de eliminación adecuado.

El siguiente apartado del manual, es el de conexión de la cámara (página 6) al robot. Ésta solo podrá conectarse en el conector delantero. Lo primero que nos dice es, que tendremos que conectar las extensiones con cuidado ya que las patillas son delicadas y pueden romperse.

Como ya se ha dicho, hay que conectarla en la parte delantera del robot, o mejor dicho en el conector delantero, ya que si se conectan en los laterales pueden dañar, tanto al robot como a la cámara. Por último, remarcar varias veces que la conexión y desconexión de la cámara del robot siempre debe hacerse con el robot desconectado.

La tercera parte se llama el K-Junior cámara en línea (página7), empieza con una pequeña información general que se dedica a explicar las partes que se ven a primera vista de la cámara:



1.- Conector de reprogramación: se trata de un conector puesto que hay para poder reprogramar el micro instalado que se ocupa de la cámara.

2.-Leds.

3.- Lente de la cámara M12x0.5.

4.- Pines de conexión con el robot.

5.- Micro-controlador.

La siguiente parte de la pregunta se llama dirección I^2C y nos muestra que, el robot controla la cámara a través de un bus de comunicación de tipo I^2C. Este bus está definido por 7 bits de datos y un último bit que nos marcará si está en modo lectura o escritura. También nos dice, que para leer datos tendríamos que mandarle el bit 0xC1, mientras que para escribir algún dato se tiene que mandar el bit 0xC0. Sigue con unas direcciones de registro, donde nos dice que, siempre hay que especificar si el registro es de lectura, de escritura o de ambos tipos. Como última parte a esta pregunta, nos muestra los siete registros que se utiliza en I^2C.

- **Registro de firmware**: nos permite leer el registro con la última versión instalada del firmware en la cámara. Su dirección es 0(0x00), y es solo de lectura.

- **Registro de lectura de pixeles**: como su nombre indica es el registro que se utiliza para la lectura de los valores de los pixeles. Cada uno contendrá un valor máximo de 8 bit, éste dependerá de la luz u oscuridad que le den. Su registro 16(0x10), solo lectura.

- **Registro de lectura del umbral de los pixeles**: nos muestra el valor del umbral de cada uno de los pixeles. Esto se muestra en un valor binario que será 0 (0x00) si no sobrepasa el umbral y 255(0xFF) cuando lo sobrepasa. Su registro es 17(0x11) y solo es de lectura.

- **Registro del valor umbral**: sirve no solo para mostrar el valor umbral de los pixeles sino también para modificarlo. Tiene un valor de 8bits sin signo. Su registro es 32(0x20) y sirve tanto para leer como para escribir el valor.

- **Registro de tiempo de exposición**: vale tanto para escribir como para leer el valor de tiempo que necesita los pixeles para tomar el valor. Aunque como los otros valores tiene 8 bits, nos recomiendan que solo cojamos valores desde 1 a 10. Su registro es 33(0x21) y como la anterior sirve tanto como para leer como para escribir.

- **Registro de leds**: se trata de un registro para conectar o desconectar el led de la cámara. Con 0 se desconecta y conectará con cualquier otro valor. Su registro de memoria es el 48(0x30), y solo es de escritura.

La siguiente y última pregunta se titula "*Uso*" (página 10), y como su nombre indica, nos explica los modos de uso de la cámara, uno por el puerto serie y la otra forma que sería a través de la programación en C (CCS). Después, pone un apartado de programación de CCS donde nos dice donde encontrarlo para descargar, también que hay una explicación sobre él en el manual general. Nos aconseja que debemos descargarnos la última versión de los programas. Para su uso es necesario introducir en la cabecera del programa la biblioteca hemlincam.h donde vendrán dadas las funciones necesarias para el uso de la cámara. La última parte de la pregunta es un glosario con las funciones, una pequeña explicación y un ejemplo de la forma de uso:

- Void HemLinCam Init(void)
  Su función principal es preparar la cámara, siempre que se vaya a utilizar la cámara deberá usarse.
  Ejemplo: HemLinCam();

- Char HemLinCam Read Version(void)
  Sirve para saber la última versión de la cámara o mejor dicho cual fue su última

actualización.
Ejemplo: printf("\n\r versión de la cámara %c", HemLinCam_Read_Version( ));

- Void HemLinCam Set Threshold(unsigned char)
  Se utiliza para poner el valor mínimo necesario que tendrá que tener el pixel para que dé un uno (255).
  Ejemplo: HemLinCam_Set_Threshold(127);

- Unsigned Char HemLinCam Read Threshold(void)
  Se utiliza para leer el valor anteriormente comentado.
  Ejemplo: valor_um= HemLinCam_Read_Threshold( );

- Void HemLinCam Set Exposure Time(unsigned char)
  Usado para dar el valor de tiempo que se necesitará para que el pixel tome un valor y así poder evitar posibles errores
  Ejemplo: HemLinCam_Set_Exposition(1);

- Unsigned Char HemLinCam Read Exposure Time(void)
  Lee el valor que anteriormente hemos descrito.
  Ejemplo:tiem=HemLinCam_Read_Exposition( );

- Void HemLinCam Read Pixels(void)
  Esta es la función que se encarga de leer todos los valores de los pixeles. Los cuales están divididos en tres cadena de valores llamados HemLinCam_Pixels_Zone pero terminados en diferente número dependiendo de la cadena que sea (1, 2 o 3).
  Ejemplo: HemLinCam_Read_Pixels( );
  if(HemLinCam_Pixels_Zone2[4]>50)

- Void HemLinCam Read Pixels Thresholded(void)
  Lee los pixeles, si superan el valor límite pondrá el pixel a 255 sino lo pondrá a

0.

Ejemplo: HemLinCam_Read_Pixels_Thresholded( );

if(HemLinCam_Pixels_Zone1[4] == 255)

- Void HemLinCam Set LED State(char)
- Función para cambiar el estado del led de la cámara.

  Ejemplo: HemLinCam_Set_Led_State(1);

Como última parte de la pregunta, nos aparece que la comunicación por puerto serie será la misma que la de K-Junior y que permitirá comunicar la cámara y el robot a través de cualquier entorno donde utilice el puerto serie como el Hiperterminal.
Por último pone unas especificaciones que serán:

Tamaño de la placa: 42 altox35anchox26grosor.
Peso: -
Fuente alimentación necesaria: 5 V.
Corriente [mA]: 10(durante la adquisición de datos) 0 (en espera).
Frecuencia máxima I^2C: 400kHz.
Número de pixeles: 102.
Nivel de grises: 8 bits (0-255).
Lente: tipo M12x0.5.

2.-PROGRAMACIÓN, EJEMPLO Y PRUEBAS.

Estudio del código proporcionado por el fabricante:

Lo primero que realizamos, es un estudio del código que hemos descargado para el robot. Este se divide en dos archivos, uno de tipo biblioteca y otro de tipo c, el cual, debería ser un ejemplo puesto por el fabricante para el robot.

Abrimos el archivo c_ejemplo, en él lo primero que encontramos, es la necesidad de introducir nuestra biblioteca KJunior.h para hacerlo funcionar. Lo siguiente, es que los

comentarios de estos programas se encuentran en francés, por lo que se nos dificulta el entendimiento del programa, así que leemos el programa para ver cómo funciona. Al leerlo, encontramos otro problema, y es que las funciones que se utilizan en este ejemplo no son del robot que hemos comprado, sino de otro de gama más alta que el nuestro. Así que nuestro siguiente paso es el de cambiar todas las funciones normales dadas por el fabricante y que el robot necesita para funcionar. Por último y después de comprobar que se ha cambiado todas las funciones necesarias, compilamos el programa y nos da un error bastante infrecuente, y es que el micro no tiene suficiente RAM como para poder crear todas las variables que se utilizan en el programa. Por lo que simplificamos el programa eliminando todas las variables no necesarias. Solo dejando las variables que proporciona el fabricante en sus dos bibliotecas. Volvemos a compilar y comprobamos que sigue dándonos el mismo problema de falta de RAM. Comprobamos que este error se da, por el exceso de variables que ha creado el fabricante, algunas de las cuales no se utilizan nunca. Nuestro siguiente paso es comentar poniendo delante de ellas dos contra barras y con esto ya no se crearán las variables, dejando espacio en la memoria RAM. Volvemos a compilar, pero nos encontramos el mismo error, por lo que debemos de comentar más variables. Pero ahora no solo tenemos que comentar las variables sino que lo haremos en partes del código del KJunior.c para quitar no solo las variables sino también la parte que nos produzca el error al utilizar las variables que nunca serán declaradas. Lo primero que quitamos será una parte de la interrupción int_RDA y su complementaria. Luego compilamos de nuevo. Esta vez conseguimos que no nos dé el error.

El siguiente paso fue la carga del programa en el robot. Para ello activamos el programa llamado Timybodloader que es el programa que utilizaremos para cargar el archivo.hex al robot. Este tipo de archivo, será el que entienda realmente el robot, y se encontrará en lenguaje maquina. Después de haber cargado el programa al robot y comprobando que la carga está completada, comprobamos que el programa funcione. Este simple programa se encargaba de ver si tenía algún obstáculo delante y si era así pararse activándose también los leds de enfrente. Si no había obstáculo seguiría hacia delante hasta que se encontrase con uno.

A continuación, encontramos otro problema cuando vemos que el robot sigue hacia delante, aunque se haya puesto un objeto delante de él. Viendo que el programa no funciona como debería. Después de leer el código y ver cómo actúa, se nos ocurre probarlo en la simulación, a través del programa Proteus con el cual no solo podemos simular el circuito sino que también podemos ver paso a paso el funcionamiento del código por lo que localizamos el fallo de una forma más sencilla. Comprobamos que el error se encuentra en la parte de la primera interrupción donde ésta no para de saltar antes de que pueda introducirse en el programa, de manera que se tendrá que modificar en un futuro.

Como vimos que no saltaba y se quedaba dentro siempre de la primera interrupción, decidimos modificar la interrupción de introducción de datos por teclado que antes habíamos eliminado, ya que su variable era demasiado extensa y no era compatible con las de la cámara. Sustituyendo ésta por una simple variable y haciendo así que solo se pueda introducir dato a dato, pero volviendo la interrupción compatible con la utilización de la cámara. Después de la última compilación, se prueba el programa en el robot poniendo objetos delante del robot y se consigue hacerlo funcionar de una manera correcta moviéndose cuando la cámara no detecta objeto y deteniéndose cuando sí. Este es un pequeño programa ejemplo que nos podría servir de base para cuando queramos hacerlo funcionar poniendo delante las plantillas.

Prueba de cámara 2 (lectura de datos)

En esta prueba lo que realizamos es ver los valores tomados por la cámara. Solo vamos a coger el vector central en este ejemplo. Para ello, primero configuramos la cámara luego configuramos el robot y lo siguiente que hacemos es ir cargando el número de cada pixel y su valor por pantalla, a través del Hiperterminal. Después de realizar el programa comprobamos su funcionamiento y vemos, que como ocurre en el anterior programa se queda colgado, por lo que necesitaremos modificarlo por presentar el mismo problema con la interrupción de tiempo. Por ahora, lo que hemos hecho es introducir una pequeña frase donde desconectamos todas las interrupciones con lo que

el programa funciona mostrando por pantalla sin parar y en una columna los valores de los pixeles.

Modificamos el programa, ya que no nos gusta que sólo se vea en una sola columna y que no se termine de ver los valores, por lo que no podemos verlos bien. A lo que pasamos a introducir tres variables, además de la variable incremental con lo que conseguimos poner los valores en tres columnas. El defecto, es que no se sabe cuando llega a los 34 pixeles por lo que debemos modificar e introducir en cada incremento la pregunta, si el incremento es mayor de 34 los pone en 0 sino lo incrementará. Por último, metemos un retardo de tiempo en la pregunta si es igual a 34 y de borrar la pantalla.

En la siguiente mejora como hemos dicho en el anterior programa, corregimos la función problemática, haciendo que nuestro robot funcione correctamente con la cámara y no se quede parado. Por lo que, ya no será necesario la desactivación de las interrupciones, y se podrá introducir en las otras aplicaciones.

Este programa, lo incluimos en un menú donde no solo saliese el valor de la columna central, sino que, dependiendo del dato elegido por teclado, pudiese salir cada una de las columnas. La tecla 1 para la primera columna, la tecla 2 para la segunda columna y la tecla 3 para la tercera columna.

Prueba de la cámara nº 3 (lectura de la distancia entre el robot y la cartulina)

En este caso, le sumamos al menú creado en el anterior programa, utilizado para elegir la columna que queremos ver (Prueba 2), le sumamos tres opciones más. Estas serán distancia a 5, 10 y 15 cm esto significaría que el robot se parase con una distancia de 5, 10, y 15 cm con respecto a la cartulina blanca. Para ello deberá pulsarse por teclado las teclas 7, 8, 9, mientras que conseguimos terminar de realizar el programa con el que podamos tomar datos de dos dígitos.

Como hemos dicho antes, para esta prueba realizamos antes unas medidas, las cuales se muestran en el anexo llamado tablas de medidas con cartulina blanca. En éstas, se muestran que por ejemplo, los valores de la columna central serán los más exactos ya que incluso con la cartulina a cinco centímetros, todos los valores de los pixeles están ya a 255 y no varían porque enfrente se encuentra una tarjeta blanca. Por tanto, cuando se aumenta la distancia tampoco variarán estos valores. Mientras, los valores de la primera columna sí que variarán ya que sólo los últimos valores son los que tienen su valor máximo. Y éstos, también dependerán de la distancia con la que se ponga la cartulina ya que alguno de éstos no alcanzaran el valor máximo para ciertas distancias de las cartulinas. Luego, comprobamos que en la tercera columna pasará lo mismo. En ésta, son los primeros valores los que toman valor máximo mientras que los demás variarán. Y como en la anterior, estos pixeles también dependerán de la distancia entre el robot y la cartulina.

Lo que planteamos con nuestro programa es que, como hay unos valores que a esas distancias llegan al máximo de su valor, podemos decir la cantidad de los pixeles que llegan a este valor máximo a traves de las tablas ya medidas. Con estos valores, los introducimos en variables enteras; ejemplo: Sabemos que a 5cm, la primera columna tiene 8 pixeles a valor máximo, la tercera siete. Por lo que en nuestro programa, en la opción de 5 cm, meteremos en las dos variables z=8 e y=7.Después, comprueba cuantos pixeles son los que tiene con valor máximo de cada columna y, si es así aumenta dos contadores creados por nosotros. Por último, se comprueba si los valores de los dos contadores coinciden respectivamente con los dados en la primera opción, si es así el robot se parará esperando a que el operador diga de volver a ponerlo en marcha o cambiar la opción. Si tiene mayor cantidad de pixeles, es porque está más lejos de la cartulina. Por lo que el robot deberá ir hacia delante hasta igualar los valores. Si tiene menos es porque está demasiado cerca, por lo que el robot debería ir hacia atrás hasta igualar los valores.

Por ahora hemos conseguido que el robot vaya hacia delante y se pare a cierta distancia pero la distancia no corresponde exactamente a la que debería. Por ello, tomaremos la tarjeta negra la cual hará cambiar los valores de la columna 2 y aproximando bastante a

la realidad la distancia tomada. Otra diferencia es que ahora se tomará solo el valor central de la cámara, ya que, es el valor más exacto pero esto también hará que se pare con cualquier cuerpo que cumpla esa condición sin depender de la distancia.

4.- Selección de aplicación.

Realizado el estudio de las posibles aplicaciones que puede tener nuestro proyecto, pasaremos a elegir la aplicación que más nos llame la atención para terminar.

Por cuestiones varias, tanto de un aumento de dificultad en cierto caso, como del bluetooth, imposibilidad de poder crear una aplicación deseada, nuestro proyecto podría tener dos principales aplicaciones.

1ª aplicación: Control del robot a través del mando a distancia, el cual, ya hemos descrito en la parte de comunicaciones por infrarrojos. Es una aplicación bastante vistosa y que podría ser válida.

2ª aplicación: Control del robot a través de la cámara. Donde aquí el control sería teniendo en cuenta los diferentes valores que puede recoger la cámara. La aplicación en sí sería una continuación de la última prueba.

Al final nos decantamos por la cámara ya que nos pareció más interesante el tema de la visión artificial.

CAPITULO V: APLICACIÓN

Elegida ya la aplicación, que será la de control del robot a través de la cámara, nosotros le daremos cierta distancia y éste al ver la forma que queremos a la distancia deseada se parará y si no seguirá en línea recta. A continuación, pasaremos a un estudio de los diferentes parámetros a tener en cuenta:

1.- Estudio de los valores que afectan a la cámara

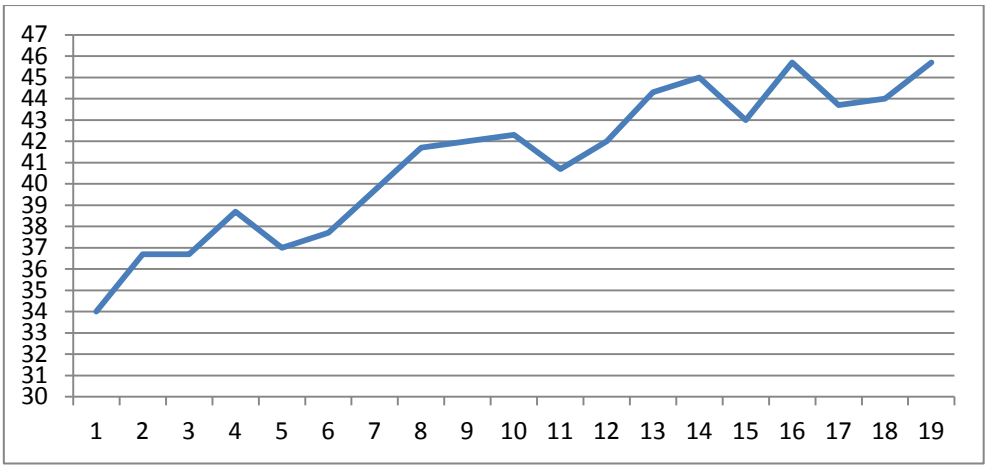
Lo que se quiere en este estudio, es ver si podemos sacar un algoritmo, con el cual podamos controlar el robot a través de la cámara, sin que tengamos que modificar los valores límites de la cámara. Esto quiere decir, que aunque las pruebas las hayamos hecho por ejemplo en una casa, no se necesite volver a programar valores con los que se controla el robot, o si lo que se quiere es que funcione en otro ambiente, que sea el propio robot el que lo vaya modificando a través de unas simples pruebas.

Para ello, utilizamos una opción creada anteriormente en nuestro programa de menú por teclado. Esta opción, realiza una matriz de datos de 2x3, dos columnas que representan los pixeles 12 y 13, las 3 filas que corresponde a las 3 columnas, en que está dividida la cámara. La primera fila corresponde a la primera columna, la segunda a la segunda y la tercera a la tercera.

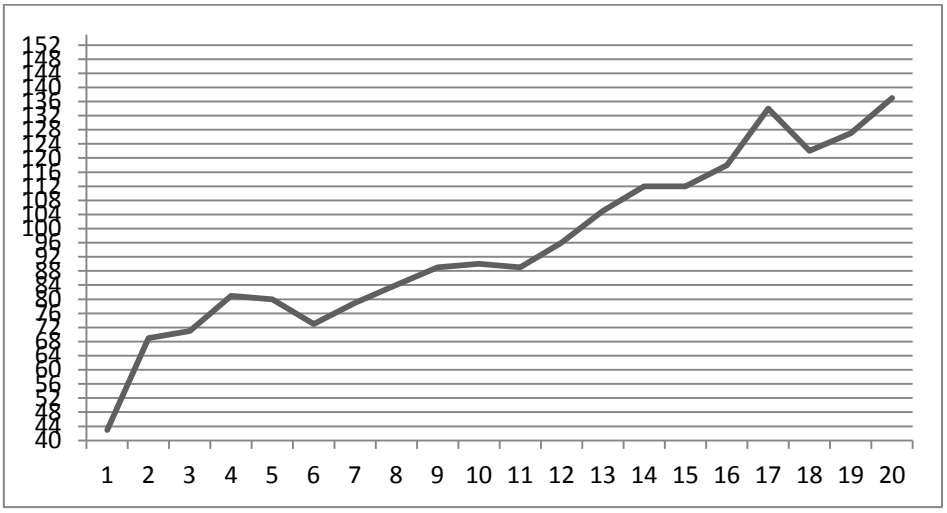
Con este programa ya funcionando y mostrando por pantalla los datos, lo siguiente que realizamos es un estudio de los valores que nos dan los pixeles dependiendo de la distancia a la que coloquemos la cartulina negra. Para ello creamos un libro de Excel, llamado tabla cámara, en el cual iremos introduciendo los diferentes datos. Lo primero que realizamos es ver si de verdad se cumple que el valor mínimo que tiene la cámara es 0 o tiene un valor de offset. Para ello introducimos el robot en una habitación a oscuras, tapando el led de funcionamiento, para que nos de un valor de 0, incluso, para disminuir el ruido lumínico al mínimo lo que hacemos es tapar el borde de la puerta con cinta

negra y bajar la persiana al máximo dejándola sin rendijas. Hacemos una primera prueba sin la cinta y nos da un valor de 5-6, ponemos también la cinta pero aun así no se consigue disminuir este valor por lo que se ve que tiene un valor de offset de 5 puntos.

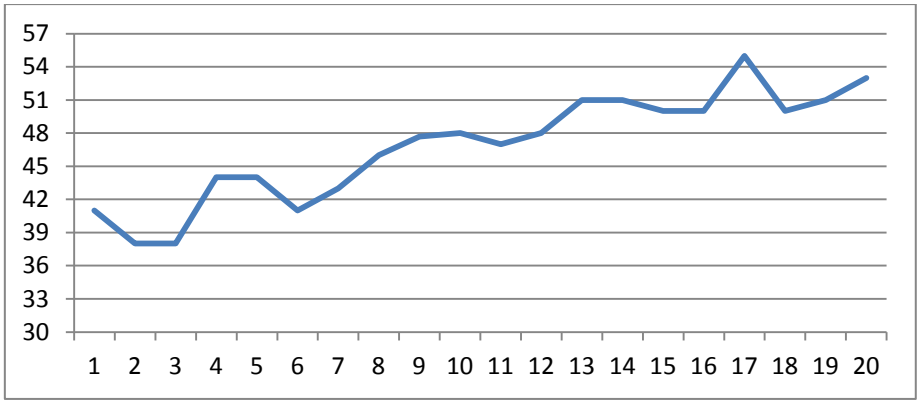
Ya demostrado que tendrá un valor mínimo que no se podrá anular o por lo menos es muy difícil de eliminar, pasaremos a la comparación de los datos a través de las siguientes gráficas de crecimiento de los valores de la cámara según la distancia en que se pongan la cartulina negra.



1º columna, datos tomados por la tarde.



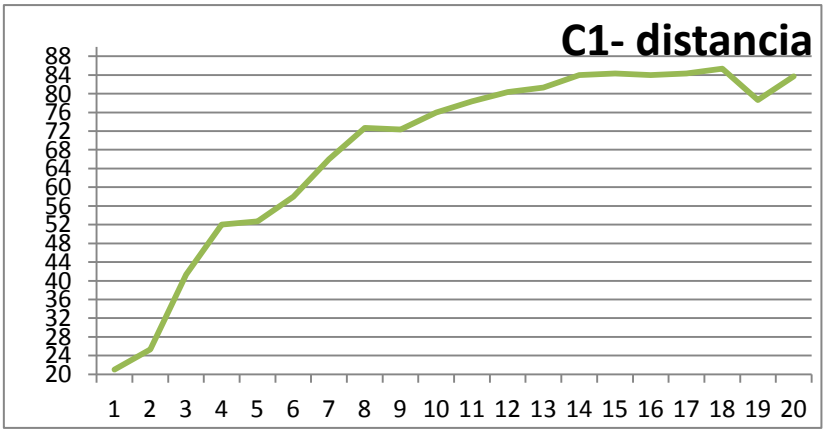
2º columna, datos tomados por la tarde.



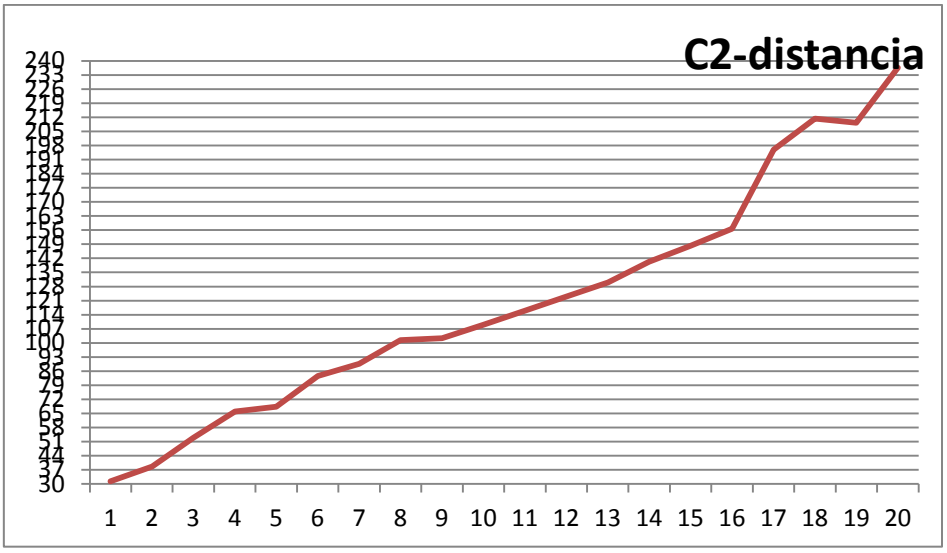
3º columna, datos tomados por la tarde.

Como se puede apreciar en estas gráficas los valores no son lineales y la toma del valor dependerá mucho de la distancia a la que se encuentre el objeto llegando el punto en que desde los 17 a los 20 cm se pueda empezar a desestimar.

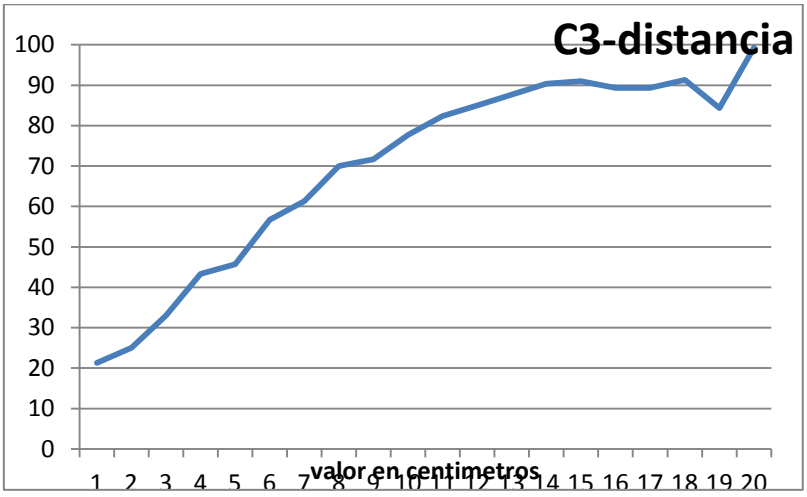
Aun así, el valor de la cámara no solo dependerá de la distancia, sino también de la luminosidad que es un factor ambiental, la cual, será muy difícil de controlar. Esto se ve mejor reflejado en la siguiente gráfica de crecimiento, hecha con los valores tomados por la mañana donde vemos que los valores son mucho mayores a la misma distancia en algunos casos.



1ª columna por la mañana



2ª columna por la mañana



3ª columna por la mañana

Comparando estas tres gráficas con las anteriores y viendo su crecimiento no lineal, ahora necesitaremos un tratamiento para esos datos, se ha elegido el tratamiento de binarización de la imagen ya que nos servirá para ver la silueta de la figura que es lo que realmente nos interesará.

2.- Binarización de la imagen

Es una forma de tratamiento de la imagen, es muy utilizada en visión artificial para analizar las siluetas de objetos. Para ello, utilizamos cámaras, que por su

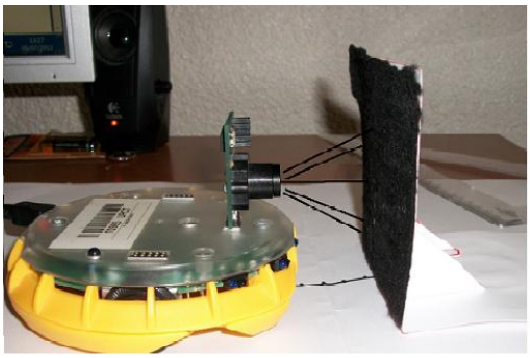
funcionamiento, no nos servirían para ver una imagen con la suficiente claridad o en colores pero que si permiten distinguir, una forma sencilla, que debería ser nuestro caso.

La binarización trata de tomar todos los valores de los pixeles de una cámara y los compara con cierto valor llamado **valor umbral**. Si los valores de los pixeles son mayores o iguales, será que están en blanco o que delante de él no se detecta nada. Si estos valores son menores, entonces será que se detecta un objeto o está en negro.

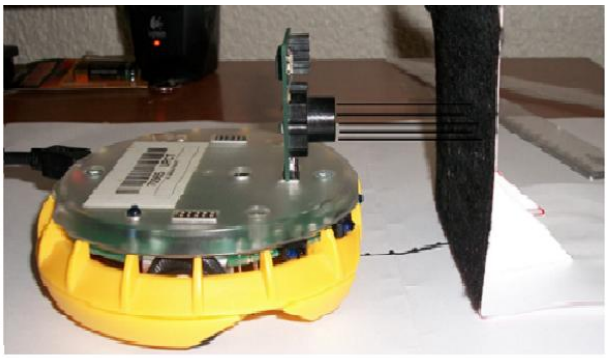
Dependiendo de si todos los valores de los pixeles tienen el mismo valor para el blanco y una caída igual para el negro, tendrá un solo valor umbral llamado **valor umbral universal**. Éste es el valor que el fabricante nos da y que utiliza para su función pensando que la cámara tendría un funcionamiento ideal y que todos los pixeles de la cámara trabajarían con el mismo valor (0-255).

Por otra parte, si estos valores límites no coinciden y dependen de las zonas en las que estén los pixeles, por ejemplo, en el caso de nuestra cámara con el objetivo, tendremos que utilizar unos valores llamados **valores umbrales locales**. Estos valores dependerán de los valores de un grupo de pixeles los cuales tendrán el mismo crecimiento. Un ejemplo será el de la columna 1 los valores del pixel 1 al 5 que tienen entre ellos una variación menor de diez, incluso menor de cinco, mientras, para el pixel 9, ya tiene una variación mayor de diez. Por lo que vemos, que con las pruebas realizadas con la cámara con el óptico, necesitaremos varios valores para la columna 1 y 3 que serán necesarios para la comprobación. Esto se debe a que el óptico o lente de la cámara, en su entrada, no es lo suficientemente grande como para dejar pasar la luz a todos los valores y nos da que muchos de ellos están en negro (46 de los 102 valores, un 45% de los valores). Pero, como hemos podido comprobar, nos será necesaria, ya que éste aumenta el rango de acción de la cámara. Además de hacer un direccionamiento de los haces de luz que recibe a través de la refracción de la lente, haciendo que incidan en los pixeles haces de luz con mayor ángulo. Esto se denota en que, con el óptico de la cámara a una distancia de 10 a 15 cm, hay valores de la columna central que estarán en blanco por el ángulo. Mientras que si la utilizamos sin óptico, la distancia en algunos

aspecto es mucho menor, incluso utilizando una cartulina negra no llegarán a más de 5 cm.



camara con optico



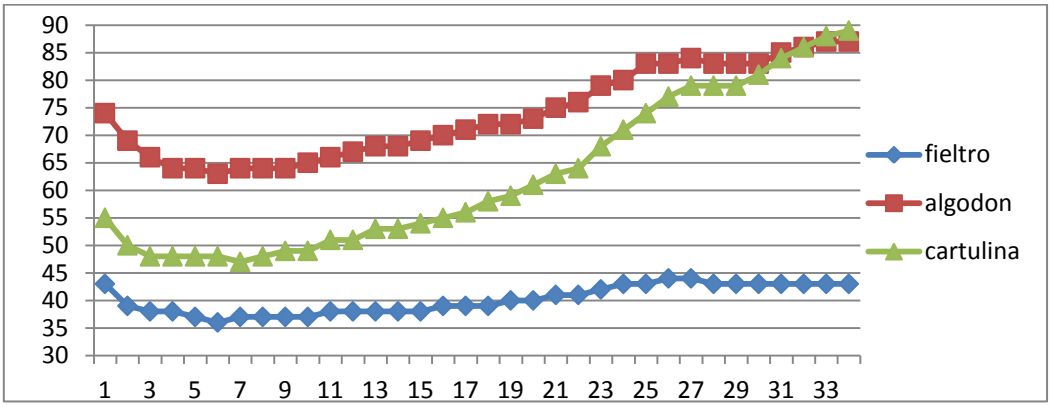
camara sin optico

El problema que nos viene con este tipo de binarización es, que para que la imagen sea lo mas exacta posible, se necesitarán bastantes valores umbrales locales, lo cual, se nos hará difícil, por no decir imposible, trabajar con ellos por la falta de memoria RAM.

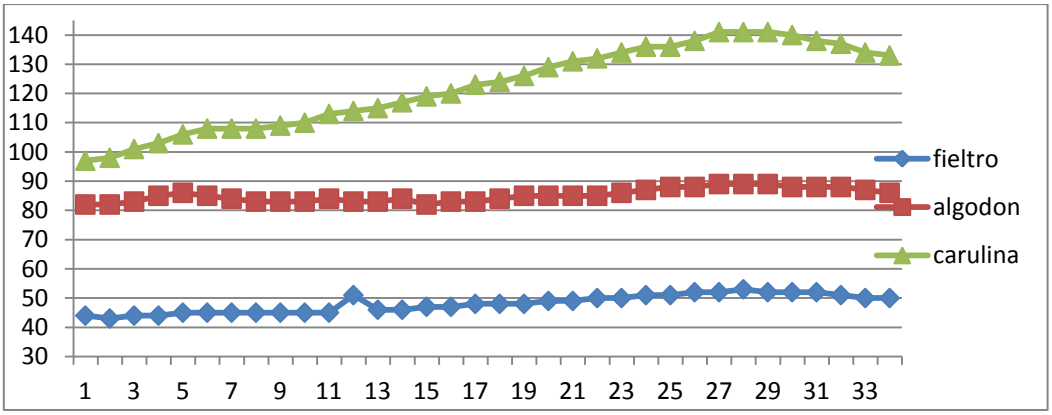
3.-Forma de la tarjeta final y material a utilizar

Aquí se pone de manifiesto que la forma que tenga no se podrá analizar de una forma correcta por la pérdida de valores causada por el óptico. Esto se demuestra en un programa que se hizo para ver la forma. Donde se suponía que si no tenía límite superior o inferior ponía que no se podía analizar. Por las pérdidas no se podía analizar el cuadrado entero pero si podía analizarse fácilmente si el límite de la parte de arriba de la columna 3 y el límite de la parte de abajo de la columna 1 eran iguales a los límites de superior e inferior de la columna central respectivamente. Esto se hacía así, porque estamos seguros de que funcionaría bien la parte de debajo de la columna 1 y la parte de arriba de la columna3. Por ello cuando coincidían estos valores (255), que son el último valor de 255 en el límite superior, antes de que se ponga negro, y el primer valor de 255 en el límite inferior después de que haya vuelto a blanco es porque la imagen sería cuadrada. Mientras que si coincide la inferior pero no la superior sería triangular ya que la parte de abajo coincidirá pero la de arriba al estar en negro antes que la columna tercera no coincidirá, por lo que será forma triangular. Si no coincide ninguno de los límites y tiene límites inferior y superior es porque tendrá una forma circular.

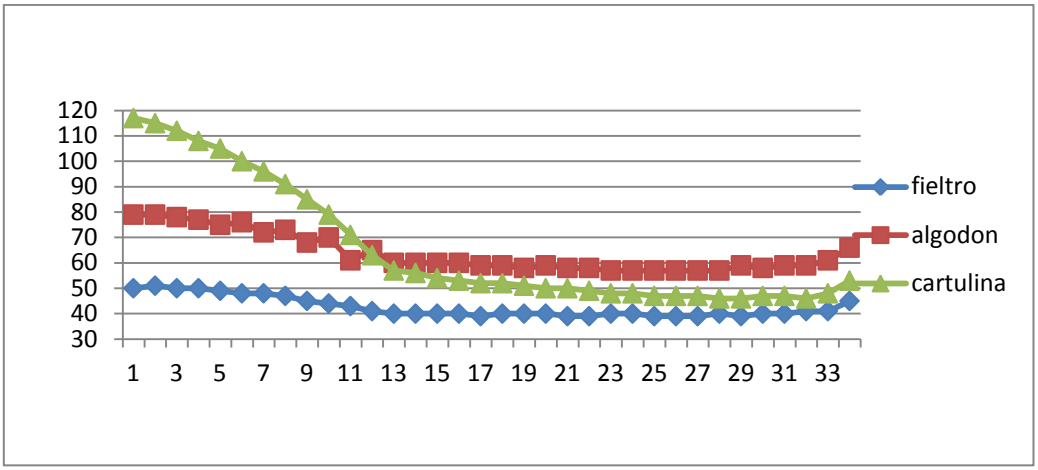
Otro aspecto que se ha comprobado, es la utilización de diferentes materiales en negro los cuales dependiendo de la rugosidad del material y lo reflectante que pudieran ser tenían una caída mayor o menor. Los materiales elegidos han sido, el fieltro y el algodón como telas naturales cuya principal característica de interés en nuestro estudio es la poca reflexión que van a producir, en comparación con una cartulina del mismo color que al ser un material artificial muy procesado tendrá una mayor reflexión. Para ello, hacemos otro estudio del cual pondremos las gráficas para que se observe claramente las diferentes caídas.



Columa1 a 2cm

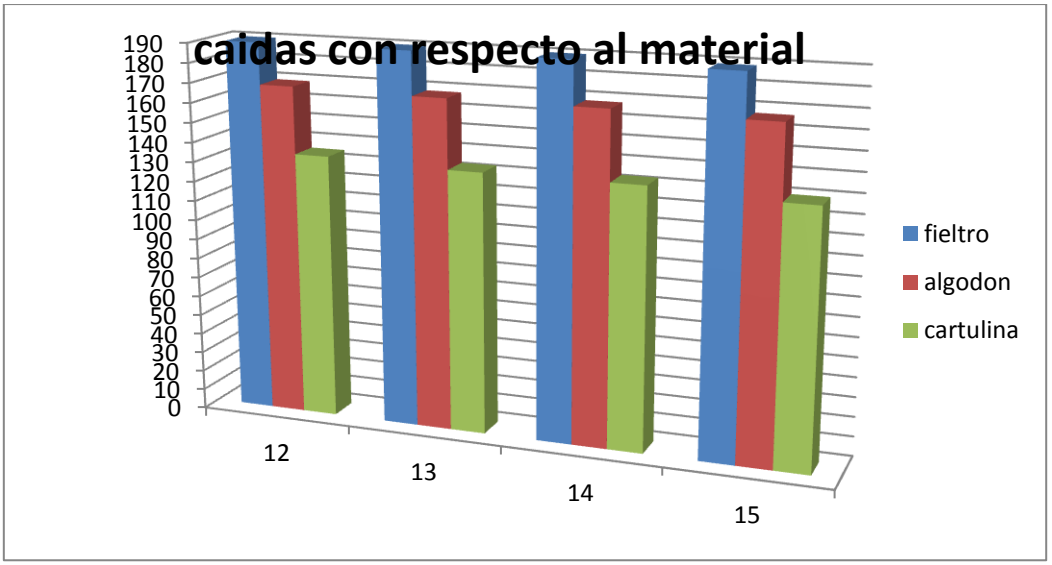


Columna 2 a 2cm



Columna 3 a 2cm

Observamos que el fieltro al ser menos reflectante, su tejido produce que sea el que menor luz devuelva, por lo que es el que más claramente se puede ver. Por ello será el que se elija. Como última prueba, mostraremos la gráfica de las caídas de los valores centrales a unos 5cm y comprobaremos que como ya vimos en las gráficas anteriores el que mayor caída tendrá será el de fieltro.



Por lo que vemos en las caídas, el valor del fieltro presenta una diferencia de de casi 30 puntos con respecto del valor del algodón, y de 60 puntos con respecto de la cartulina. Con esto ya queda demostrado que para nuestro caso lo mejor es utilizar fieltro el cual no solo se verá más claramente, sino que, en el caso de que se quite el óptico, podrá

aumentar la distancia máxima en comparación con la cartulina de unos 5-6 a unos 10-12 cm como máximo dándonos unos 5 cm más.

4.- Realización

Como última parte de este apartado realizaremos un programa que como ya habíamos comentado haga que el robot se mueva hacia delante o hacia atrás dependiendo de ciertos valores hasta cierta distancia que nosotros hayamos marcado. Para ello utilizamos el menú ya creado donde podremos utilizar el teclado para la elección de las diferentes opciones.

La primera opción se llama datos mínimos y máximos, ésta es una opción que muestra y almacena tanto los valores mínimos como los valores máximos de cada una de las 3 columnas. Esta opción se tendrá que elegir:

1. La primera vez que se enchufe en una habitación diferente o dirección diferente, ya que, cuando cambian estos factores cambiará la luminosidad.
2. Cuando pase cierto tiempo para una mayor exactitud de los valores, ya que como hemos visto en la gráfica, la luminosidad ambiental también se ve afectada por el factor tiempo que hace que disminuya o aumente, por lo que cada cierto tiempo se deberá volver a meter en esta opción para la toma de nuevos datos.
3. Para seleccionar la distancia a la que se quiere parar el robot, pues los valores mínimos y máximos que utilizará el robot serán los que se almacenen en esta parte del programa.

La segunda y la tercera opción del programa se trata del movimiento en sí donde el robot irá en principio moviéndose hacia delante y comparando los valores actuales de cada uno de los pixeles de la cámara, con los que hemos obtenido en la primera opción y, si están dentro de sus márgenes. Esto quiere decir, que si el valor del pixel es menor o igual que el máximo, y mayor o igual que el mínimo, es que está dentro de los límites y aumentarán un contador. Si el valor del pixel es menor que el del mínimo entrará en otro contador diferente y si es mayor en otro. Después, comprueba si todos los valores están entre los valores máximos y mínimos y si es así parará. Por el contrario, si todos

los valores no están entre estas variables, comprueba si hay alguno menor que el mínimo. Si es así es porque está muy cerca, por lo que irá marcha atrás hasta que estén todos los valores dentro de los márgenes. Por el contrario, si no hay ninguno menor, pero no todos los valores están entre los márgenes, es porque el robot estará muy alejado por lo que irá hacia delante hasta que los valores sean los correctos.

Al principio probamos el programa para que las 3 columnas estén entre los valores límite, pero esto causaba que el robot nunca se detuviera, ya que cuando no era la columna 1 la que no tenía el valor correcto era la 2 o la 3. Por tanto, cuando el robot se acercaba a la zona deseada se quedaba en un vaivén hacia delante y atrás. Esto se solucionó poniendo solo una columna, la central, como dominante y haciendo que se parase para esta columna.
La cuarta opción es diferente a las demás, ya que no tiene nada que ver con la opción de la cámara se trata de un pequeño programa de comunicación entre los robots a través del infrarrojo. Este programa mostrará por pantalla y activará los dos leds de enfrente si recibe dato. También podremos comprobar si el nuestro recibe dato si, como en el otro caso, el otro robot muestra por pantalla el dato mandado.

La quinta, sexta y séptima opción trata de mostrar los pixeles de la cámara por la pantalla respectivamente de cada columna.

Para poder realizar nuestro proyecto debemos utilizar los siguientes pasos a seguir para la parte de la aplicación referida al control del robot a través de la cámara:

1. Conectar el robot.
2. Si ya se tiene configurado la conexión por bluetooth conectar el Hiperterminal. Si no se ha realizado ninguna conexión a través de bluetooth configurarla como se pone en el capítulo III y en el IV en la respectiva parte de la comunicación por bluetooth (páginas 27 y 52).
3. Configurar si es necesario el Hiperterminal (en caso de Windows XP), los baudios, bit de parada, paridad y control de flujo.

4. Aparecerá ya la pantalla en blanco donde lo primero será colocar el robot a la distancia de la tarjeta que queramos que se pare.
5. Lo siguiente consiste en pulsar la tecla numérica 1, donde mostrará por pantalla si todos los valores están por debajo de 255 y cuáles son los valores límites (tanto inferior como superior) de las columnas.
6. Después pulsamos la tecla salto de línea y recolocamos el robot a una nueva distancia mayor o menor que la anteriormente elegida.
7. Dependiendo, si todos los valores de la segunda columna están por debajo de 255 se cogerá la opción 2 pulsando la tecla numérica 2. Si por el contrario tiene valores iguales a 255 se cogerá la opción de movimiento 3 pulsando la tecla numérica 3.

8. Después el robot marchara hacia delante o hacia atrás dependiendo de si se ha alejado o acercado a la tarjeta. Deteniéndose supuestamente en una posición sino exacta, muy cercana a la anterior.
9. Para que el robot vuelva a ponerse a buscar, se pulsará la barra espaciadora. Para realizar otra opción, pulsar la tecla salto de línea. Como ya hemos dicho anteriormente habrá que tomar datos cada vez que queramos marcar una nueva distancia al robot.

Para la parte de comunicación, al ser independiente al de la cámara, se expondrá a continuación.

1. Configuramos el primer robot, el que contiene nuestro menú de la aplicación como hemos hecho en los 3 primeros pasos anteriores.
2. Conectamos a través del USB al segundo robot con el ordenador. Si nunca lo hemos configurado seguimos los pasos que viene en el capítulo III la parte de comunicación por USB (página 28-29).
3. Abrimos otra ventana de hiperterminal, sin cerrar la del otro robot. Buscamos el puerto COM del nuevo robot conectado.
4. Configuramos como en la anterior los parámetros de hiperterminal, ahora de forma obligatoria para todos los sistemas operativos. Pulsamos enter.

5. Se abrirá una nueva ventana en blanco. Pulsaremos el reset y con la ayuda de un pequeño destornillador pondremos los micro-interruptores en valor 4, donde se mostrará por pantalla el dato recibido por infrarrojos
6. Pinchamos en la primera pantalla, la de menú de la aplicación, y pulsamos la tecla 4.
7. Ahora pulsaremos las teclas numéricas del uno al nueve. En la otra pantalla veremos aparecer el número en el dato recibido, ya que, el dato se ha recibido en el otro robot a través del infrarrojo del robot.

El único problema que presenta esta parte de la aplicación, es que coincidan los TIMER de los dos robots, ya que si no es así, puede que el dato que se reciba no sea el correcto.

CAPITULO VI: Conclusiones

En esta última parte de nuestro proyecto, expondremos las conclusiones sobre el mismo así como el estudio hecho del robot K-Junior y su extensión llamada hemlincam.
Las conclusiones que hemos llegado a lo largo de la realización de este proyecto han sido las siguientes:

El primer punto es que, quizás hubiese sido más interesante haber fabricado nuestro propio robot. Por una parte, sería mucho más educativo, ya que se aprendería más profundamente todas las bases tanto de la electrónica como las de programación de robótica. Además, nos ahorraríamos bastante dinero tanto en cuestión de fabricación como de programación.

El segundo punto es que, como la empresa creadora de este robot ha mostrado siempre en su página o en el manual, este robot solo sirve como una introducción a la robótica donde poder ver las cosas más básicas de la comunicación y la interacción del robot con el medio y otros robots. Es un buen ejemplo de la utilización de los infrarrojos como sensores. También nos muestra la codificación que hasta hace poco era utilizada por los mandos a distancia. Esta codificación era fácil y nos suponía una manera de control del robot a través de un mando a distancia de las televisiones de tubo que pudiesen encontrar en cualquier hogar.

En tercer lugar, este proyecto se deja abierto para una continuación basada en una red de robots comunicados entre ellos a través del bluetooh. En dicho proyecto se asentaron las bases pero no se consiguió la realización de la red del robot por problemas, éstos serán expuestos a continuación.

Por último, repasaremos los problemas que hemos encontrado en la realización de nuestro proyecto.

Empezaremos en la introducción en el firmware del W12, este presentaba el problema de que aunque se hubiese intentado todas las diferentes maneras de introducirse a través

del código esc, puestos en los diferentes manuales que la empresa fabricante nos muestra en internet, no se consiguió introducirse en este firmware. Esto, nos costó la pérdida de 2 a 3 meses de tiempo entre intentos y búsquedas de información.

Los siguientes problemas dados, serán todos referidos a la extensión de la cámara en línea. Ésta, empezó dándonos problemas desde el principio, con el estudio del ejemplo que nos deja el fabricante en la página de la cámara y del robot, el cual estaban todos los comentarios explicativos en francés, después tuvimos que sustituir todas las funciones pertenecientes al ejemplo, ya que eran de otro robot, por las del nuestro. Pero esto no fue el mayor problema que se encontró. El mayor problema que presenta este robot es la corta memoria RAM que posee de serie, ya que, hace imposible la creación de las variables usadas para guardar los datos recogidos de la cámara y la utilización de su función de toma de datos por teclado, todas ellas suministradas por el fabricante. Por lo que se podría hacer dos cosas, ampliar la memoria conectándole una memoria RAM externa, o la opción más sencilla y barata, sustituir una variable que nos ocupaba mucho espacio (la cadena variable de introducción de datos) en la memoria por otra mucho más corta. Haciendo que la toma de datos por teclado sea de una sola variable pero, permitiendo así la posible utilización de la cámara en línea. Esto no fue todo el problema obtenido de la cámara, sino que, se le debe unir el mal funcionamiento del objetivo de la misma que hace que muchos de los valores de las columnas laterales, tanto de la primera columna como de la tercera den por defecto negro, un valor menor de 255, cuando deberían de estar dando un valor de 255 según el fabricante. Esto nos produjo una serie de fallos que intentamos solucionar con la utilización de la binarización de la imagen y la utilización de valores umbrales locales. Pero al usar estos valores, aunque fuesen los mínimos necesarios, el micro no tenía la suficiente memoria como para después utilizarlo en el resto del programa, por lo que sólo nos ayudó en la parte del estudio de la cámara y de los perfiles. Esto demuestra que, en nuestro caso, el mayor problema que presenta el robot en sí mismo es la mala elección del micro-controlador y su escasa memoria.

El problema, se ve subsanado en la siguiente versión del robot, el K-Juniorv2, el cual presenta un micro-controlador del mismo fabricante (PIC) pero de una familia superior, un 18F en vez de un 16F. Éste funciona el doble de rápido que funciona el nuestro.

Además, tiene una memoria flash, RAM y ROM bastante mayor, siendo como mejor ejemplo la diferencia entre sus memorias RAM de unas 10 veces mayor el de la segunda generación en comparación con el de la primera generación (la RAM del primero es de 368 bytes, la RAM del segundo es de 3862 bytes). Esto, nos hubiese supuesto una posible utilización de la cámara sin la necesidad de cambiar el programa que nos da el fabricante, además de la posibilidad de utilizar todos los valores umbrales necesarios para el buen funcionamiento de la cámara.

En el manual de la segunda generación, también nos muestra que cambia la sirena y nos pone el valor que tiene que tomar para cada nota. Los motores tienen un movimiento más lineal, mostrando el valor que le queda a la batería, entre otras mejoras.

En conclusión, nuestro proyecto es un buen ejemplo de cómo puede afectar la norma precio-calidad o mejor dicho precio-características, tanto en la elección del micro como en los demás componentes, y como ésto puede suponer problemas a la hora de su utilización o a la hora de utilizar ciertas extensiones que el fabricante da de serie.

FUENTES BIBLIOGRÁFICAS

- K-TEAM, (2008). *K-junior Manual.10.*

- K-TEAM, (2008).*H2_REV04_schema* (esquemático del robot).

- K-TEAM, (2008).*K-Junior OS MANUAL Usage and Functions* (funciones del fabricante y explicación,).

- K-TEAM, (2008). *KJC Functions Glossary*.

- http://svcrobotics.googlecode.com/files/Generation%20JUNIOR.pdf
  (En esta página encontramos un pequeño resumen sobre el robot, sus extensiones y sus características principales a grandes rasgos).

- http://ftp.k-team.com/K-Junior/manuals/Intro_K-Junior_C_flow.pdf
  (Aquí encontraremos un manual donde nos explica cómo se puede programar el K-Junior y cómo utilizar su cargador de programas.).

- http://ftp.k-team.com/K-Junior/manuals/App1_-_Moving_straight_and_detecting_obstacle.pdf
  (En este enlace nos encontramos una pequeña explicación sobre los motores, los sensores y cómo hacer un pequeño programa ejemplo de detección de obstáculos.).

- http://ftp.k-team.com/K-Junior/extensions/lincam/KJ-LinCam_UserManual.pdf
  (En esta web nos encontramos el manual escrito por K-Team en junio del 2010 y nos explica tanto el funcionamiento como la instalación de la cámara HemLimCam).

- http://www.etitudela.com/celula/downloads/visionartificial.pdf
  (Enlace donde podemos encontrar información sobre visión artificial).
- http://pjmicrocontroladores.wordpress.com/2006/11/06/%C2%BFque-es-un-microcontrolador/
  (Enlace que nos explica de una forma bastante detallada qué es y para qué sirve un micro-controlador).

- http://cfievalladolid2.net/tecno/cyr_01/robotica/movil.htm

- http://www.slideshare.net/kfordonez/robots-moviles

- http://robotica.li2.uchile.cl/EL63G/capitulo4.pdf
  (Estos tres últimos enlaces nos muestran qué es un robot móvil, que tipos hay y como se pueden construir)