Projecte Final de Carrera

Enginyeria Tècnica de Telecomunicacions.

Especialitat en Sistemes Electrònics

Mapeo con robot móvil:
Construcción y seguimiento

Jónathan García González

Director: Joan Oliver i Malagelada

Departament de Microelectrònica i Sistemes Electrònics

Escola d'Enginyeria
Universitat Autònoma de Barcelona (UAB)

Febrer 2012

El sotasignat, Joan Oliver i Malagelada, Professor de l'Escola Tècnica Superior d'Enginyeria (ETSE) de la Universitat Autònoma de Barcelona (UAB),

CERTIFICA:

Que el projecte presentat en aquesta memòria de Projecte Fi de Carrera ha estat realitzat sota la seva direcció per l'alumne Jónatan García González.

I, perquè consti a tots els efectes, signa el present certificat.

Bellaterra, 13 de Febrer de 2012.

Signatura: Joan Oliver i Malagelada

AGRADECIMIENTOS

Me gustaría agradecer a mi familia, en especial a mis padres, todo su apoyo y confianza en mí, ya que sin su ayuda no podría haber llegado hasta donde hoy me encuentro.

A mi compañero Jorge Martín, darle mi gratitud por haberme ayudado en todo lo que ha podido, pues es el mejor compañero que uno puede tener.

Dar las gracias a nuestro tutor Joan Oliver por habernos recibido siempre con una sonrisa en la cara y hacer un verdadero placer trabajar a su lado.

Cuando a una persona le gusta lo que hace es capaz de trasmitir su energía a los que le rodean.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN.

Una labor en la que actualmente se dedican muchos esfuerzos de investigación es la de realizar mapas con robots móviles. Al ser el robot el que realiza el mapa, permite su elaboración en todo tipo de entornos, incluso en los que el ser humano no puede acceder, tales como cuevas, zonas radioactivas, e incluso realizar un mapa en Marte. Las técnicas que se utilizan para la realización de mapas, aparte de tener solamente esa finalidad, se hacen indispensables para dar al robot los conocimientos de su posición y entorno. Estos, con el tratamiento adecuado, aportarán al robot una mayor autonomía, aunque su fin sea otro.

La realización de mapas con robots móviles no es una tarea sencilla. Implica la construcción de un robot con todos los elementos sensoriales y actuadores necesarios para captar el entorno en el que se mueve. Por otro lado, tampoco son sencillos los algoritmos que determinan tanto su navegación como la detección del entorno. El tratamiento de los datos obtenidos tampoco es fácil, es por eso que se suelen utilizar algoritmos probabilísticos con este fin.

Dado que las dimensiones del proyecto son elevadas, se ha dividido el trabajo a realizar en dos tareas, una más técnica que abarca el montaje, calibración de

componentes y obtención de datos, y una segunda destinada a la caracterización de componentes, modelado del recorrido y entorno.

En esta memoria, se muestra más la parte de calibración y test y en una segunda memoria, *"Mapeo con robot móvil: Caracterización y modelado [1]"*, la parte de tratamiento de datos. Aún así, y como el trabajo ha sido realizado de un modo muy cooperativo, se ha hecho difícil realizar un buen fraccionamiento de las dos memorias. Debido a esto se encontraran capítulos similares en ambas.

1.1.DISTRIBUCIÓN DE LA MEMORIA

A partir de este capítulo 1, donde se encuentra una pequeña introducción y se explican los objetivos planteados, se irán exponiendo los capítulos siguientes:

- **Capítulo 2**

  Se muestra la planificación realizada por ambos miembros del proyecto, y los recursos utilizados.

- **Capítulo 3**

  En este capítulo se nombran y explican los componentes más usados para la realización de un robot autónomo y de un robot autónomo de mapeo y para finalizar, el montaje de este último.

- **Capítulo 4**

  En este capítulo se explican todos los algoritmos realizados para el control de los componentes del robot, y el algoritmo encargado de realizar el mapeo.

- **Capítulo 5**

  Este capítulo incorpora las pruebas realizadas para calibrar los componentes y corregir los posibles errores.

- **Capítulo 6**

  En este capítulo se muestra los resultados obtenidos al realizar el mapeo.

- **Capitulo 7**

  Conclusiones.

1.2.OBJETIVOS

El objetivo principal de este proyecto consiste en la realización de un robot autónomo, dotado de los componentes necesarios con los que poder realizar el mapeo de una habitación o recinto, sin necesidad de ser guiado ni corregido por acciones externas.

Para conseguir el objetivo principal se tendrán que usar distintos tipos de sensores/actuadores, de los cuales se tendrá que realizar un estudio para determinar su funcionamiento, consiguiendo observar in situ los conocimientos adquiridos en estos años de universidad. Gracias a los sensores se podrá realizar un mapeo con los resultados obtenidos y de esa forma poder representarlos de forma gráfica. Por lo tanto, para realizar el objetivo principal se tendrán que llevar a cabo objetivos secundarios.

Uno de estos objetivos secundarios, será el de adquirir conocimientos de los componentes disponibles en el mercado, y mediante su análisis poder llegar a determinar la mejor elección. Además se tendrá que desarrollar e implementar los drivers de control para dichos componentes, ya que por si solos carecen de utilidad.

Otro objetivo secundario consistirá en el aprendizaje de los distintos programas que se necesitarán para la programación de los componentes o de la comunicación, así como para la realización de gráficos del espacio mapeado. Se tendrá que elaborar un software de comunicación con el robot, que permitirá ejecutar acciones y extraer los datos que se usaran para realizar el objetivo principal, es decir, el de mapear una superficie.

Por tanto, podríamos decir que aparecen objetivos secundarios en base a las tareas que se deben realizar, estos son principalmente:

- **Adquisición de conocimientos y capacidad de análisis**
  - De todos y cada uno de los componentes utilizados.
  - De los entornos de programación.

- **Creación de programas**
    - Programación de drivers de control.
    - Programa de captura de medidas.
    - Programa de caracterización de componentes.
    - Programa de comunicación. Robot ↔ Pc
    - Programa de navegación.
    - Programa de mapeo.
- **Calibración de los componentes**
- **Elaboración de una memoria**
    - Capacidad de transmitir los conocimientos adquiridos.

Para finalizar, hay que destacar un último objetivo secundario, y no por ello el menos importante, el de seguir aprendiendo y adquiriendo los conocimientos que permitirán abarcar el mundo laboral con una mínima solidez y confianza. Debido a lo interesante que parece el tema, se puede asegurar que aprender y disfrutar irán de la mano.

Capítulo 2. PLANIFICACIÓN Y RECURSOS UTILIZADOS.

En este capítulo se muestra la planificación seguida para poder realizar el proyecto, además se hace una breve explicación de las tareas que forman dicha planificación.
También se hace referencia a los recursos que se han utilizado, a nivel hardware y software.

2.1.PLANIFICACIÓN DEL PROYECTO

En este apartado se explica la planificación del proyecto y las tareas realizadas para poderlo llevar a cabo. Hay que destacar que el proyecto es suficientemente grande como para que hayan trabajado dos personas, y por lo tanto las tareas están distribuidas pensando en dos proyectistas. A continuación se hace una breve explicación de en que consiste cada una de las tareas realizadas, a la vez que se identifican los recursos utilizados.

- **T.1. Especificaciones del proyecto y reuniones con el tutor**

Como su nombre indica son todas las reuniones que se han llevado a cabo con el tutor, bien para corregir errores en el trabajo realizado, como para solucionar los diversos problemas que han ido surgiendo e intercambiar impresiones.

  - Miembros involucrados:
    - Jorge Martín
    - Jonathan García

- **T.2. Construir el robot**

Esta tarea consiste en el montaje de todos los componentes que forman el robot, a la vez que se realizaban las conexiones necesarias de cada uno de ellos con la placa de control. Una vez realizada esta tarea se puede avanzar en el proyecto, por ese motivo al ser la primera, es la más importante.

  - Miembros involucrados:
    - Jorge Martín
    - Jonathan García

- **T.3. Crear los programas de control del robot**

Consiste en realizar un estudio y lograr gobernar todos los posibles movimientos del robot. Para ello se profundizó en la teoría del pulso PWM y en las posibilidades que el micro controlador ofrece para poder implementarlo. Una vez conseguido se procedió a la realización de los algoritmos de movimiento.

  - Miembros involucrados:
    - Jorge Martín
    - Jonathan García

- **T.4. Establecer comunicación con el robot**

Para poder controlar el robot y así conseguir calibrar y estudiar los componentes, hace falta establecer una comunicación con el PC. En una primera aproximación se hizo uso de *Hyperterminal* con el que conseguir una comunicación estable. Posteriormente se realizó una aplicación, mediante el entorno de programación *Python*, con el que lograr, aparte de la comunicación, realizar un tratamiento óptimo de los datos.

- Miembros involucrados:
  - Jorge Martín
  - Jonathan García

- **T.5. Estudio y programación de los *encoders***

Consiste en estudiar el funcionamiento y obtener una correcta calibración de los *encoders*.

- Miembros involucrados:
  - Jorge Martín
  - Jonathan García

- **T.6. Estudio, programación y calibración de la brújula**

Consiste en el estudio de las posibles formas de conexionado que ofrece la brújula (PWM, I2C), a la vez que se investiga su funcionamiento y la manera de realizar una correcta calibración para disminuir los posibles errores.

- Miembros involucrados:
  - Jorge Martín
  - Jonathan García

- **T.7. Estudio, programación y calibración de los sensores**

En esta tarea se realiza un exhaustivo estudio de los sensores involucrados en la detección de obstáculos, caracterizándolos y calibrándolos correctamente.

Para ello se precisa de la realización de algoritmos con los que poder obtener las conclusiones oportunas.

- Miembros involucrados:
  - Jorge Martín
  - Jonathan García

- **T.8. Crear los algoritmos necesarios para el mapeo**

Esta tarea está distribuida en dos partes. La primera parte consiste en la realización del algoritmo de control del robot con la que realizar el seguimiento de la pared, y la segunda parte consiste en la elaboración del algoritmo con el que poder graficar a partir de los datos obtenidos de los sensores y del robot.

- Miembros involucrados:
  - Jorge Martín (algoritmo para graficar)
  - Jonathan García (algoritmo de control)

- **T.9. Realización de las pruebas y corrección de los errores**

  Consiste en la realización de todas las pruebas con las que determinar los errores. Permitiendo variar los componentes y/o los códigos por tal de solucionarlos. Consiste en una tarea conjunta por ambos miembros del proyecto.

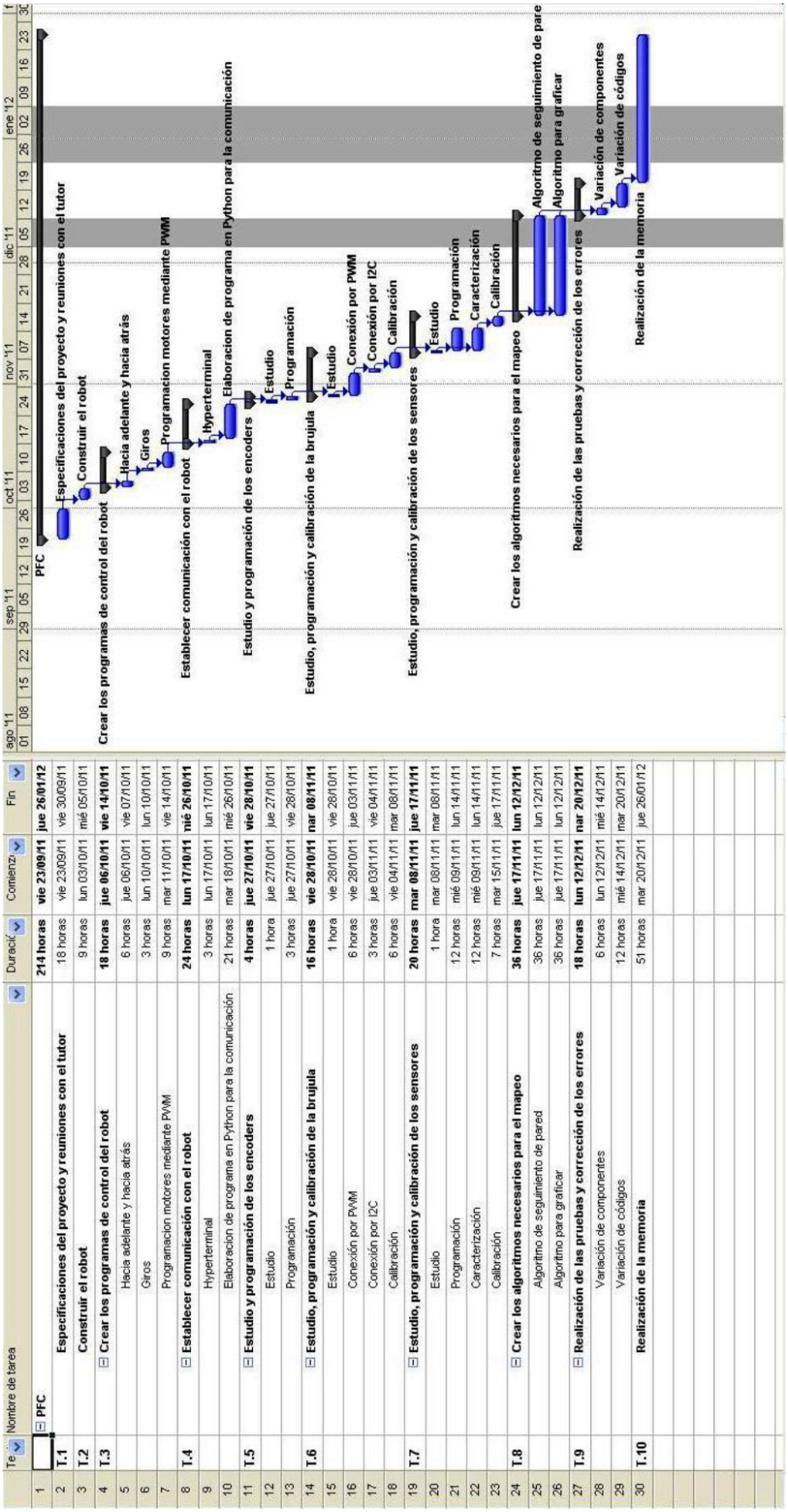
  - Miembros involucrados:
    - Jorge Martín
    - Jonathan García

- **T.10. Realización de la memoria**

  Tal y como su nombre indica, consiste en la realización de la memoria del proyecto, y por ese motivo se reparte entre los miembros del proyecto, realizando así cada componente del proyecto su propia memoria.

  - Miembros involucrados:
    - Jorge Martín
    - Jonathan García

Todas las tareas anteriormente nombradas y explicadas se pueden apreciar en la siguiente ilustración, donde se observa la planificación del proyecto, realizada con la herramienta que proporciona *MS office Project.* Cabe destacar que aunque se aprecia una elaboración muy continua, el resultado final no ha sido exactamente igual. Principalmente, se ha tenido que volver sobre los pasos realizados, al encontrar en los estudios, componentes que no funcionaban correctamente y que se han tenido que sustituir por otros. De igual manera destaca la implementación y uso de la brújula, que una vez implementado su uso mediante PWM, se tuvo que volver a realizar mediante I2C, ya que se conseguían los datos de una manera más rápida y fluida.



	Te	Nombre de tarea	Duració	Comienzo	Fin
1		PFC	214 horas	vie 23/09/11	jue 26/01/12
2	T.1	Especificaciones del proyecto y reuniones con el tutor	18 horas	vie 23/09/11	vie 30/09/11
3	T.2	Construir el robot	9 horas	lun 03/10/11	mié 05/10/11
4	T.3	Crear los programas de control del robot	18 horas	jue 06/10/11	vie 14/10/11
5		Hacia adelante y hacia atrás	6 horas	jue 06/10/11	vie 07/10/11
6		Giros	3 horas	lun 10/10/11	lun 10/10/11
7		Programacion motores mediante PWM	9 horas	mar 11/10/11	vie 14/10/11
8	T.4	Establecer comunicación con el robot	24 horas	lun 17/10/11	mié 26/10/11
9		Hyperterminal	3 horas	lun 17/10/11	lun 17/10/11
10		Elaboracion de programa en Python para la comunicación	21 horas	mar 18/10/11	mié 26/10/11
11	T.5	Estudio y programación de los encoders	4 horas	jue 27/10/11	vie 28/10/11
12		Estudio	1 hora	jue 27/10/11	jue 27/10/11
13		Programación	3 horas	jue 27/10/11	vie 28/10/11
14	T.6	Estudio, programación y calibración de la brujula	16 horas	vie 28/10/11	mar 08/11/11
15		Estudio	1 hora	vie 28/10/11	vie 28/10/11
16		Conexión por PWM	6 horas	vie 28/10/11	jue 03/11/11
17		Conexión por I2C	3 horas	jue 03/11/11	vie 04/11/11
18		Calibración	6 horas	vie 04/11/11	mar 08/11/11
19	T.7	Estudio, programación y calibración de los sensores	20 horas	mar 08/11/11	jue 17/11/11
20		Estudio	1 hora	mar 08/11/11	mar 08/11/11
21		Programación	12 horas	mié 09/11/11	lun 14/11/11
22		Caracterización	12 horas	mié 09/11/11	lun 14/11/11
23		Calibración	7 horas	mar 15/11/11	jue 17/11/11
24	T.8	Crear los algoritmos necesarios para el mapeo	36 horas	jue 17/11/11	lun 12/12/11
25		Algoritmo de seguimiento de pared	36 horas	jue 17/11/11	lun 12/12/11
26		Algoritmo para graficar	36 horas	jue 17/11/11	lun 12/12/11
27	T.9	Realización de las pruebas y corrección de los errores	18 horas	lun 12/12/11	mar 20/12/11
28		Variación de componentes	6 horas	lun 12/12/11	mié 14/12/11
29		Variación de códigos	12 horas	mié 14/12/11	mar 20/12/11
30	T.10	Realización de la memoria	51 horas	mar 20/12/11	jue 26/01/12

Ilustración 1: Planificación y diagrama de Gantt

2.2.RECURSOS UTILIZADOS

Para la realización del proyecto se han empleado distintos recursos, los cuales se pueden diferenciar en dos grandes bloques:

- **Hardware**

  Todo lo referente al hardware se explica a lo largo de esta memoria, donde se encontrará una explicación de todos los componentes usados y de como se programan para su correcto funcionamiento.

- **Software**

  A continuación se hace una explicación breve de todos los programas usados para poder llevar a cabo la resolución de este proyecto y que función han realizado.

  - **AVR Studio**

    Este programa ha sido el utilizado para programar el microprocesador ATMega 128, con el cual se han realizado todas las pruebas y el código final responsable de mapear.

    Dicho programa consiste en una interfaz muy sencilla de utilizar, ya que con unos pocos pasos se puede realizar la programación y configuración del microprocesador. Esta programación se realiza utilizando el lenguaje C++, aunque también se puede programar en lenguaje ensamblador, pero en nuestro caso se decidió usar C++ por su fácil entendimiento.

    Además dispone de un interfaz de simulación del microprocesador, donde se puede simular el comportamiento de este y saber como reacciona ante cualquier situación.

  - **Python**

    Inicialmente el robot era controlado por el ordenador a través del *hyperterminal*, debido a que no requeríamos ningún proceso externo de los datos enviados por el robot.

Posteriormente se decidió usar el programa *python*, el cual usa su propio lenguaje, que consiste en un nivel mas alto que *C++,* lo que hace que aun sea más sencillo de entender y de programar.

Gracias a ello se pudo realizar operaciones posteriores, tale como la conversión a grados de los valores digitales de la brújula o incluso realizar la formula de 4º orden para convertir el valor digital del sensor de infrarrojos en centímetros. Todo ello gracias a que se utiliza la velocidad de procesado del ordenador, siendo así mucho más rápido. En la siguiente ilustración se muestra la ventana generada por *python* al ejecutar el programa realizado para el control del robot.

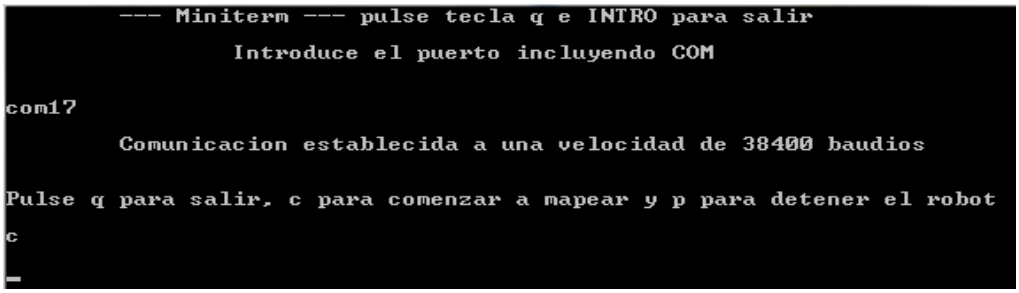


Ilustración 2: comunicación python-robot

En la ilustración se puede apreciar como establece una comunicación serie a través del puerto COM 17 con una velocidad de 38400 baudios, al igual que una pequeña explicación de las teclas asociadas a las instrucciones que queremos que realice el robot.

- **Matlab**

  Este programa es una herramienta muy potente para realizar gráficos a través de los datos proporcionados, con una previa programación, y por ese motivo se ha usado para realizar el gráfico correspondiente al mapeo.

  Todo lo relacionado al código usado para realizar el gráfico del mapeo con este programa se puede encontrar en la memoria *Mapeo con robot móvil: Caracterización y modelado*.

Capítulo 3. COMPONENTES Y ARQUITECTURA DE UN ROBOT PARA EL MAPEO.

3.1.UN ROBOT AUTÓNOMO.

Para que un robot autónomo pueda funcionar tiene que estar formado por distintos componentes, los cuales se pueden diferenciar en 5 apartados.

- Sensores
- Locomoción
- Comunicación
- Inteligencia
- Alimentación

A continuación se hace una explicación genérica de cada apartado y los distintos dispositivos que lo componen.
Es sólo una enumeración de los principales dispositivos que pueden integrarse en un robot autónomo sencillo, sin pretender ser exhaustivo.

- **Sensores**

  Los sensores son necesarios para que el robot pueda ser autónomo ya que son capaces, por sus propias características, de transformar la magnitud física medida en una señal eléctrica para que finalmente se pueda utilizar en el control del robot.

  Existe una amplia variedad de dispositivos dependiendo de la magnitud física que deseamos medir.

  A continuación se explican los más conocidos y tal vez los más utilizados para realizar un robot autónomo.

  - **Sensores de distancia**

    La finalidad de estos tipos de sensores es detectar distancias, ya sean para medirlas o para evitar que el robot impacte contra algún objeto.

    Dependiendo de su funcionalidad pueden ser:

    - **Capacitivos**

      Sensores capaces de detectar objetos midiendo el cambio en la capacitancia, la cual depende de la constante dieléctrica del material a detectar, su masa, tamaño, y distancia hasta la superficie sensible del detector.

    - **Fotoeléctricos (infrarrojos)**

      Sensores capaces de detectar objetos midiendo la intensidad de la luz reflejada en ellos. Estos sensores requieren de un componente emisor que genera la luz, y un componente receptor que detecta la luz reflejada por el objeto.

    - **Ultrasónico**

      Sensores capaces de detectar objetos a distancias de hasta 8m. Este sensor emite impulsos ultrasónicos. Estos reflejan en un objeto, el sensor recibe el eco producido y lo convierte en

señales eléctricas, las cuales son elaboradas en el aparato de valoración. Estos sensores trabajan según el tiempo de transcurso del eco, es decir, se valora la distancia temporal entre el impulso de emisión y el impulso del eco.

- **Sensores magnéticos**

  La finalidad de estos tipos de sensores es medir una variación en el campo magnético, de esta forma se puede saber la dirección del robot respecto al campo magnético terrestre.

  Dependiendo de su funcionalidad puede ser:

  - **Acelerómetros de efecto Hall**

    Este tipo de sensor se utiliza para medir la inclinación y las aceleraciones que experimenta usando una masa sísmica donde se coloca un imán y un sensor de efecto Hall que detecta los cambios en el campo magnético.

  - **Brújula electrónica**

    Como su nombre indica este tipo de sensor se utiliza para medir el campo magnético terrestre utilizando un chip capaz de detectar dicha variación.

- **Sensores de velocidad**

  La finalidad de estos tipos de sensores, como su nombre indica, es medir la velocidad a la que se mueve un objeto, el cual puede ser una rueda, una cadena, o cualquier mecanismo que requiera de dicha medición.

  En este caso el sensor de velocidad mas utilizado para robótica es el tacómetro de pulso (más conocido como *encóder*), el cual utiliza un haz de luz que cada vez que es interrumpido genera un pulso digital, de esta forma aplicando la ecuación del perímetro y multiplicándola por la

cantidad de pulsos medidos se obtiene la distancia recorrida, y a través de la frecuencia de este pulso se obtiene el tiempo, para que así finalmente se obtenga la velocidad aplicando su ecuación.

- **Sensores de luz**

  Los sensores de luz se usan para detectar el nivel de luz y producir una señal de salida representativa respecto a la cantidad de luz detectada. Un sensor de luz incluye un transductor fotoeléctrico para convertir la luz a una señal eléctrica y puede incluir electrónica para condicionamiento de la señal, compensación y formateo de la señal de salida.

  El sensor de luz más común empleado en robótica es el LDR (*Resistor dependiente de la luz*), el cual consiste en un resistor que cambia su resistencia cuando cambia la intensidad de la luz, haciendo que su resistencia sea mínima cuando la luz incidente es máxima, y viceversa.

- **Locomoción**

  El uso de componentes de locomoción son necesarios para poder realizar un robot autónomo, ya que para que se pueda desplazar es necesario como mínimo dos ruedas y sus respectivos motores. Dichas ruedas se pueden colocar de diversas formas, según la arquitectura deseada, el robot, realizará un tipo de movimiento en concreto. Esta arquitectura se encuentra explicada detalladamente en la memoria *Mapeo con robot móvil: Caracterización y modelado [1]*, concretamente en el capítulo 4.

  A continuación se nombran los componentes de locomoción mas usados en la construcción de robots autónomos terrestres.

  - **Componentes de tracción**

    Para poderse desplazar se ha de dotar al robot de un componente dedicado a tal fin. En este caso se pueden encontrar dos formas de realizar dicho propósito.

- **Ruedas**

  Las ruedas son las que proporcionan la tracción necesaria al robot, por ese motivo es uno de los componentes más importante del robot. Estas pueden ser simples o preparadas para que los *encoders* a través de ellas puedan medir distancia recorrida.

- **Ruedas *oruga***

  Las orugas son una alternativa a las ruedas, gracias a las cuales permite construir robots todo terreno, capaces de moverse por donde las ruedas no pueden, ya que proporcionan una mayor tracción.

– **Motores**

Este componente es el encargado de dotar de movimiento al robot. Se pueden diferenciar varios tipos según su forma de funcionamiento. Los mas utilizados para la robótica son los motores DC, los motores servo eléctricos y los motores paso a paso.

- **Motores DC**

  Este tipo de motor ofrece un movimiento lineal y rotatorio al aplicarle una diferencia de potencial en sus extremos.

- **Motores servo eléctricos**

  Este motor es similar al motor DC, con la variación de que no ofrece un movimiento rotatorio, en su lugar tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición.

- **Motores paso a paso**

  Este motor funciona a través de los impulsos eléctricos que recibe, realizando desplazamientos angulares discretos, lo que significa que es capaz de avanzar una serie de grados dependiendo de sus entradas de control.

- **Mecánica de control**

  Los motores suelen necesitar mucha más potencia que el resto de dispositivos, lo que provoca en comparación un consumo elevado de corriente. Este hecho hace que sea totalmente necesario su alimentación y control, de una forma independiente y separada al resto de componentes.

  Para ello se utilizan los denominados *puentes en H*, encargados de cumplir lo anteriormente descrito. Se pueden encontrar muchos dispositivos en el mercado, dependiendo del fabricante, de los componentes internos usados y de la potencia requerida por el motor, pero todos ellos funcionan de la misma forma.

- **Comunicación**

  El robot tiene que poderse comunicar con el exterior, ya sea inalámbricamente o por cable, para poder transmitir los datos procesados e incluso llegar a ser controlado desde un pc.

  - **Por cable**

    Esta manera de comunicarse consiste en establecer una conexión a través de un cable, pero cuando se habla de transmisión por cable se hace referencia a una conexión de datos lenta y relativamente corta en distancia. Idea provocada por el cable convencional de cobre.

    Alguno de los protocolos más usuales, utilizados en la comunicación por cable son:

    - Ethernet

- Serie
- Óptico

Las distancias de comunicación en las que estos protocolos funcionan son muy diversas.

Por ejemplo, en un enlace para dispositivos RS-232, donde se usa cable convencional de cobre, la distancia máxima entre dos nodos es de unos 15 metros, transmitiendo a una velocidad de 19200 Bps. Sin embargo una línea de fibra óptica puede transmitir a esa velocidad hasta a una distancia de 2,5 Km.

Por ese motivo, dependiendo de las características deseadas, variará el material del que está fabricado el cable y la forma, consiguiendo así transmitir más información a velocidades superiores.

- **Inalámbrica**

  Gracias a este tipo de comunicación se pueden establecer conexiones entre dispositivos de una forma fácil y rápida. Además hay que tener en cuenta que la velocidad de transmisión y la distancia máxima a la que puede funcionar es incomparable con la transmisión por cable.

  En este caso, hay varias formas de poder establecer una conexión, dependiendo de la velocidad y la distancia deseada a la que se quiere transmitir dicha información. A continuación se comentan las posibles formas de establecer una conexión inalámbrica.

  - **Comunicación por Wi-Fi**

    Este tipo de comunicación, de igual forma que la comunicación bluetooth, se realiza por radiofrecuencia lo que permite que los dispositivos no tengan que estar alineados entre si. Dicha comunicación puede llegar a transmitir a una velocidad máxima de 54 Mbits/s, y si se ayuda de una antena adecuada puede alcanzar distancias de asta 50 Km.

- **Comunicación por bluetooth**

  Este tipo de comunicación se realiza por radiofrecuencia lo que permite que los dispositivos no tengan que estar alineados entre si, puede llegar a ofrecer una conexión de asta 100 metros de distancia a una velocidad máxima de 24 Mbits/s.

- **Comunicación por RF**

  Este tipo de comunicación no requiere que los dispositivos tengan que estar alineados entre si. La distancia de transmisión esta limitada por el dispositivo que se use para realizar la comunicación, siendo necesario un dispositivo mas grande y de mayor potencia para una distancia mayor.

  A diferencia de los tipos de comunicación explicados anteriormente, los cuales disponen de un protocolo ya creado, en este caso el protocolo se ha de implementar, lo que hace que su uso sea algo más complejo

- **Comunicación por infrarrojos**

  Este tipo de comunicación consiste en una comunicación bidireccional punto a punto, empleando un haz de luz infrarroja que requiere que los dispositivos estén alineados, cosa que hace que la comunicación sea muy delicada. Dicha comunicación puede llegar a transmitir información a una velocidad máxima de 16 Mbits/s y una distancia máxima de un metro.

- **Inteligencia**

  Todos los componentes anteriormente nombrados tienen que estar controlados por un micro controlador, esto hace que sea autónomo, ya que se encargará de realizar todas las operaciones programadas basándose en los datos recibidos por los sensores.

Estos dispositivos pueden variar en velocidad de procesado, memoria y tamaño, además dependiendo del fabricante su programación será mas o menos compleja.

- **Alimentación**

  Para que el robot pueda funcionar es necesario que tenga una buena alimentación, capaz de cubrir todos los componentes instalados.

  Se puede alimentar a través del cable, directamente de la red eléctrica, o a través de unas baterías.

  - **Por cable**

    Tiene la ventaja de no depender de un tiempo límite de trabajo, y la desventaja de tener el alcance limitado a la longitud del cable.

  - **Por baterías**

    Esta forma ofrece la ventaja de no tener un alcance limitado, aunque su tiempo de funcionamiento depende del consumo del robot y la capacidad que ofrezcan las baterías.

3.2.COMPONENTES DE UN ROBOT PARA MAPEAR

Para construir un robot capaz de mapear se necesitan un conjunto determinado de sensores/actuadores. Hay que destacar que el número de posibilidades de construcción de robots para mapear es ilimitado, aunque en la construcción de un robot simple, pero eficiente, con unos pocos componentes se puede llegar a obtener unos resultados óptimos.

En este proyecto se ha construido un robot simple, para mapear, que utiliza los componentes de bajo coste explicados a continuación.

3.2.1. SENSORES.

En este apartado se explican los sensores usados para poder realizar el mapeo.

- **Sensor de infrarrojos GP2D12**

  El sensor GP2D12 es un sensor capaz de detectar objetos entre 10 y 80 centímetros, medida la cual facilita mediante su salida analógica. Este tipo de sensor se basa en el principio de triangulación para realizar las medidas. El led infrarrojo emite un haz que será rebotado por el objeto y posteriormente recogido por el PSD (*Position Sensing Device*, Dispositivo de Percepción de Posición). Dependiendo del ángulo de incidencia del haz rebotado en la lente, se activa una u otra célula del *array* lo que permite estimar la distancia a la que se encuentra el objeto.

Ilustración 3: Sensor GP2D12

- **Sensor de ultrasonidos MAXSONAR – EZ1**

  El MaxSonar-EZ1 es el sónar más pequeño y de menos consumo del mercado. Es capaz de detectar objetos desde 0 hasta 6,45 metros, medida la cual puede facilitar mediante su salida analógica, serie o generando un pulso PWM. Para ello transmite una señal ultrasónica de 42 Khz y, mediante un contador interno, calcula el tiempo que tarda dicha señal en regresar. Si el tiempo es superior a 37,5 ms significa que no se ha detectado obstáculo alguno.

Ilustración 4: Sensor MaxSonar-EZ1

- **Brújula digital CMPS03**

  La brújula digital CMPS03 es un sensor de campos magnéticos que una vez calibrado ofrece una precisión de 3-4 grados y una resolución de decimas. El valor que obtiene del campo magnético lo puede facilitar mediante un pulso PWM o una conexión I2C. Para ello se usan dos sensores KMZ51 de Philips colocados en ángulo de 90 grados uno respecto al otro, de esa forma, permite al microprocesador calcular la dirección de la componente horizontal del campo magnético natural.

Ilustración 5: Brújula digital CMPS03

- ***Encoder* ENC01A**

  El *encoder* ENC01A es un codificador de cuadratura, que mediante dos sensores infrarrojos de reflectancia en el interior de la rueda, mide el movimiento de unos dientes incorporados en la llanta de la misma. Los sensores están espaciados para proporcionar formas de onda de aproximadamente 90 grados fuera de fase, permitiendo determinar el sentido del giro. Este sensor, junto con las ruedas especificas para él, es capaz de proporcionar una resolución de 48 pulsos por vuelta. Cada pulso del sensor se obtiene mediante una salida digital.

Ilustración 6: *Encoder* ENC01A

3.2.2. LOCOMOCIÓN

En este apartado se explican los componentes usados para poder desplazar el robot.

- **Ruedas dentadas**

  Este tipo de rueda consiste en una rueda común pero que esta especialmente diseñada para el uso de los *encoders* ENC1A, ya que en su interior incorpora una especie de *dientes* para que los *encoders* puedan detectar su giro, además por su diseño esta preparada para disminuir la posibilidad de derrape. Tiene un diámetro de 4,2 cm, lo que hace que por cada vuelta completa recorra unos 13,2 cm.

Ilustración 7: Ruedas dentadas

- **Motor DC con reductora**

  Estos pequeños motores, de alta calidad, están dotados de una caja reductora para otorgar mayor fuerza de torsión sacrificando velocidad lineal y están pensados para trabajar a 6 V, aunque, por lo general pueden funcionar con tensiones de 3 a 9 V. A su tensión nominal, dichos motores pueden llegar a operar a 1300 RPM.

Ilustración 8: Motor con caja reductora

- **Puente en H L293D**

  El integrado L293D incluye cuatro circuitos para manejar dos motores de forma independiente, de forma que de cada motor se pueda controlar la velocidad, el sentido e incluso poder llegar a frenar el motor. Además dicho dispositivo mantiene separada la corriente y tensión de los motores, del resto del circuito. En la siguiente ilustración se muestra un esquema del funcionamiento interno de un puente en h común.

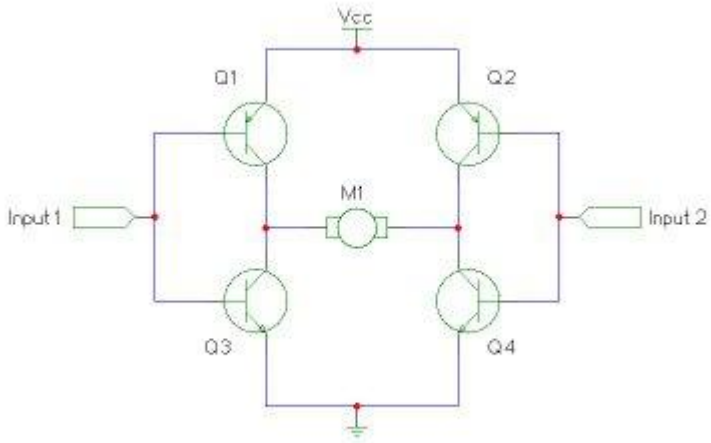


Ilustración 9: esquema de un puente en H

3.2.3. COMUNICACIÓN

Para el funcionamiento de este robot se usan tres tipos de comunicación diferentes. A continuación se explica cada uno de ellos y en que situación se utilizan.

- **Comunicación RS-232**

  Este tipo de comunicación es la usada para conectar el robot con el PC, y de esa forma poder enviar las ordenes a realizar y los datos obtenidos.

  Dicho protocolo consiste en enviar un dato detrás de otro acompañado de los bits de inicio, de fin de comunicación y sus correspondientes comprobantes de envío, es decir, los ACK. Esto hace que sea un sistema de transmisión simple de

usar pero algo lento, ya que por cada byte enviado se requiere de 2 bites más como mínimo. Esto junto el tiempo de espera del ACK, hace que el protocolo RS-232 utilizado en el bus serie no sea eficiente para enviar muchos datos en un tiempo mínimo. De todas formas, contando que el robot únicamente necesita enviar 4 bytes cada 40 ms, y recibir 1 byte para la instrucción de comienzo y otro para la de final, hace que este sistema de comunicación sea mas que suficiente.

- **Comunicación bluetooth**

  Este tipo de comunicación es inalámbrica, y por ese motivo se usa para liberar el robot del cable necesario para transmitir los datos al PC. Así pues la finalidad de este dispositivo es codificar el protocolo RS-232 para poder establecer comunicación entre el pc y el robot a una velocidad de 38400 baudios.

- **Comunicación I2C**

  Este tipo de comunicación consiste en un bus específico de control para instrumentación electrónica. Este bus es un bus serie que puede ser multimaestro y solo necesita de dos líneas para funcionar, la *sda* (línea de datos) y *scl* (línea de reloj), además permite direccionar 128 dispositivos y trabajar a frecuencias de entre 100 y 400 Khz.

  En este caso la comunicación con la brújula digital se ha realizado mediante el bus I2C, ya que ofrece unas características necesarias para el correcto funcionamiento del robot. En la ilustración 10 se puede observar que para comenzar la comunicación con la brújula hay que poner el valor del bit de comienzo a 0, seguidamente mandar la dirección de la brújula, en este caso la 0xC0. Cuando se recibe el bit ACK se le envía el número del registro del cual se quiere obtener la lectura, y seguidamente se vuelve a esperar el bit ACK. Una vez recibido, se vuelve a enviar la dirección de la brújula, pero esta vez con el bit menos significativo a 1, es decir, 0xC1 ya que de esa forma se le indica a la brújula que queremos leer el registro anteriormente solicitado y por última vez se vuelve a esperar al bit ACK. Cuando dicho bit se ha recibido se envía a la

brújula, por ultimo, la cantidad de bytes que deseamos leer y en cuanto recibamos el ACK se pone el Stop bit a 1, indicando que estamos listos para recibir el dato del registro solicitado.

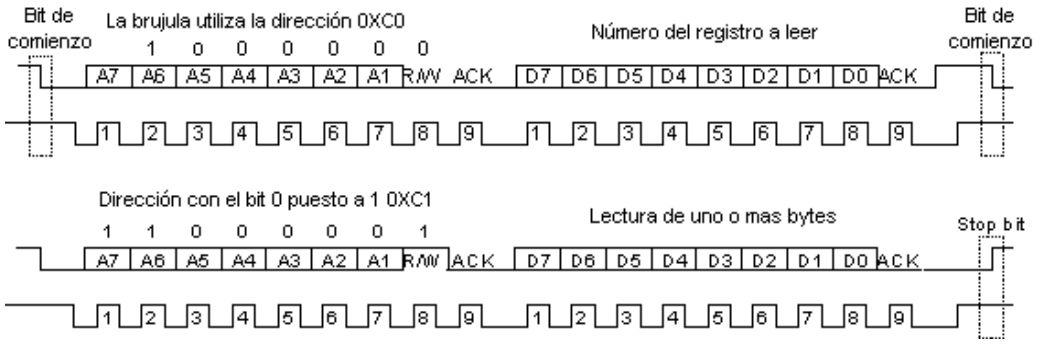


Ilustración 10: Comunicación de la brújula mediante I2C

3.2.4. INTELIGENCIA

Todos los componentes anteriormente nombrados están conectados al micro controlador ATMega128, el cual es el encargado de realizar las operaciones necesarias para su debido control. Dicho micro controlador es una máquina hardware que funciona a una frecuencia de 16 Mhz, con 128 Kbytes de memoria para código y 4 Kbytes de RAM y viene montado sobre una placa ET-AVR debido a que su arquitectura es en encapsulado SMD. Dicho micro controlador dispone de multitud de dispositivos integrados, como dos *timers* de 8 bits y dos de 16 bits, capaces de generar hasta 8 señales PWM, un conversor analógico/digital, capaz de multiplexar hasta 8 de las 54 conexiones de las que dispone y dos puertos serie entre otras cosas.

3.2.5. CONSUMO DEL ROBOT

El consumo medio del robot a pleno rendimiento es de 350 mA, repartidos tal y como se muestra en la siguiente tabla.

COMPONENTE	CONSUMO MEDIO
Placa ET-AVR con ATMega128	30mA
Conjunto de sensores	100mA
Motores	160mA
Comunicación Bluetooth	60mA

Ilustración 11: Tabla de consumo

Para cubrir dicho consumo se utilizan 4 baterías de 1,5v de tensión y con una capacidad de 1500 mAh, por lo tanto, teniendo en cuenta el consumo anteriormente nombrado, con dicha alimentación, el robot puede tener una autonomía a pleno rendimiento de unas 4 horas.

3.3.MONTAJE DEL ROBOT

El robot presenta una composición modular, compuesta de tres partes. La parte inferior, donde se encuentran los componentes destinados a la locomoción y fuente de energía. La capa intermedia, destinada al control, donde va alojado el micro. Y la capa superior, donde irán todos los periféricos destinados a la captación de medidas.

En un principio, tal y como se muestra en la ilustración 12, se utilizaron dos motores ubicados en la parte inferior del robot y conectados al L293D, que serán gobernados por el micro a través de un pulso PWM con una frecuencia de 2 Khz, ya que se encuentra dentro de los valores óptimos según las especificaciones del fabricante de este dispositivo.

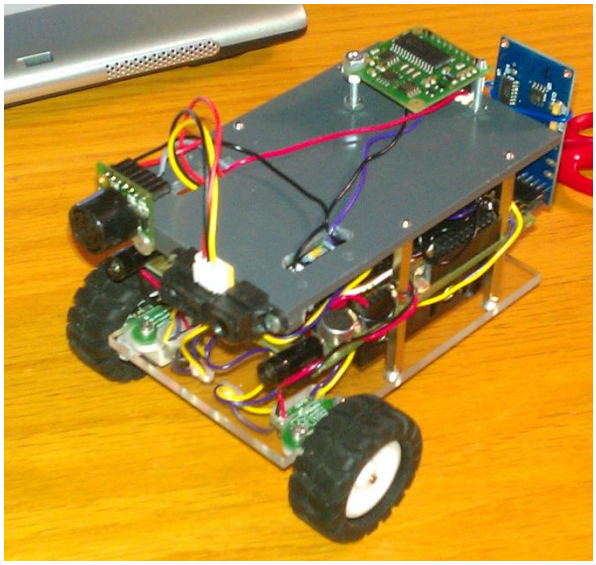


Ilustración 12: Montaje inicial

A continuación, se les acopló unas ruedas, ambas situadas en la parte frontal del robot. Junto a estas se añadió los *encoders*, cuya función es medir la distancia recorrida por el robot, y de los cuales se utilizó únicamente una de las dos señales que proporciona. Dicha señal se conectó como interrupción externa, usando tanto el flanco de subida como el de bajada, para así disponer de 1 pulso por cada 5,5 milímetros recorridos.

En la parte posterior se le adjunto una bola, teniendo ésta la capacidad de girar en todos lo sentidos.

Para indicar la dirección del robot se instaló la brújula digital CMPS03 en la parte superior, retirado de la posición de los motores, para así evitar las posibles interferencias que se pudieran ocasionar. Dicha brújula se conecta utilizando el código I2C, ya que el tiempo necesario para adquirir una lectura tiene que ser inferior a los 102 ms que ofrece la conexión PWM, con la que podrían haberse producido perdidas en la precisión de la posición, y de esta manera se obtiene una mejor resolución.

Seguidamente, en la parte frontal de la parte superior, se colocaron dos sensores, el sensor de infrarrojos (IR) y el sensor de ultrasonidos (UL) para poder realizar el correspondiente mapeo (recogiendo cada 4 ms todas las medidas detectadas). En la parte intermedia se ubicó la placa ATMega a la cual se le conectó los anteriores dispositivos nombrados (motores, brújula…) y las salidas analógicas de los sensores a los pines correspondientes con el conversor analógico digital. Esta placa es la

encargada de recoger todos los datos y enviarlos cada 40 ms a través del puerto serie usando un dispositivo bluetooth.

Más adelante, debido a inconvenientes que surgieron durante las pruebas en el campo, se decidió cambiar la bola trasera por dos ruedas más acopladas a 2 motores, ya que se observó que a causa de ella, el robot, desviaba su trayectoria. Así pues, el robot final consta de 4 ruedas motrices, conectadas 2 a 2 en paralelo, de forma que para el controlador es como si tan solo hubiera 2 ruedas, derecha e izquierda. También se cambió la ubicación de los sensores, para facilitar la realización del mapeo, dejando así el sensor de ultrasonidos en la parte frontal y el sensor de Infrarrojos en la parte lateral, pero en este caso obtuvimos unos resultados poco concordantes, así que acabamos substituyendo el sensor de ultrasonido por un sensor de infrarrojos, obteniendo así la estructura definitiva tal y como se muestra en la ilustración 13.

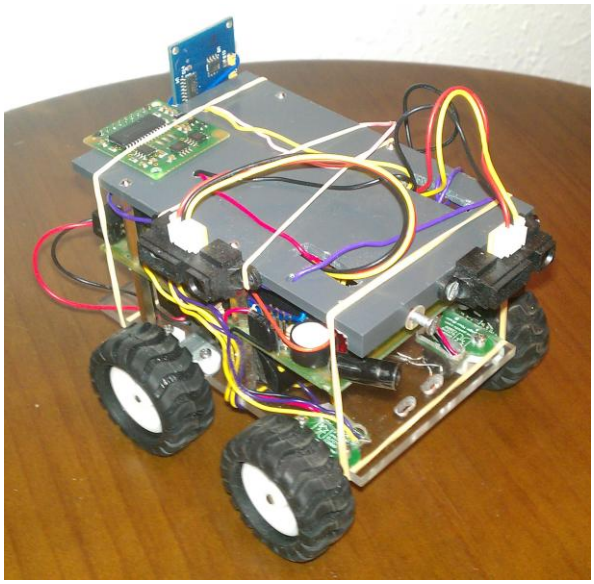


Ilustración 13: Montaje final

CAPÍTULO 4. ALGORITMOS DE CONTROL DEL ROBOT

Gran parte del funcionamiento del robot esta sujeto a los algoritmos que controlan el movimiento y la recepción de los datos. Estos algoritmos se implementan en *C* y se cargan en el micro controlador mediante la herramienta *AVRStudio*. En este caso, el micro controlador que se usa es el ATMega128, tal y como se ha explicado anteriormente.

Hay que destacar que del correcto funcionamiento de estos algoritmos depende buena parte de la precisión conseguida en el mapeo, ya que sin un buen control de los dispositivos, se pueden llegar a producir un cumulo de errores. Por ese motivo se describe el funcionamiento de estos algoritmos.

4.1.ALGORITMO DE CONTROL DE LOS MOTORES

Tal y como se a explicado anteriormente los motores funcionan a través de un pulso PWM generado y controlado por el micro controlador. Dicho pulso se genera gracias a uno de los *timers* del controlador ATMega, el cual después de programarse, realiza de forma automática el cambio del valor del pin, de alta a baja y viceversa. De esta forma se puede generar un pulso, la frecuencia del cual se determina con la velocidad a la

que el timer realiza el cambio del valor del pin. Dicho pin se conecta al L293D (puente en H) de forma que activa los motores cuando recibe un pulso en alta. Así pues, siguiendo las especificaciones óptimas del fabricante de dicho puente en H, se genera un pulso con una frecuencia de 2 KHz, el cual es el encargado de controlar los motores.

4.2.ALGORITMO DE CONTROL DE LA BRÚJULA DIGITAL

Inicialmente, la brújula se conectó a la placa ATMega usando la salida que proporcionaba un pulso PWM, donde el tiempo que estaba en alta era igual a los grados medidos, de forma que si variaban los grados variaba el tiempo del pulso en alta. Para poder medir dicha variación de tiempo se conecta la brújula a dos pines del controlador ATMega, correspondientes con las interrupciones externas. De esta forma, después de haber configurado correctamente las interrupciones, cuando se realiza el cambio del pulso de baja a alta, se genera una interrupción en el micro controlador. En ese momento se inicia uno de los contadores del ATMega, de forma que cuando se realiza la interrupción generada por el cambio del pulso de alta a baja, se detiene el contador y se sabe el tiempo que ha estado en alta, sabiendo de esa forma los grados medidos.

Pero este método tiene como inconveniente que para cada medida realizada es necesario 65ms más el tiempo utilizado del pulso en alta, correspondiente con los grados medidos, el cual puede variar de 1 ms a 37 ms. De esa forma el tiempo necesario para cada medida realizada varía de los 66 ms a 102 ms.

Entonces después de los resultados anteriormente descritos y contando que interesa realizar una medida en el menor tiempo posible, se optó por conectar la brújula utilizando la conexión I2C, ya que dicha conexión puede facilitar un dato cada 10 µs.

4.3.ALGORITMO DE CONTROL DE LOS ENCÓDERS

Los *encóders* ENC01A generan 2 señales, de las cuales únicamente se utiliza una de las 2, concretamente la señal A.

Dicha señal se conecta al micro controlador ATMega como interrupción externa, pero inicialmente, solo se conectó de flanco de subida, de forma que cuando el pulso

cambiaba de baja a alta se generaba una interrupción, que a su vez incrementaba una variable. Finalmente se decidió conectar esa misma señal también como flanco de bajada, de forma que cuando el pulso cambie de alta a baja también se incrementase la misma variable, así se obtendrá el doble de resolución por cada pulso realizado.

4.4.ALGORITMO DE CONTROL DE LOS SENSORES

En un principio, tal y como se ha explicado anteriormente, el robot contaba con un sensor de ultrasonidos y otro de infrarrojos, pero finalmente se optó por utilizar dos sensores de infrarrojos.

En todo caso en ambos situaciones el algoritmo de control es el mismo, ya que se usa la salida analógica que proporcionan los sensores. Así pues, dichas señales se conectan a las entradas analógicas del micro controlador ATMega, y se configura el conversor analógico-digital de forma que realice las conversiones seguidas, es decir, en cuanto acaba una conversión inicia la otra. Entonces para poder usar dos o más sensores, se optó por hacer una instrucción en el micro controlador de forma que cada vez que realice una conversión cambie de sensor.

4.5.ALGORITMO DE MAPEO

El algoritmo de mapeo consiste en una serie de instrucciones, que juntamente con los algoritmos de control de los componentes anteriormente descritos, hace que sea posible mapear.

Así pues, una vez establecida la conexión con el pc, el robot, espera a que se le de la instrucción de comenzar, correspondiente con la tecla “c”.

En cuanto el robot recibe dicha instrucción, activa la función de comenzar, cuya función consiste en activar los motores hacia delante y mantener dicho estado mientras no detecta ningún objeto a unos 40 cm frente a él. Una vez detectado cualquier cosa a esa distancia, el robot comienza a girar hacia la izquierda, y se mantiene realizando dicha operación mientras el sensor delantero detecta algo a una distancia de unos 60 cm, de esa forma se consigue que el robot quede casi paralelo al objeto detectado. Una vez el sensor delantero no detecta nada dentro de los valores

establecidos, se activa el sensor lateral al mismo tiempo que un *timer* responsable de enviar todos los datos recogidos por los componentes sensoriales cada 40 ms. Hay que destacar que en caso de que el robot realice un giro correspondiente con aproximadamente 180^{o} y aun así el sensor delantero siga detectando algún objeto, se activará el sensor lateral y el *timer* responsable de enviar los datos obtenidos, para conseguir de esa forma poder mapear un espacio relativamente pequeño, aunque seguirá girando asta que el sensor delantero deje de detectar algo a unos 60 cm.

Una vez activado el sensor lateral y el *timer* de envío, se activa el motor hacia delante. Durante este proceso en todo momento se observan los valores recogidos por el sensor lateral, de forma que si la distancia al objeto es superior a 42 cm o inferior a 38 cm, el robot rectificará dicha distancia acelerando o frenando los motores del lado correspondiente, en caso contrario mantendrá la velocidad prestablecida.

El proceso anteriormente descrito se realizará siempre y cuando el sensor delantero no detecte nada a una distancia de unos 40 cm. En caso contrario detendrá los motores, y realizará el giro a la izquierda de la misma forma que se ha explicado anteriormente.

Así pues comenzaríamos de nuevo todo el proceso descrito. Este bucle únicamente se detendrá cuando desde el PC se le envíe la instrucción de parar, sino lo repetirá tantas veces como sea necesario.

Hay que destacar que en todo momento se le puede dar la orden de parar pulsando la tecla ‘p’, de esa forma podemos detener el robot cuando se desee.

Todo este proceso queda descrito gráficamente en la siguiente ilustración, y más detallado en el anexo I.

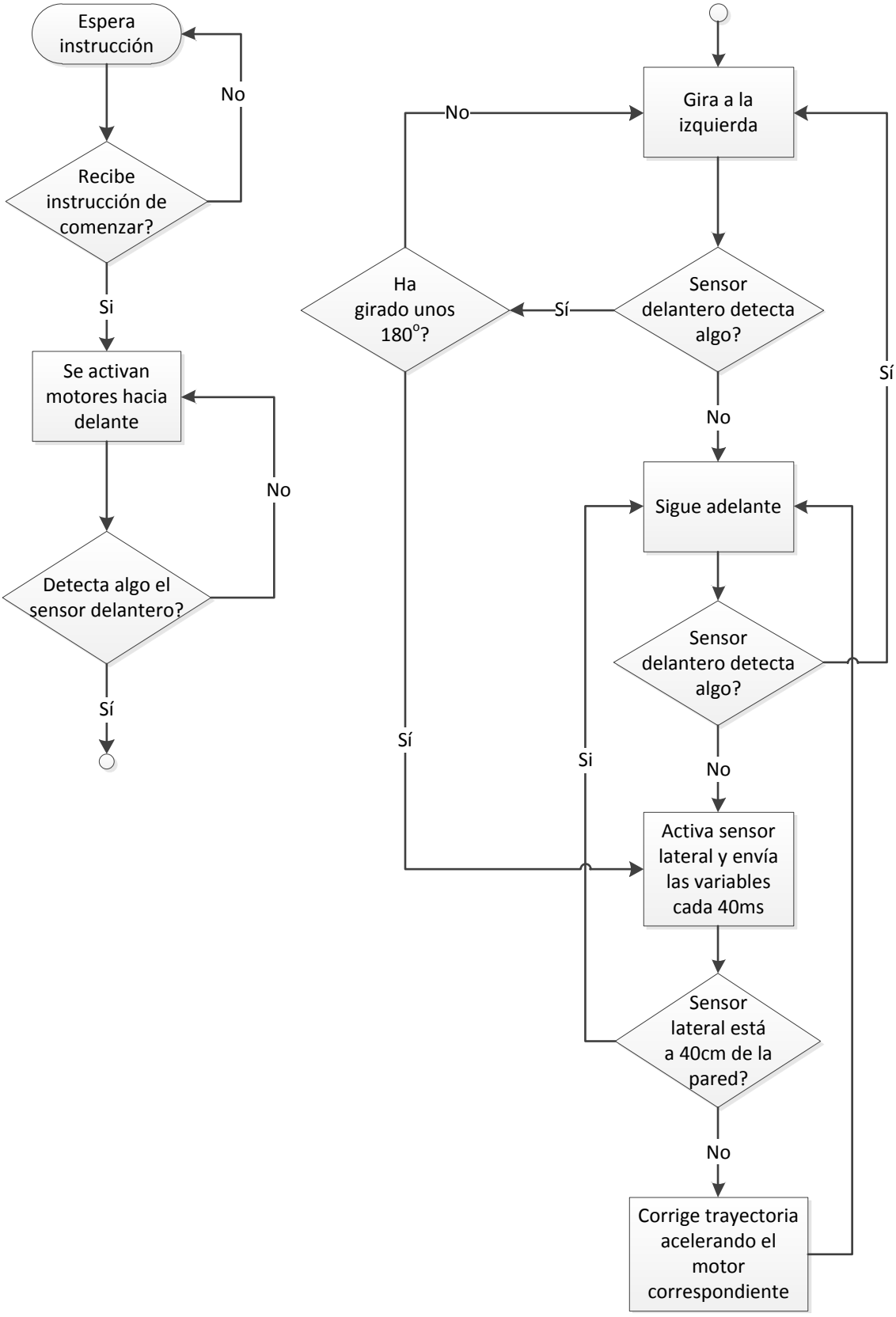


Ilustración 14: Diagrama del algoritmo de mapeo

CAPÍTULO 5. PRUEBAS REALIZADAS

A continuación se explican todas las pruebas realizadas para conseguir hacer funcionar correctamente los dispositivos comentados anteriormente.
Así pues se realizaron pruebas para calibrar la brújula correctamente, para determinar el rango de funcionamiento de los sensores y también para fijar la velocidad adecuada de los motores.

5.1.CALIBRACIÓN DE LA BRÚJULA DIGITAL CMPS03

Después de varias pruebas sobre el campo, se observó una pequeña variación en los grados medidos por la brújula, de tal forma que hace que los resultados obtenidos no sean del todo correctos. Por lo tanto se decidió realizar una medición cada 5^{o} reales, tomados de una brújula, y compararlos con los grados medidos por la brújula CMPS03.
Dichos resultados se graficaron para poder apreciar de una forma mas directa la variación de la brújula CMPS03 con los valores reales obtenidos por la brújula utilizada para la comparación. La siguiente ilustración muestra dicha gráfica, donde podemos ver la recta decreciente correspondiente con la brújula de calibración junto con los resultados obtenidos por la CMPS03.

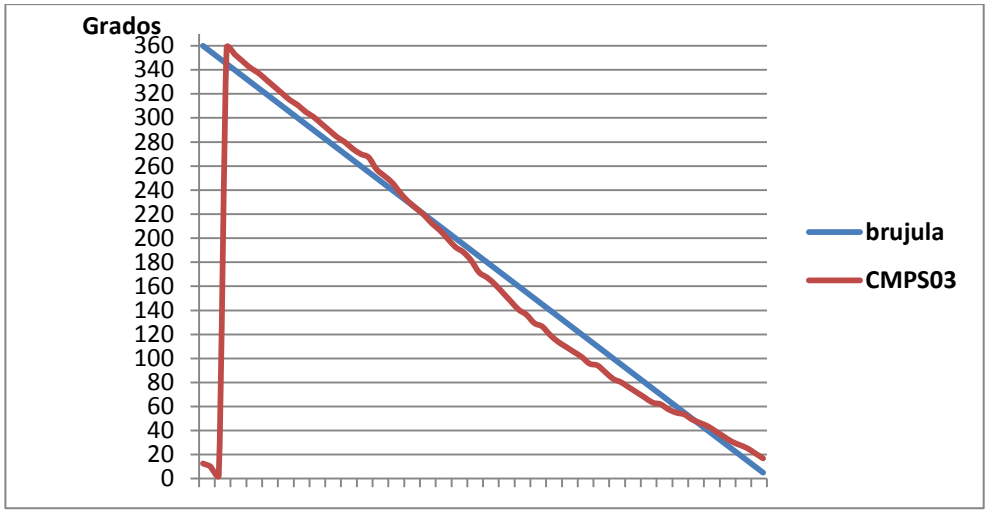


Ilustración 15: Comparación de medidas de la CMPS03

Como se puede ver en la ilustración 15 la brújula digital CMPS03 varia ligeramente entre los intervalos de 40° a los 0° y entre los 240° a los 360°, entre los valores correspondientes con 200° y los 240° apenas se diferencia, de la misma forma que entre los 60° y los 30°. Sin embargo, entre los valores restantes la variación es significativa. Estas diferencias de variación se pueden apreciar mejor en la ilustración 16, la cual muestra el resultado de restarle a la brújula CMPS03 el valor real obtenido por la brújula utilizada para la calibración.

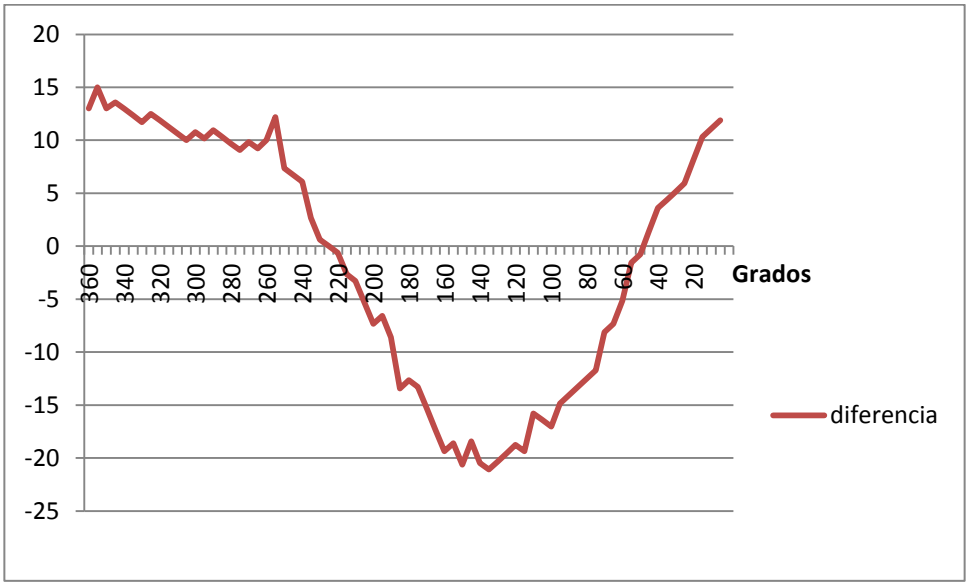


Ilustración 16: Diferencia de grados de la CMPS03

Una vez obtenidos estos resultados se decidió modificar el código del programa que recibe los datos de la brújula, de forma que dicho programa recibe el valor que mide la brújula CMPS03, y aplica un ajuste para aproximar lo mejor posible la medida de la brújula con el valor real.

Dicha variación se puede apreciar en el anexo II, donde se encuentra el código completo del programa encargado de recibir y guardar todos los datos de los sensores.

5.2.CALIBRACIÓN DE LOS SENSORES

Para un correcto funcionamiento del robot es necesaria la calibración de todos los dispositivos que lo componen, así como los sensores de captación de medidas, tales como el sensor de infrarrojos y el de ultrasonidos, los sensores encargados de medir el giro de la rueda, tales como los *encóders*, o el sensor encargado de medir la dirección del robot, tal como la brújula digital. De este último ya se ha explicado la calibración que se llevó a cabo para su correcto funcionamiento, por lo tanto a continuación se explica como se realizó la calibración del resto de dispositivos.

- **Sensor de infrarrojos**

  Según las especificaciones del fabricante el rango de funcionamiento de este dispositivo es de los 10 a los 80 cm, por lo tanto se decidió hacerlo funcionar entre los 35 y 45 cm, ya que dicho valor se encuentra cerca de la mitad de su intervalo de trabajo.

  Así pues se realizaron un seguido de pruebas para saber el valor digital cuando el sensor mide las distancias anteriormente nombradas ya que este sensor será el responsable de realizar el mapeo.

  Después de varias pruebas se observo que el conversor estaba mal configurado, y que requería un ajuste de la capacidad interna, para así poder tener el valor máximo del sensor igualado con el valor máximo del conversor.

  Una vez realizado este ajuste se continuó con las pruebas para la calibración del sensor.

  Así pues la primera prueba realizada después de la calibración del conversor consistió en colocar el sensor frente a una pared y realizar unas 1000 medidas

cada centímetro dentro de los intervalos nombrados. Una vez recogidos todos los datos se realizó la media aritmética y se representó en una gráfica, la cual se muestra en la siguiente ilustración.

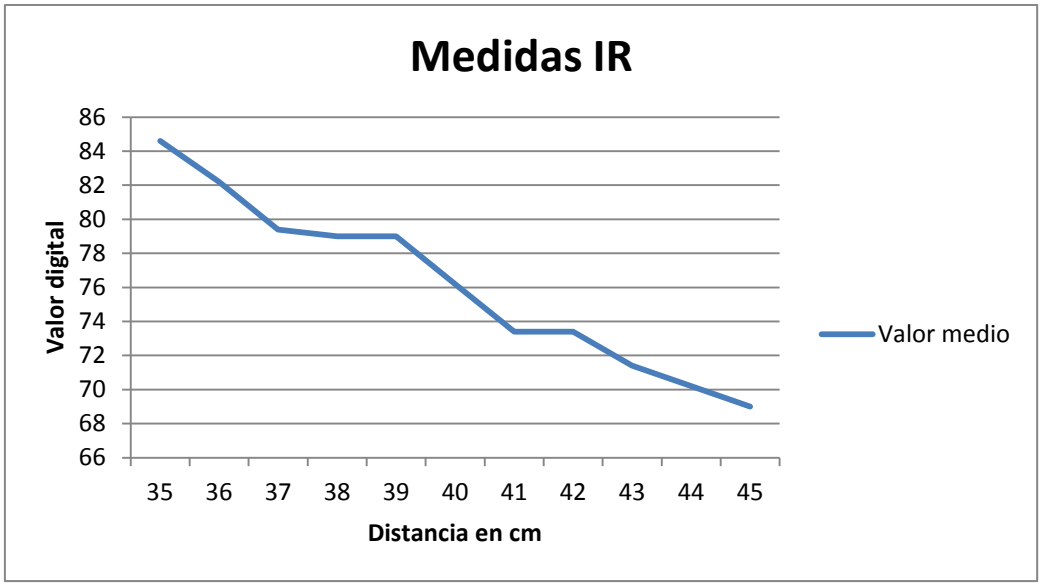


Ilustración 17: Valor medio digital del IR

Por lo tanto observando los valores representados en la ilustración 17 podemos apreciar como cuanto mas lejos está el sensor del objeto menor es el valor que recibimos, así como también podemos deducir que el rango de trabajo del sensor de infrarrojos estará comprendido entre el valor 73 y el 79.

Más tarde se realizó otra prueba, y tal vez la más importante, para saber si el dispositivo cumplía los requisitos deseados para poder realizar el objetivo principal de este proyecto, es decir, mapear.

Dicha prueba consistió en hacer movimientos paralelos a una pared, a lo largo de un metro tomando un total de unas 1000 medidas. Ese recorrido se repitió para cada distancia. Dichas distancias se comprendían entre los 30 y 50 cm con intervalos de 5 cm.

Dichas medidas se muestran en la ilustración 18, donde se puede apreciar claramente que cuanto mas lejos esta el objeto menor es el valor que obtiene el sensor.

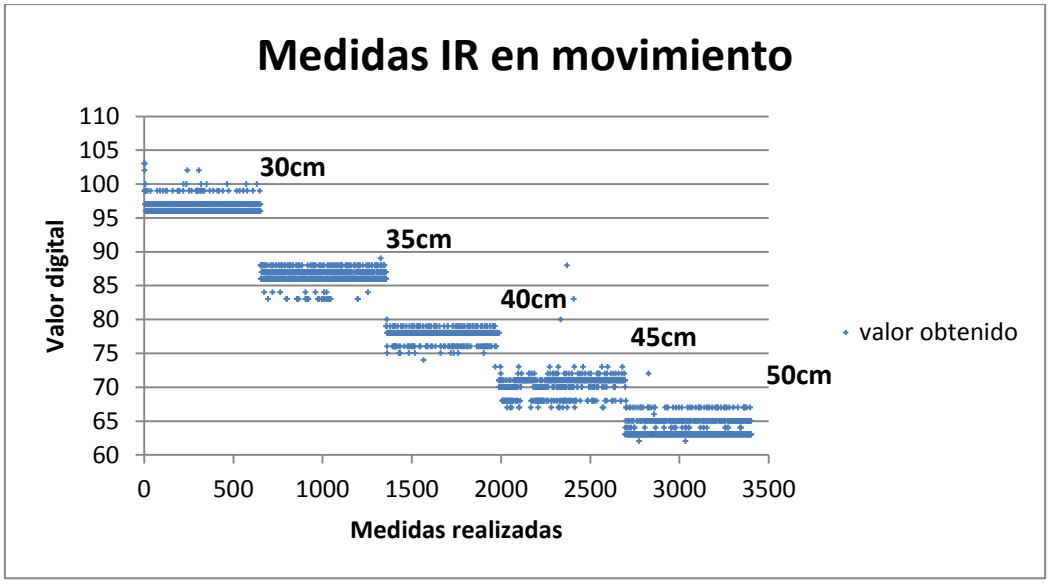


Ilustración 18: Valores digitales del IR en movimiento

También se puede ver una linealidad entre las medidas tomadas, lo que hace que el uso del sensor sea más eficiente y más preciso. También se puede ver que para las distintas medidas el sensor a utilizado una diferencia de 40 valores, algo muy significativo, ya que gracias a esta gran variación de valores se puede tener casi un centímetro de precisión.

- **Sensor de ultrasonidos**

  En este caso este sensor tiene un rango de trabajo de los 0 a los 6,45 metros, pero como el sensor de infrarrojos trabajará entorno a los 40 cm, se decidió realizar las pruebas necesarias para calibrar este sensor entorno a la misma distancia que el IR.

  Así pues se realizó las mismas pruebas que con el sensor de infrarrojos.

  En la ilustración 19 se muestran las medidas obtenidas en la primera prueba, donde se puede apreciar una clara diferencia con el sensor de infrarrojos, que contra mas lejos esta el objeto mayor es el valor que proporciona.

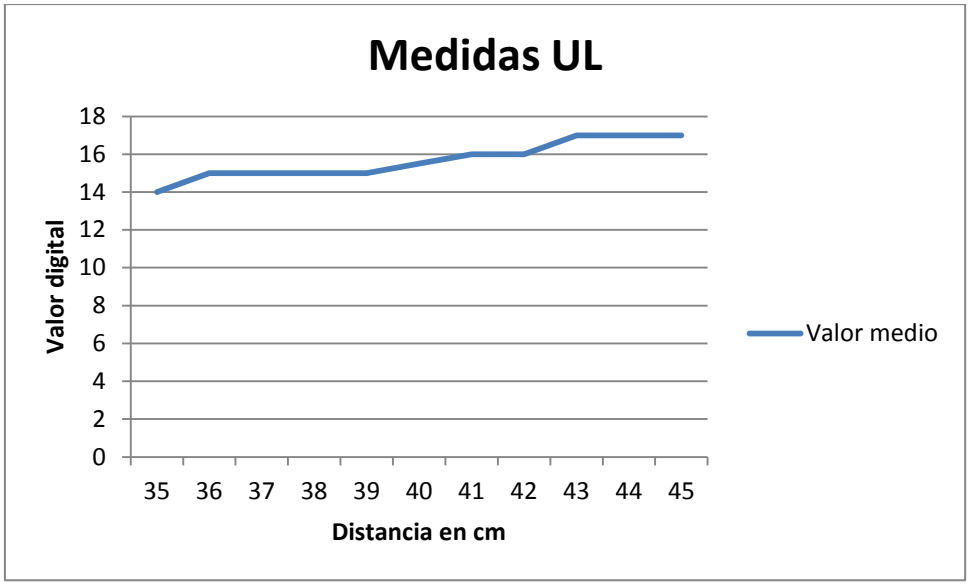


Ilustración 19: Valor medio digital del UL

Además en este caso la variación del sensor para las distancias deseadas es mínima, lo que hace que el sensor sea poco preciso y por ese motivo se descartó este sensor para realizar el mapeo.

Aun así el sensor de ultrasonidos por sus altas prestaciones en la distancia medida, hasta 6,45 metros, podía ser utilizado para evitar que el robot chocara con objetos frontalmente, ya que para este propósito no hace falta mucha precisión.

Así pues se decidió realizar la segunda prueba del sensor realizada y explicada con el sensor de infrarrojos, es decir, recoger medidas cuando el robot esta en movimiento.

En la ilustración 20 se muestran los resultados obtenidos para las mismas distancias que el infrarrojo, donde se puede apreciar de forma rápida que la precisión del sensor es casi nula.

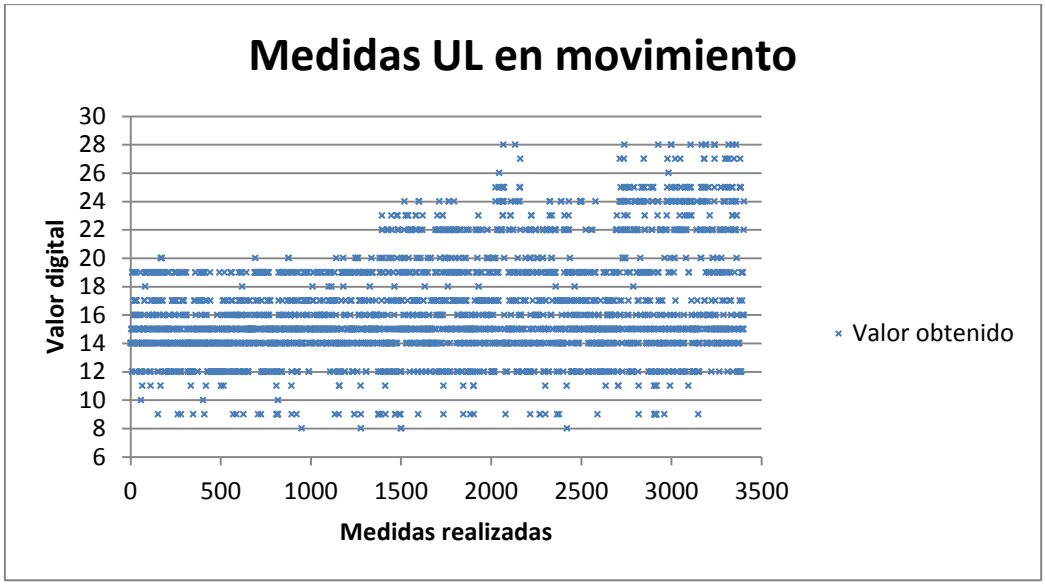


Ilustración 20: Valores digitales del UL en movimiento

En este caso no se puede diferenciar el límite donde acaba una medida y comienza la siguiente, y aun habiendo una diferencia de 20 cm en las distancias medidas, los valores obtenidos carecen de sentido. Aun así se puede apreciar un débil aumento en los valores recogidos por el sensor, pero nada precisos, ya que la dispersión es muy elevada.

Así pues, después de estos resultados, se llegó a la conclusión de que el ultrasonido cuando el robot esta en movimiento disminuye considerablemente su precisión, haciendo que sea imposible determinar un valor para una distancia tomada, por lo tanto se descartó completamente su uso.

En su lugar se opto por usar otro sensor de infrarrojos para detectar los obstáculos frontales.

- ***Encoders***

  Para la calibración de los *encoders* es necesario saber la distancia recorrida por cada vuelta realizada de las ruedas, y para ello se aplicó la ecuación del diámetro de una circunferencia, es decir, $2 \cdot \pi \cdot r$ donde r corresponde al radio de la rueda.

Por lo tanto como según el fabricante, los *encóders* generan 24 pulsos por vuelta y siendo el radio de la rueda de 2,1 cm, aplicando la ecuación anterior se obtiene que, por cada vuelta, el robot se desplaza 13,19 cm. Así pues aplicando una simple regla de tres se obtiene que por cada pulso realizado el robot se desplaza casi 0,55 cm.

Seguidamente se realizaron pruebas en el campo, donde se observó una pequeña variación de las medidas anteriormente descritas. Esta variación se debe a que la alineación de las ruedas no es perfecta, y por lo tanto las ruedas giran más de lo que el robot se desplaza.

Una vez detectado este problema, se programó el robot para que cuando el *encóder* llegase a medir 146 pulsos, equivalente a 80 cm, detuviera el robot.

Así pues, una vez programado, se hizo en el suelo un recorrido con una distancia de 80 cm, y se puso el robot al inicio de dicho recorrido. Seguidamente después se puso en marcha el robot, y se observó que antes de alcanzar los 80 cm del recorrido, el robot se detuvo. Entonces se midió la distancia que le faltaba para llegar al final del recorrido. Una vez hecho esto se aplicó una simple regla de tres para saber cuanto se desviaba por cada pulso realizado por el *encóder*. Entonces si por 146 pulsos se desviaba 3,7 cm, por cada pulso realizado se desvía unos 0,025 cm.

Una vez calculado el error hay que restarlo al valor tomado por el *encóder*, ya que la distancia que recorría era inferior a la real.

Por lo tanto si queremos conservar todos los decimales en las operaciones la ecuación final que se le aplicará al valor de los *encóders* es la mostrada a continuación.

$$valor\ en\ cm = (valor\ del\ encoder \cdot 4{,}2 \cdot \pi / 24) - (valor\ del\ encoder \cdot 0{,}02586)$$

Dicha ecuación se puede observar en el anexo II correspondiente con el programa de captación de los sensores del robot.

En este programa se muestra la formula en milímetros, ya que de esa forma se consigue una precisión mayor.

5.3.CALIBRACIÓN DE LOS MOTORES

La calibración de los motores consiste en determinar una velocidad adecuada para el funcionamiento del robot, ya sea para los movimientos de adelante y atrás, así como para los de giro.

Dicha velocidad viene determinada por el pulso PWM, generado por los *timers* del microprocesador ATMega, de forma que si aumentamos el tiempo a comparar por el *timer* obtenemos una velocidad menor y viceversa.

Así pues para determinar una velocidad adecuada se tuvo que llevar a cabo varias pruebas en el campo, consistentes en programar una velocidad y observar su comportamiento. Una vez realizadas varias pruebas, se determinó que a menor velocidad su funcionamiento seria más eficaz, ya que en un mismo espacio se podrían realizar más medidas.

Cuando se determinó la velocidad adecuada para su funcionamiento, tanto para desplazarse hacia delante y hacia atrás, como para realizar los giros sin que se quedase atascado, se tuvo que resolver un problema observado al realizar las pruebas.

Este problema consistía en el desplazamiento lateral involuntario del robot hacia la izquierda, debido a que la alineación de las ruedas no era perfecto y a que los motores de cada lado funcionan de forma inversa, es decir, si el motor derecho gira en sentido horario el izquierdo gira en sentido anti horario.

Este inconveniente, aunque parezca una tontería, provoca que aun aplicando el mismo PWM a los dos motores, cada uno gire a una velocidad diferente, que aun siendo mínima la diferencia, al sumarle el posible desvío en la alineación, hace que el robot se desvíe hacia la izquierda 30 cm por cada metro realizado.

Así pues para corregir este error simplemente se decidió acelerar el motor izquierdo de una forma progresiva e ir realizando pruebas asta observar que el desplazamiento involuntario del robot era casi nulo.

CAPÍTULO 6. RESULTADOS

Una vez realizados todos los puntos anteriores, el robot ya estaba preparado para realizar su cometido, mapear una superficie.

Hay que destacar que todos los resultados parciales de la caracterización de los componentes se explican en el *capitulo 7* de la memoria *Mapeo con robot móvil: Caracterización y modelado [1]*.

Ambos capítulos forman el núcleo de los resultados obtenidos en el proyecto, donde en esta memoria se ha explicitado, principalmente, la construcción del robot y los algoritmos de control del mismo, y en la memoria [1] se hace más referencia en el análisis de los datos y cálculos finales de mapeo.

Puesto que la extracción de datos se encuentra en la memoria [1] aquí solo se exponen los resultados más relevantes que han sido fruto de los datos obtenidos con el robot.

Así pues después de varias pruebas se obtuvo la gráfica mostrada en la ilustración 21 donde se puede apreciar el recorrido del robot en color azul y los datos recogidos por el sensor lateral en color verde.

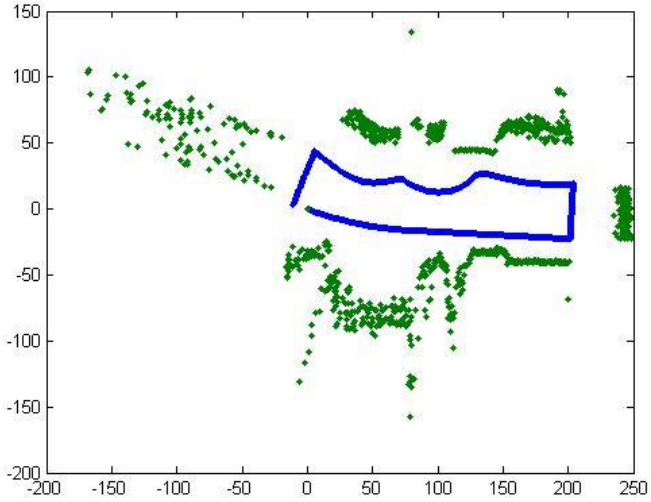


Ilustración 21: Mapeo de una habitación

En esta prueba, tal y como se puede observar, el sensor lateral no recogía datos en los giros debido a que cuando el robot encontraba un obstáculo de frente, éste se paraba, por ese motivo aparecen huecos en blanco en las esquinas.

Así pues, después de estos resultados, se decidió cambiar el algoritmo de mapeo.

Una vez modificado el código, se realizó una pequeña prueba en la que el robot tan solo realizaba el recorrido obteniendo medidas de dos paredes que formaban una esquina. Como se aprecia en la ilustración 22 se solucionó el error, proporcionando mediciones mientras realiza el giro.

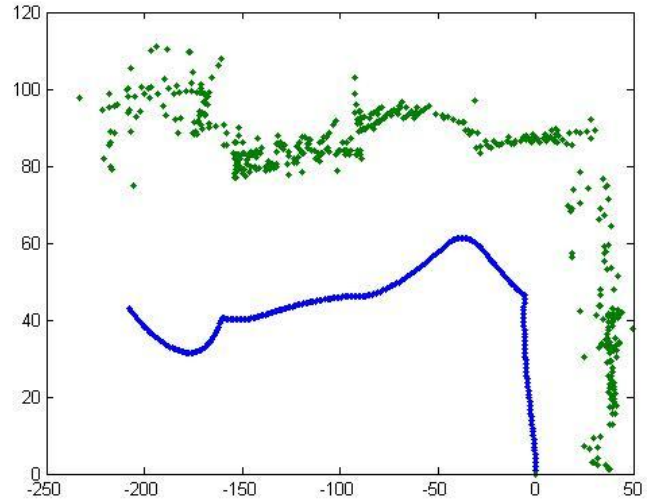


Ilustración 22: Mapeo de una esquina

Visto esto se decidió realizar las últimas pruebas en diversos entornos predeterminados, en los que el robot tuviese un principio y un final, apreciando así el resultado en su totalidad.

Para situar el robot en un entorno más conocido por estudiantes y profesores de la ETSE, se tomo la decisión de introducirse en un aula de la universidad para realizar diversas pruebas. En la ilustración 23 se visualiza el recorrido que realizará el robot alrededor del estrado donde los profesores imparten sus clases.

Ilustración 23: estrado de una clase de la ETSE

En él se han incorporado un par de cajas de cartón para que el recorrido fuese un poco más interesante, haciendo que el robot modifique su trayectoria para no chocar con ellos. En la ilustración 24 se muestra el resultado de haber mapeado el recorrido de la ilustración 23, donde disponemos del recorrido en color azul y su correspondiente mapeo en color verde. En la parte de la esquina, siguiendo el recorrido del robot, se aprecia claramente como el robot se para en un punto al detectar obstáculo y realiza un giro mientras sigue mapeando, a su correspondiente distancia, la forma del hueco entre la puerta y el estrado de la clase. Siguiendo la gráfica en sentido ascendente, realiza un giro hacia la derecha bordeando la esquina del estrado y donde encuentra la

primera caja, la sortea y corrige su trayectoria para volver a conseguir la distancia programada a la que seguir mapeando el entorno. Finalmente, encuentra la segunda caja donde el procedimiento se repite.

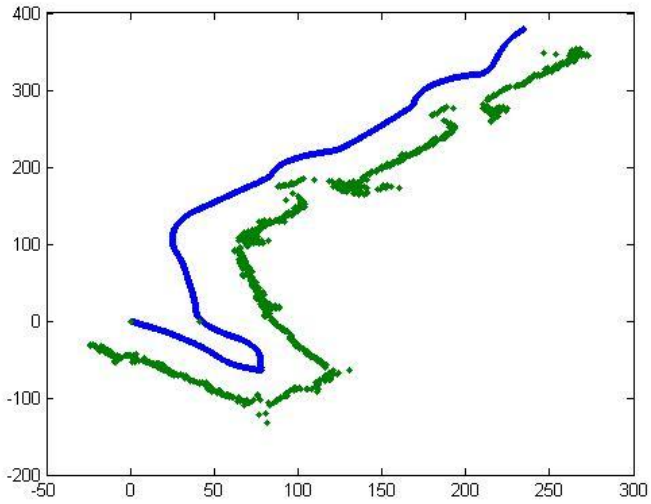


Ilustración 24: Recorrido y mapeo del estrado de una clase

Continuando las pruebas, se decidió improvisar en el mismo escenario, un entorno cerrado (ilustración 25) e intentar conseguir que el robot comenzara y finalizara su recorrido en el mismo punto.

Ilustración 25: Entorno cerrado para mapear

En el resultado mostrado en la ilustración 26, se pueden destacar algunos puntos:

- Las medidas aportadas por los sensores y *encóders* no acaban de estar bien tratadas, puesto que la zona correspondiente al estrado (completamente recta) describe una parábola que no corresponde correctamente con la realidad.
- Se aprecia que el robot comienza en un punto pero no consigue acabar en él, o por lo menos dentro de la misma trayectoria.

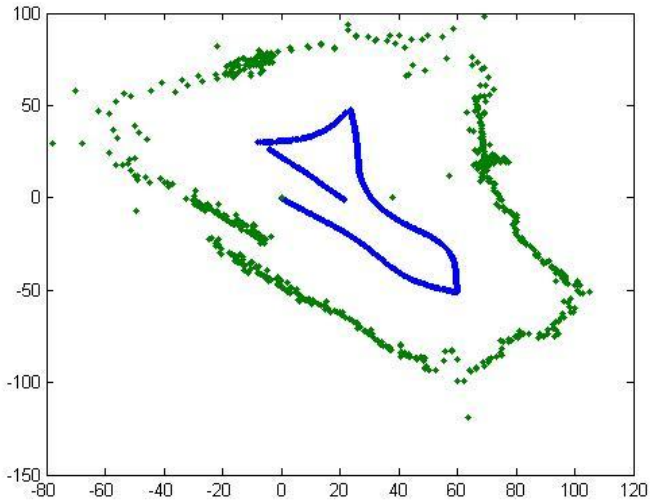


Ilustración 26: Recorrido y mapeo del entorno cerrado

Para intentar mejorar estos aspectos, se realizaron modificaciones en el código interno del microprocesador. En él se debía tener en cuenta que cuando el robot intenta corregir su trayectoria, una vez que se encuentra fuera de la distancia programada, una de las ruedas se acelera más que la otra con el fin de hacer girar al robot.
En los resultados obtenidos, mostrados en la ilustración 27, se obtiene un dibujo en el que se mejora el trazado del recorrido (ósea del robot) y del mapeo, ya no aparece la nombrada parábola, que ha sido sustituida por una recta que corresponde con el obstáculo encontrado.

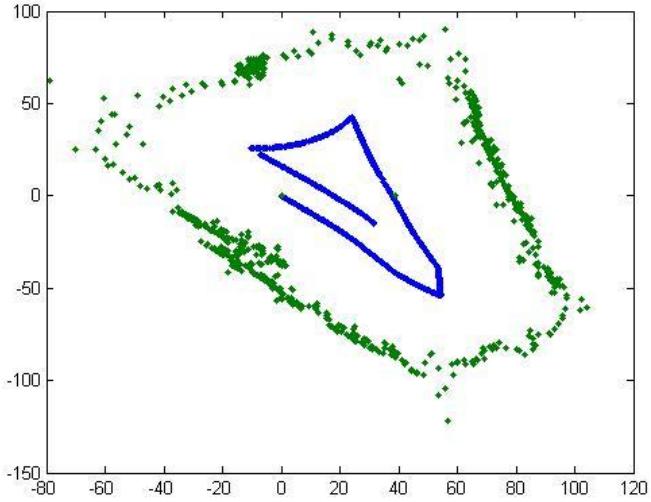


Ilustración 27: Recorrido y mapeo mejorado

De igual manera, las medidas para el mapeo, del principio y final del movimiento se han aproximado casi llegando a coincidir.

Con respecto al recorrido, la parábola que aparece en la parte superior de la figura se debe a que el robot efectivamente ha realizado esa trayectoria. Esto es así ya que los obstáculos encontrados (mochilas de los portátiles) son muy oscuros. Por las características que presenta el sensor IR, le cuesta llegar a detectar los colores oscuros y por lo tanto el robot se fue aproximando a ellos. Otro aspecto del recorrido a tener en cuenta, es que este, al moverse el robot de forma autónoma y entre un rango de valores de distancia al obstáculo de entre 38 y 42 cm, se puede encontrar dentro de este rango y por lo tanto no modifica su trayectoria para acabar exactamente donde empezó.

CAPÍTULO 7. CONCLUSIONES

Tal y como se ha podido observar a lo largo de esta memoria, el reconocimiento y mapeo de una superficie no es algo trivial, ya que se tubo que realizar las correspondientes calibraciones de los componentes y los algoritmos de control de los mismos. Por ese motivo se tuvo que realizar un estudio de todos los dispositivos usados, ya que un mal funcionamiento de uno de ellos provocaría errores en el resultado final.

A la hora de mapear, aun habiendo realizado una correcta calibración y conseguido un óptimo funcionamiento, se observaron errores sistemáticos, producidos por la desalineación de las ruedas y por la incertidumbre de las mismas, es decir, que presenten diferentes diámetros, deformaciones, etc.

Por otro lado también se observaron errores no sistemáticos debidos a que las ruedas patinen, o a que el entorno donde trabaje pueda presentar un suelo desigual.

Al ir introduciendo todos estos errores, se puede concluir que el sistema utilizado dará unos resultados estimados, en los que se deberá tener en cuenta que a mayores distancias recorridas, los errores se irán acumulando y produciendo un peor resultado final.

Teniendo todo esto en cuenta, es gratificante comprobar que los resultados obtenidos han cumplido con creces los objetivos planteados. Se ha conseguido realizar un robot completamente autónomo y con las características necesarias para realizar un correcto mapeo y que este sea identificable con el entorno en el que se encontraba.

Por otro lado se debería comentar algunos objetivos que se han conseguido sin previo planteamiento. Entre ellos destacar la sincronización de trabajo formando un equipo, en este caso reducido, del personal involucrado, y por otro lado aumentar la capacidad de investigación encontrando solución a problemas que se han ido planteando y aprendiendo de ellos.

BIBLIOGRAFÍA

[1] Memoria Mapeo con robot móvil: Caracterización y modelado

Autor: Jorge Martín Pérez

[2] Datasheet ATMega128

http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[3] Datasheet brújula digital CMPS03

http://superrobotica.com/S320160.htm

[4] Datasheet sensor de infrarrojos GP2D12

http://www.superrobotica.com/download/sharp/gp2d12.pdf

[5] Datasheet sensor de ultrasonidos MAXSONAR - EZ1

http://www.msebilbao.com/notas/downloads/Manual%20del%20MaxSonar-EZ1.pdf

[6] Datasheet puente en H L293D

http://www.datasheetcatalog.org/datasheet/stmicroelectronics/1330.pdf

[7] Datasheet encoders ENC01A

http://www.pololu.com/catalog/product/1217

ANEXO I

ALGORITMO DE CONTROL DEL ROBOT EN C PARA MAPEAR

```
#include <avr\io.h>
#include <avr\interrupt.h>
#include <avr\iom128.h>
#include <util\twi.h>

/*****************************************************************************************
****************************** INTERRUPCIONES ***********************************
*****************************************************************************************/

static void interrupciones(void)
{
	DDRE |= 0b00000000;
	EICRB = (1<<ISC40)|(1<<ISC41)|(1<<ISC51)|(1<<ISC60)|(1<<ISC61)|(1<<ISC71);
	EIMSK = (1<<INT4)|(1<<INT5)|(1<<INT6)|(1<<INT7);  //habilitamos interrupciones externas
}

/*****************************************************************************************
************************** BRUJULA DIGITAL CMPS03 ***********************************
*****************************************************************************************/

const uint8_t TWI_SLA_CMPS03 = 0xC0;
volatile uint8_t cmps03, error;

static void iniciTwi(void)
{
	TWBR = 72;
}

ISR(TWI_vect) //interrupcion generada por la brujula
{
	volatile uint8_t str;
	str = TWSR;
	switch(str)
	{
		case TW_START:
				TWDR = TWI_SLA_CMPS03 & 0xFE;
				TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWIE);
				break;
		case TW_MT_SLA_ACK:
				TWDR = 0x01;
				TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWIE);
				break;
		case TW_MT_DATA_ACK:
				TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWIE);
				break;
		case TW_REP_START:
				TWDR = TWI_SLA_CMPS03|0x01;
				TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWIE);
				break;
		case TW_MR_SLA_ACK:
				TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWIE);
				break;
```

```
            case TW_MR_DATA_NACK:
                        cmps03 = TWDR;
                        TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN)|(1<<TWIE);
                        break;
            default:
                        TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN)|(1<<TWIE);
      }
      error = str;
}

/*****************************************************************************
******************************** SENSORES GP2D12 ***********************************
*****************************************************************************/

volatile uint8_t sensor = 0;              //variable donde se guarda el valor del sensor
volatile uint8_t sensorUSADO = 0;  //variable para saber que sensor se ha usado e ir intercambiandolos
volatile uint8_t IRD = 0;                 //variable usada para guardar el valor del infrarrojo delantero
volatile uint8_t IRL = 0;                 //variable usada para guardar el valor del infrarrojo lateral
volatile uint8_t jj = 0;                  //variable usada para recordar que hay un obstaculo cercano
volatile uint8_t ji = 0;                  //variable para recordar que se ha dejado de detectar un obstaculo

static void conversor(void)            //configuracion del conversor
{
          DDRF = 0x00;
      ADMUX = (0<<REFS0)|(1<<ADLAR)|(0<<MUX0);
      ADCSRA =
(0<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)|(1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADFR);
}

static void timer_3(void)   //Timer3 utilizado para el sensor
{
      OCR3A = 125;        // valor a comparar con output compare A equivalente a 2ms
      TCCR3A = (1<<COM3A1)|(1<<COM3B1);  // configuramos en FAST MODE
      TCCR3B = (1<<CS32);         // preescaler a 256
      ETIMSK = (1<<OCIE3A);     //activamos output compare A
}

ISR(TIMER3_COMPA_vect)         //interrupcion generada por el timer3 para alternar sensores
{
      TCNT3 = 0;
      if (sensorUSADO==0)           //sensor lateral
      {
            IRL = sensor;
            ADMUX = (0<<REFS0)|(1<<ADLAR)|(1<<MUX0); //activamos sensor IR delantero
            ADCSRA = (1<<ADPS1)|(1<<ADPS0)|(1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADFR);
            sensorUSADO = 1;      //recordamos que hemos usado el sensor lateral
      }
      else if(sensorUSADO==1) //sensor delantero
      {
            IRD = sensor;
            ADMUX = (0<<REFS0)|(1<<ADLAR)|(0<<MUX0);  //activamos sensor IR lateral
            ADCSRA = (1<<ADPS1)|(1<<ADPS0)|(1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADFR);
            sensorUSADO = 0;       //recordamos que hemos usado el sensor delantero
            if(IRD>38)                //miramos si el sensor delantero detecta un obstaculo cercano
            {
                  jj++;
                  ji=0;
            }
            else if(IRD<22) //miramos si el sensor delantero a dejado de detectar un obstaculo cercano
```

```
		{
			ji++;
			jj=0;
		}
		else
		{
			jj=0;
			ji=0;
		}
	}
}

ISR(ADC_vect)	//interrupcion generada al finalizar la conversion
{
	sensor = ADCH;
}

/*****************************************************************************************
***************************** ENCODERS ***************************************************
*****************************************************************************************/

volatile uint16_t encI =0;		//valor encoder izquierdo
volatile uint16_t encD =0;		//valor encoder derecho

ISR (INT4_vect)	//interrucpion de flanco de subida del encoder izquierdo
{
	encI++;
}

ISR (INT5_vect)	//interrucpion de flanco de bajada del encoder izquierdo
{
	encI++;
}

ISR (INT6_vect)	//interrucpion de flanco de subida del encoder derecho
{
	encD++;
}

ISR (INT7_vect)	//interrucpion de flanco de bajada del encoder derecho
{
	encD++;
}

/*****************************************************************************************
********************************** LEDS ***************************************************
*****************************************************************************************/

volatile uint8_t funcion = 0;		//variable usada para saber la funcion de los leds
volatile uint8_t led = 4;		//variable usada para saber el estado de los leds
volatile uint8_t kit = 0;		//variable usada para realizar que los leds imiten a kit
volatile uint8_t parpadeo = 0;		//variable usada para parpadear los leds
volatile uint8_t i = 0;			//variable usada para no realizar el timer en cada interrupción
volatile uint8_t caja = 0;	//variable usada para saber si estoy en una caja

static void timer_2(void)	//timer usado para los leds
{
	TCCR2 = (0<<COM21)|(1<<CS22)|(0<<CS21)|(1<<CS20); //preescaler a 1024
	TIMSK = (0<<TOIE2);
```

```
}

ISR(TIMER2_OVF_vect) //interrupcion generada por el timer2 para alternar leds, se ejecuta cada 16,3ms
{
	if(i==10)	//conseguimos que se ejecute cada 176ms
	{
		caja++;
		if(funcion==0)		//funcion de kit en los leds
		{
			if(kit==0)
			{
				if(led<128)
				{
				led = led << 1;
				PORTA = led;
				}
				else
				{
				led = led >> 1;
				PORTA = led;
				kit = 1;
				}
			}
			else
			{
				if(led>4)
				{
				led = led >> 1;
				PORTA = led;
				}
				else
				{
				led = led << 1;
				PORTA = led;
				kit = 0;
				}
			}
		}
		else if(funcion==1)	//Funcion de parpadeo en los leds
		{
			if(parpadeo==0)
			{
				PORTA = 0x00;
				parpadeo = 1;
			}
			else
			{
				PORTA = 0xFF;
				parpadeo = 0;
			}
		}
		i=0;
	}
	else
	{
		i++;
	}
}
```

```
/*****************************************************************************************
************************************ USART ***********************************************
*****************************************************************************************/

volatile uint8_t recepcion;              //variable donde guardamos la instruccion que enviamos por USART
volatile uint8_t relog = 5;              //variable usada para enviar las variables cada 40ms
volatile uint16_t enc = 0;               //variable usada para enviar el valor recorrido
volatile uint8_t entera = 0;               //variable que contiene el byte mas significante de lo recorrido
volatile uint8_t sobrante = 0;           //variable que contiene el byte menos significante de lo recorrido
volatile uint8_t z = 0;                  //variable encargada de almacenar el tiempo de envio de variables
volatile uint8_t zz = 0;                 //variable encargada de almacenar el tiempo de envio de variables
volatile uint8_t giro = 0;               //variable usada para saber si estoy girando

static void setupUSART(void)        //configuracion del USART
{
      UCSR0B = (1<<RXCIE0) | (1<<RXEN0)|(1<<TXEN0);
      UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);
      UBRR0H = 0;                                        //Comunicació a 38400bauds
      UBRR0L = 25;
}

ISR(USART0_RX_vect)   //interrupcion generada por el puerto de comunicaciones USART
{
      recepcion = UDR0;   //variable donde gurdamos la instruccion recibida por USART
}

static void timer_0(void)   //timer usado para enviar las variables
{
      TCCR0 = (1<<COM01)|(1<<CS02)|(1<<CS01)|(1<<CS00);
      OCR0 = 125;                  // valor a comparar con output compare equivalente a 8ms
      TIMSK = (0<<OCIE0);          //desactivamos output compare
}

ISR(TIMER0_COMP_vect)       //interrupción del timer0 para enviar variables, se genera cada 8ms
{
      TCNT0 = 0;
      if(giro==0)    //miramos si el robot esta girando
      {
            zz=0;
            if(z==relog)    //enviamos variables cada 8*(valor de relog)
            {
            TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWIE);
            enc = (encD + encI)/2;
            entera = enc/256;
            sobrante = enc%256;
            while ( !( UCSR0A & (1<<UDRE0)) ){}
            UDR0 = entera;
            while ( !( UCSR0A & (1<<UDRE0)) ){}
            UDR0 = sobrante;
            while ( !( UCSR0A & (1<<UDRE0)) ){}
            UDR0 = cmps03;
            while ( !( UCSR0A & (1<<UDRE0)) ){}
            UDR0 = IRL;
            z=0;
            }
            else
            {
            z++;
            }
```

```
	}
	else
	{
		z=0;
		if(zz==relog)
		{
		TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWIE);
		while ( !( UCSR0A & (1<<UDRE0)) ){}
		UDR0 = 0;
		while ( !( UCSR0A & (1<<UDRE0)) ){}
		UDR0 = 0;
		while ( !( UCSR0A & (1<<UDRE0)) ){}
		UDR0 = cmps03;
		while ( !( UCSR0A & (1<<UDRE0)) ){}
		UDR0 = IRL;
		zz=0;
		}
		else
		{
		zz++;
		}
	}
}

/*****************************************************************************************
****************************** PWM DE LOS MOTORES ******************************
*****************************************************************************************/

volatile uint16_t velI = 0;		//variable donde se guarda el OCR del motor izquierdo
volatile uint16_t velD = 0;		//variable donde se guarda el OCR del motor derecho

static void iniciMOTOR(void)		//Timer1 configurat en FAST MODE
{
	DDRB = (1<<PORTB1)|(1<<PORTB2)|(1<<PORTB3)|(1<<PORTB5)|(1<<PORTB6);
	PORTB &= (0<<PORTB1);	//desactivamos el ENABLE
	OCR1A = 674;			// output compare A motor izquierdo
	OCR1B = 684;			// output compare B motor derecho
	velI = OCR1A;			//guardamos la velocidad inicial
	velD = OCR1B;			//guardamos la velocidad inicial
	TCCR1A = (1<<WGM10)|(1<<WGM11)|(1<<COM1A1)|(1<<COM1B1);  // conf. FAST MODE
	TCCR1B = (1<<WGM12)|(1<<CS11);				// PRESCALER A 8
	TIMSK |= (1<<TOIE1)|(1<<OCIE1A)|(1<<OCIE1B);		//habilitamos interrupciones
}

ISR(TIMER1_OVF_vect)		// VECTOR INTERRUPCION OVF
{

}

ISR(TIMER1_COMPA_vect)		// VECTOR INTERRUPCION OUTPUT COMPARE A
{

}

ISR(TIMER1_COMPB_vect)		// VECTOR INTERRUPCION OUTPUT COMPARE B
{

}
```

```
/*****************************************************************************************
************************** MOVIMIENTOS DEL ROBOT ********************************
*****************************************************************************************/

volatile uint8_t sentido = 0;              //variable para saber en que sentido de la marcha iba

void adelante(void)
{
      if (sentido==1)          //venimos de ir marcha atras
      {
            OCR1A = 1024 - velI;              //invertimos velocidad
            OCR1B = 1024 - velD;              //invertimos velocidad
            velI = OCR1A;
            velD = OCR1B;
            sentido = 0;                //recordamos que ibamos hacia delante
      }
      else
      {
            OCR1A = velI;                     //cargamos velocidad inicial
            OCR1B = velD;                     //cargamos velocidad inicial
      }
      PORTB = (1<<PORTB1)|(1<<PORTB2)|(1<<PORTB3); //activamos los motores hacia delante
}

void atras(void)
{
      if (sentido==0)         //venimos de ir hacia delante
      {
            OCR1A = 1024 - velI;              //invertimos velocidad
            OCR1B = 1024 - velD;              //invertimos velocidad
            velI = OCR1A;
            velD = OCR1B;
            sentido = 1;                //recordamos que ibamos hacia atras
      }
      else
      {
            OCR1A = velI;                     //cargamos velocidad inicial
            OCR1B = velD;                     //cargamos velocidad inicial
      }
      PORTB = (1<<PORTB1)|(0<<PORTB2)|(0<<PORTB3); //activamos los motores hacia atras
}

static void izquierda(void)
{
      OCR1A = 274;
      OCR1B = 750;
      PORTB = (1<<PORTB1)|(1<<PORTB2)|(0<<PORTB3); //activamos los motores hacia la izquierda
}

static void derecha(void)
{
      OCR1A = 750;
      OCR1B = 274;
      PORTB = (1<<PORTB1)|(0<<PORTB2)|(1<<PORTB3); //activamos los motores hacia la derecha
}

static void derecha_L(void)          //giro largo para realizar esquinas hacia dentro
{
      OCR1A = 250;
```

```
	OCR1B = 780;
	PORTB = (1<<PORTB1)|(1<<PORTB2)|(1<<PORTB3); //activamos los motores hacia delante
}

static void parar(void)
{
	PORTB = (0<<PORTB1);	//Desactivamos los motores
}

/*****************************************************************************************
****************************** ALGORITMO DE MAPEO ***********************************
*****************************************************************************************/

void comenzar(void)		//instruccion de comienzo
{
	adelante();
	funcion = 1;
	TIMSK = (1<<TOIE2);
	while((recepcion!='p')&&(jj<60))
	{}
	funcion = 0;
	while(recepcion!='p')
	{
		izquierda();
		caja = 0;
		while((recepcion!='p')&&(ji<60)&&(caja<20))
		{}
		parar();
		encD = 0;
		encI = 0;
		giro = 0;
		TIMSK = (1<<OCIE0)|(1<<TOIE2);
		adelante();
		while((recepcion!='p')&&(jj<60))
		{
			if(IRL<50)
			{
				derecha_L();
				while((IRL<65)&&(recepcion!='p')&&(jj<60))
				{
				}
				adelante();
			}
			if(IRL<73)	//el robot se aleja
			{
				if(OCR1A>501)
				{
				 OCR1A--;	//Aceleramos motor izquierdo

				}
				else if(OCR1B<700)
				{
				 OCR1B++;	//Frenamos motor derecho
				}
			}
			else if(IRL>79)		//el robot se acerca
			{
				if(OCR1A<780)
				{
```

```
                        OCR1A++;          //Frenamos motor izquierdo

                        }
                        else if(OCR1B>501)
                        {
                        OCR1B--;          //Aceleramos motor derecho
                        }
                }
                else
                {
                        OCR1A = velI;
                        OCR1B = velD;
                }
        }
        parar();
        giro = 1;
    }
    TIMSK = (0<<OCIE0)|(0<<TOIE2);
    PORTA = 0x00;
}

/*********************************************************************************************
****************************** PROGRAMA PRINCIPAL ******************************
*********************************************************************************************/

int main(void)
{
    DDRA |= 0b11111100;
    DDRE = 0x02;
    sei();              //habilitamos interrupciones
    conversor();
    iniciMOTOR();
    timer_0();
    interrupciones();
    iniciTwi();
    timer_3();
    timer_2();
    setupUSART();
    while(1)
    {
        switch(recepcion)
        {
            case 'c':
                relog = 5;          //mandamos variables cada 5*8ms = 40ms
                comenzar();
                parar();
                recepción = 0;
                break;
            case ‘w’:
                adelante();
                recepción = 0;
                break;
            case ‘a’:
                izquierda();
                recepción = 0;
                break;
            case ‘d’:
                derecha();
                recepcion = 0;
```

```
                    break;
                case 's':
                    atras();
                    recepcion = 0;
                    break;
                case 'p':
                    parar();
                    TIMSK = (0<<OCIE0)|(0<<TOIE2);
                    PORTA = 0x00;
                    recepción = 0;
                    break;
            }
        }
}
```

ANEXO II

ALGORITMO DE COMUNICACIÓN EN PYTHON

```
import sys
import serial
import threading
import msvcrt
import math

salir = '\x71'          #Tecla q
fin = 0
t = 0

def reader():

  #-- Cuando fin=1 se termina el bucle
   while not(fin):

      data = s.read()        #recibimos valor mas significativo de los encoders y guardamos en data
      try:
         d = ord(data)
         a = d * 256

         data = s.read()              #recibimos valor menos significativo de los encoders
         d = ord(data)
         b = a + d
         t = (b * 42 * 3.141592653589793 / 24) - (b * 0.2586)          #pasamos a mm y corregimos error

         data = s.read()          #recibimos valor de la brujula
         ff = ord(data)
         dd = ff * (360.0/256)          #pasamos valor digital a grados y corregimos el error
         if ((dd <= 360)&(dd > 245)):
            f = dd - 10
         elif ((dd < 246)&(dd > 200)):
            f = dd
         elif ((dd < 201)&(dd > 170)):
            f = dd + 10
```

```
        elif ((dd < 171)&(dd > 80)):
            f = dd + 17
        elif ((dd < 81)&(dd > 55)):
            f = dd + 10
        elif ((dd < 56)&(dd > 35)):
            f = dd
        elif ((dd < 36)&(dd >=10)):
            f = dd - 10
        elif ((dd < 10)&(dd >=0)):
            f = dd + 350
        else:
            pass

        data = s.read()          #recibimos valor de sensor
        j = ord(data)
        g = j/100.0
        y = 16.75*g**4 - 119.26*g**3 + 311.7*g**2 - 365.71*g + 184.03 #pasamos el valor digital a mm

        fichero=open("mapeo.txt","a")          #guardamos variables en el fichero mapeo.txt
        fichero.write("%d\t"%t)
        fichero.write("%d\t"%f)
        fichero.write("%d\n"%y)
        fichero.close()
    except:
        continue

def writer():
  while 1:
    try:
        c = raw_input('\n')          #-- Esperar a que se pulse una tecla

    #-- Si es la tecla de fin se termina
        if c == salir:   #variable de salida definida arriba
            break
        else:
            s.write(c)          #-- Enviar tecla por el puerto serie



    except:         #-- Si se ha pulsado la tecla q terminar
```

```
        print "Abortando..."
        break

print "\t--- Miniterm --- pulse tecla q e INTRO para salir\n"
print "\t\tIntroduce el puerto incluyendo COM\n"
puerto = raw_input('\n')
s = serial.Serial()
s.port = puerto

s.baudrate = 38400
s.timeout = 1
s.open()

print "\n\tComunicacion establecida a una velocidad de 38400 baudios\n"
print "\nPulse q para salir, c para comenzar a mapear y p para detener el robot"
r = threading.Thread(target = reader)
r.start()
writer()
fin = 1
r.join()
print "\n"
print "--- Fin ---"
s.close()
```

Resum:

El projecte tracta de la fabricació d'un robot mòbil que sigui capaç de realitzar el mapeig d'una superfície, evitant els obstacles que es pugui trobar en el transcurs del seu recorregut. És un projecte complex, per aquest motiu la part de procés de dades s'ha fet en un projecte posterior. Aquesta memòria tracta del muntatge i calibració dels components, a més de la realització dels algorismes de control dels mateixos, per tal de realitzar el mapeig de la superfície, aconseguint així l'objectiu plantejat.

Resumen:

El proyecto trata de la fabricación de un robot móvil que sea capaz de realizar el mapeo de una superficie, evitando los obstáculos que se pueda encontrar en el transcurso de su recorrido. Es un proyecto complejo, por ese motivo la parte de procesamiento de los datos se ha realizado en un proyecto posterior. Esta memoria trata del montaje y calibración de los componentes, además de la realización de los algoritmos de control de los mismos, por tal de realizar el mapeo de la superficie, consiguiendo así el objetivo planteado.

Summary:

The project deals with the manufacture of a mobile robot that is able to plot a map its environment, avoiding obstacles you may encounter during its journey. Is a complex project, which is why the data processing was carried out in a subsequent project. This report explains the assembly and calibration of components, besides the realization of the control algorithms thereof, by such of performing the surface mapping, thus achieving the stated objective.