UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

PLATAFORMA GENÉRICA PARA DESARROLLO CON ROBOTS MÓVILES

Ismail Rebah Bouaiachi

Diciembre 2009

PLATAFORMA GENÉRICA PARA DESARROLLO CON ROBOTS MÓVILES

AUTOR: Ismail Rebah Bouaiachi
TUTOR: Guillermo González de Rivera Peces

Grupo HCTlab
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Diciembre de 2009

Palabras clave:

Robot, plataforma móvil, diseño modular, sensores autónomos, cooperatividad, I^2C.

Resumen:

El objetivo de este PFC es diseñar y construir una serie de módulos, junto con su API correspondiente, que permitan componer un robot de manera sencilla y rápida para su aplicación a diferentes tareas.

Cada uno de los robots debe ser construido de manera que sea modular, esto es, que sea posible la sustitución, eliminación o incorporación de diferentes sensores y la lógica inherente a ellos de manera que no se deba realizar modificación alguna al resto de sensores que componen el robot, ni a la unidad central.

Gracias a dicha modularidad construir un robot consistirá en ensamblar un conjunto de sensores con sus controladores correspondientes de manera que la plataforma construida estará hecha a medida para realizar la tarea que se le encomiende, consiguiendo ventajas tales como el ahorro de recursos y una significativa reducción tanto del tiempo de diseño como de la complejidad del sistema.

Abstract:

The aim of this project is to design and build a series of modules, in addition to their corresponding API, with which to make a robot easily and fast in order to use it in different tasks.

Each of the robots must be built so that they are modular, that is, we would can to replace, delete or add various sensors and logic inherent in them without to make any modification to the rest of sensors of the robot, or the central unit.

Thanks to this modularity to build a robot will be to assemble a group of sensors with their corresponding drivers so that the platform will be built tailor-made to perform the task entrusted to it, getting benefits such as saving resources and a significant reduction in both design time as the complexity of the system.

Agradecimientos

En primer lugar quiero agradecer a mi tutor, Guillermo González de Rivera Peces, la oportunidad de realizar el presente PFC. Quisiera expresarle también mi gratitud por toda la ayuda que me ha prestado y sin la cual no habría logrado finalizar lo que hoy es una realidad. Gracias por las explicaciones de todo tipo de temas, por la información, por los abundantes recursos materiales y principalmente por la comprensión de mi avance.

Recuerdo con cariño muchos momentos vividos en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid. Desde el primer año se convirtió en mi residencia particular y allí he pasado, en los últimos cinco años, más tiempo que en mi propia casa.

Este último año he pasado grandes momentos con mis compañeros del HCTlab, interminables jornadas de trabajo y risas. Jamás olvidare la ayuda que me prestó Alberto Sánchez González; desde el primer momento hasta el último me asesoró, ayudó y explicó todo cuanto necesité por ridículo que pudiera parecer. En cuanto a Ricardo Ribalda Delgado, muchas gracias por ahorrarme miles de horas de trabajo imposible con TU analizador lógico y por solucionar algunos temas de programación.

Mi etapa universitaria ha sido muy agradable, en ella he conocido a personas generosas, amables, simpáticas, y me llevo, además de amigos, muchos momentos irrepetibles. Muchos de los momentos que he vivido tanto en la universidad como fuera de ella se deben a María Davó Siguero, a la que no podía dejar de mencionar. Gracias por ser mi secretaria personal, por aguantarme en los momentos más difíciles, por estar conmigo en los momentos felices, por involucrarte en mi proyecto y por ayudarme en innumerables ocasiones tanto personales como profesionales, gracias por ofrecerme tu cariño, tu amistad y tu portátil :P, y en definitiva, gracias por ser quien eres…

Finalmente una mención muy especial merece mi familia y concretamente mis padres, porque sin ellos, mis estudios universitarios no habrían sido más que un sueño. Gracias por estar ahí, por ser mi apoyo, por comprenderme, por tratar de ayudarme, gracias por haberme educado del modo en que lo habéis hecho. Tengo tanto que agradeceros que jamás podré terminar de hacerlo. A vosotros os dedico este proyecto de fin de carrera.

Ismail Rebah Bouaiachi
5 de diciembre de 2009

Índice de contenidos

Índice de figuras

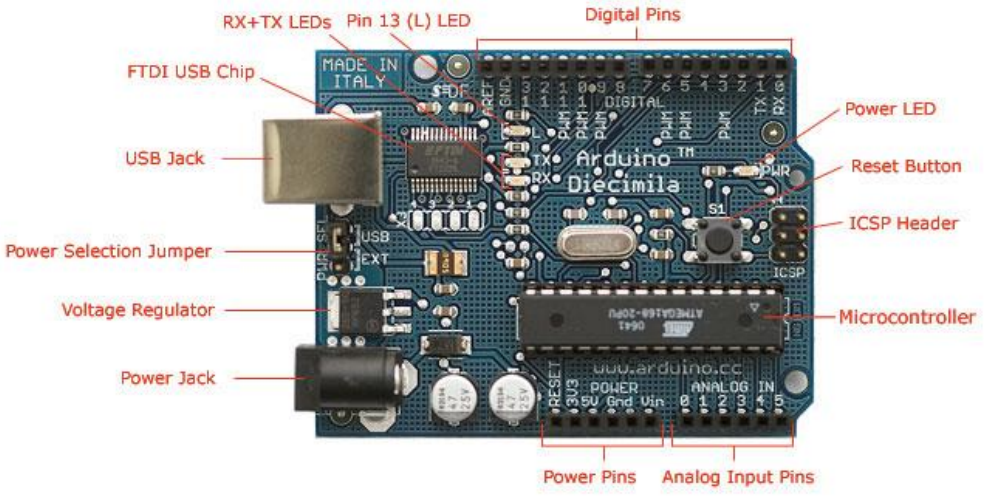
Índice de tablas

1 Introducción

1.1 Motivación

En la actualidad, el diseño y la construcción de un robot, móvil o no, está muy enfocada a la aplicación concreta a la que se desea destinar el mismo. El problema de este tipo de actuación es que si la aplicación a la que se destinó originalmente el robot cambia sus características o bien se desea destinar a otra diferente, es necesario hacer cambios significativos a la plataforma (robot) que implican no sólo un coste económico sino también posiblemente una disminución de la eficiencia en la tarea que se pretende realizar.

La solución al problema anterior consiste en el uso de plataformas genéricas como la de la Figura 1-1. Así, la misma plataforma, puede destinarse a una u otra tarea.



Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.

Figura 1-1: Plataforma robótica de propósito general Arduino Diecimila

Pero esta solución tiene a su vez otro problema, al ser genérica, la plataforma está dotada de una serie de funcionalidades que posiblemente no sean necesarias en algunas aplicaciones particulares y que repercutirán en el tamaño y la eficiencia del robot. Además, teniendo en cuenta que el robot trata de ser genérico es posible que disponga de una gran cantidad de elementos que no son necesarios para la aplicación a la que se desea destinar el mismo, lo que conlleva un consumo energético mayor que el óptimo puesto que existen elementos que están consumiendo energía pero que sin embargo no son necesarios.

Tratando de solucionar ambos problemas a la vez, es decir, con la motivación de crear una plataforma capaz de destinarse a tantas aplicaciones como el usuario sea capaz de plantear y que a la vez sea compacta y óptima para la aplicación a la que se quiere destinar nace este Proyecto de Fin de Carrera (PFC).

Así, este PFC pretende crear los elementos necesarios para construir una plataforma genérica modular para desarrollo con robots móviles, de manera que construir un robot, consista en ensamblar una serie de módulos que, a pesar de operar de manera independiente, sean capaces de gestionarse fácilmente mediante un módulo maestro o principal a través de algún tipo de protocolo de comunicaciones.

De este modo se tiene una plataforma genérica capaz de usarse en diferentes aplicaciones y a la vez, gracias a su modularidad, compacta y totalmente orientada a la aplicación a la que se destinará.

Construir una plataforma robótica e implementar el software necesario para que ésta realice alguna tarea incluso de baja o mediana complejidad es algo difícil y que no está al alcance de cualquier usuario; sino que es necesario tener algunos conocimientos no solamente de programación sino también de electrónica y en algunos casos de protocolos de comunicación para transmitir información si fuera necesario.

Así la motivación de este PFC se convierte no solamente en crear una plataforma genérica y modular sino también en crear todo el software necesario para su uso mediante una API[1] de manera que un usuario sin experiencia ni conocimientos previos pueda usar la plataforma para resolver cualquier problema que se plantee y sea capaz de adaptarla gracias a la modularidad de sus componentes.

Gracias a la API cualquier persona podría simplemente poner "avanzar plataforma 7,3m. hacia delante" y no necesitaría conocimientos acerca de motores, ni de señales PWM ni de interrupciones, etc.

Esto supone un gran avance y es que de este modo se acerca al usuario sin conocimientos un mundo hasta ahora desconocido; el apasionante mundo de la robótica…

[1] Una interfaz de programación de aplicaciones o API es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio.

1.2 Objetivos

El objetivo de este PFC es diseñar y construir una serie de módulos, junto con su API correspondiente, que permitan componer un robot de manera sencilla y rápida para su aplicación a diferentes tareas.

Al tener diferentes módulos, cada uno dedicado en exclusividad a una tarea, se consigue que cada uno de ellos sea lo más eficaz posible en la función que realiza y por tanto que la unión de todos sea también la más eficiente.

Por ejemplo, en caso de hacer que la plataforma avance una distancia concreta sería necesario que la aplicación principal estuviera gran parte del tiempo monitorizando el avance de los motores para determinar cuándo se ha recorrido la distancia requerida. No obstante, si existiera un módulo dedicado a esa tarea se conseguiría liberar a la aplicación principal de una gran carga y podría dedicarse a la gestión de otras tareas consiguiendo un aumento tanto de la eficiencia como de la efectividad.

Otra de las cosas interesantes de la modularidad es que permite tener un robot dotado de, únicamente, los componentes necesarios para desempeñar la tarea que requiere. De este modo, se consigue un ahorro notable.

Para mostrar de manera más concreta el ahorro generado gracias a la modularidad se propone al lector imaginar el siguiente escenario.

Imagine que el robot no fuera modular y que únicamente se necesitara que realizase una serie de medidas de distancia en una habitación y se situara aproximadamente en el punto medio de ésta.

Sería posible llevar a cabo dicha tarea con el uso de dos motores y de uno a cuatro sensores de ultrasonidos dependiendo de la velocidad a la que se desee llevar a cabo.

Si se decidiera usar una plataforma genérica (dotada de botones para interactuar con el usuario, pantallas LCD para mostrar información, cámaras de video para captar imagen, sensores de contacto, puertos de comunicaciones, etc.) para esta aplicación habría una gran cantidad de hardware no necesario que se habría tenido que pagar pero no se usaría. Sin embargo, si el robot fuera modular, bastaría con ensamblar un módulo para generar el movimiento y un módulo para realizar medidas de ultrasonidos.

Queda patente pues, el **ahorro económico** que supondría el uso de una plataforma modular frente al uso de una plataforma no modular.

Obsérvese del mismo modo que con el uso de una plataforma modular se conseguiría un robot del **tamaño mínimo** necesario mientras que con la plataforma no modular el *hardware extra* provocaría que el robot fuera de mayor tamaño.

Otra de las ventajas de una plataforma modular frente a una que no lo es reside en el hecho de la eficiencia y la ocupación software. En una plataforma no modular, aunque no se estén usando una serie de funcionalidades disponibles, sí que está programado el firmware necesario para su uso, ocupando memoria del programa y haciendo que sea necesaria una mayor capacidad software.

En una plataforma modular, el código dedicado a cada módulo se encuentra separado de la aplicación principal. Esto quiere decir que el código necesario para mover los motores se halla en el módulo dedicado a esa tarea; el código dedicado a realizar medidas de distancia se halla en el módulo dedicado a esa tarea; etc. Así, en conjunto, el **código que se halla en la plataforma es el óptimo** para la tarea que se desempeña. No habiendo código innecesario ocupando espacio.

Finalmente cabe resaltar un aspecto de gran importancia, el consumo energético. En una plataforma no modular, el hardware que no se usa también consume energía ya que está físicamente conectado al sistema. No obstante, en una plataforma modular, puesto que sólo están conectados al sistema los componentes necesarios **el consumo de energía es el mínimo posible** para la realización de la tarea.

Una vez justificada la razón de crear una plataforma genérica modular es necesario, en primer lugar, determinar qué módulos serán los que se diseñen y construyan. Se ha considerado que lo mínimo necesario para una plataforma genérica es que sea capaz de trasladarse de un lugar a otro o de girar un ángulo determinado, que sea capaz de realizar medidas de distancia mediante diferentes sensores ya que es posible que no sea viable usar algunos tipos de sensores en ciertos entornos (por ejemplo infrarrojos en lugares con reflejos de luz solar) y que sea capaz de mostrar los datos bien por una pantalla auxiliar o bien por conexión con un ordenador. De este modo surgen tres módulos básicos:

Módulo de locomoción

- Surge para resolver el problema de la movilidad. Se trata de un módulo cuyo objetivo fundamental es dotar a la plataforma de la capacidad de trasladarse de un lugar a otro recorriendo para ello distancias concretas en linea recta, haciendo giros de cualquier ángulo y combinado ambas tareas.

Módulo US&IR

- Surge para resolver la problemática de la realización de medidas de distancias. Se trata de un módulo cuyo objetivo fundamental es calcular la distancia que hay entre el módulo y el objeto más cercano. Es deseable que tenga la capacidad de realizar medidas mediante el uso de varios fenómenos físicos por si alguno de ellos fuera inviable en el entorno en el que se desarrolla la tarea.

Módulo de comunicaciones

- Surge para resolver la problemática de la comunicación de la plataforma con el mundo exterior. Se trata de un módulo cuyo objetivo fundamental es que el robot sea capaz de interactuar bien con el usuario o bien con otras máquinas, tanto enviando información como recibiéndola. Es deseable que no sólo sea capaz de mostrar la información mediante su envío a un P.C. sino que sea posible la integración de una pantalla para situaciones en las que no sea posible tener un ordenador conectado físicamente a la plataforma.

Pero, no sólo basta con determinar qué módulos básicos han de crearse sino que también es necesario reflexionar acerca del modo en que se interconectará toda la plataforma para trabajar de manera conjunta y organizada con un mismo fin.

Es importante escoger un protocolo de comunicaciones que permita a los módulos interactuar con la aplicación principal (situada en un módulo maestro) de manera sencilla y que no requiera ni demasiadas conexiones ni demasiado hardware externo para que sea rápido y simple conectar y desconectar los módulos del sistema principal.

En este caso se ha escogido que el método de interconexión de los diferentes módulos con la aplicación principal sea mediante el protocolo de comunicaciones I^2C. Los detalles del protocolo se muestran en sucesivos capítulos, pero en líneas generales, la elección se basa en:

- ✓ La simplicidad del protocolo y la gran cantidad de circuitos que pueden interconectarse a un mismo bus de comunicaciones.
- ✓ Que únicamente son necesarios dos hilos para interconectar todo el sistema.
- ✓ El muy escaso número de elementos hardware necesarios para llevar a cabo la implementación del protocolo.
- ✓ El hecho de que sea un protocolo muy extendido en el mundo de la robótica. Tanto que existen sensores que se manejan directamente a través de este protocolo y que por tanto podrían interconectarse directamente al sistema.

Finalmente, se muestran de manera más concreta los objetivos que se pretenden alcanzar con este proyecto.

- Diseño y construcción física de un módulo capaz de controlar motores de corriente continua (Módulo de Locomoción).
- Implementación de la API para controlar el módulo de locomoción a través del protocolo de comunicaciones I^2C.
- Diseño y construcción física de un módulo capaz de medir la distancia que hay entre el propio módulo y el objeto más cercano a él (Módulo US&IR).
    - Será capaz de manejar sensores de ultrasonidos y sensores de infrarrojos.
- Implementación de la API para controlar el módulo US&IR a través del protocolo de comunicaciones I^2C.
- Diseño y construcción física de dos variantes de un mismo módulo capaz de intercambiar información con el usuario o con otras máquinas como por ejemplo ordenadores (Módulo de Comunicaciones).
    - El modelo A será capaz de manejar un puerto serie RS-232 y un puerto USB.
    - El modelo B será capaz de manejar además del puerto serie RS-232 y el puerto USB una pantalla LCD que muestre texto y/o imágenes.
- Implementación de la API para controlar tanto el modelo A como el modelo B del módulo de comunicaciones a través del protocolo de comunicaciones I^2C.

- Implementación de la API para controlar la pantalla LCD del modelo B del módulo de comunicaciones a través del puerto USB o del puerto serie RS-232 tanto en Windows como en Linux.
- Diseño y construcción física de un módulo capaz de gobernar todos los módulos anteriores (Módulo Maestro).
- Implementación de la API para controlar de manera transparente todas las funcionalidades de cualquier módulo por parte de un maestro de ejemplo (que se construirá) a través del protocolo de comunicaciones I^2C.
- Construcción de herramientas físicas capaces de cargar el firmware en todos los módulos y modificar después el firmware del módulo maestro (Programadores).
    - Se construirán dos variantes de programadores. Uno simple que funcione por el puerto paralelo y otro más complejo que pueda cargar el firmware desde un puerto USB.
- Implementación de una aplicación en el maestro de ejemplo para probar la plataforma aplicada a un problema concreto.

En la Figura 1-2 se ilustra el hardware que se espera obtener al finalizar el PFC.

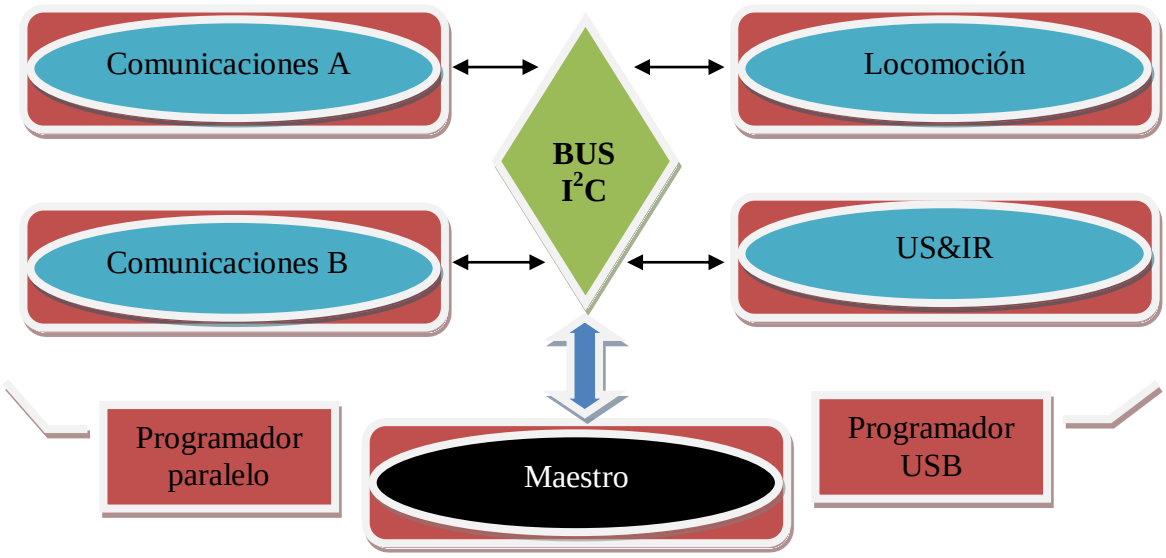


Figura 1-2: Hardware a obtener tras la finalización del PFC

1.3 Organización de la memoria

Esta memoria se divide en ocho capítulos y 6 anexos que se detallan a continuación.

En el ***primer capítulo*** se hace una introducción al proyecto, definiéndose la motivación de éste, así como los objetivos que pretenden alcanzarse con el mismo. El capítulo acaba con una explicación de la forma en la que está estructurado el proyecto y lo que puede encontrarse en cada capítulo.

En el ***segundo capítulo*** se abordan las partes comunes a todos los módulos (locomoción, US&IR y comunicaciones), es decir, aquellas partes que todos los módulos han de tener independientemente de la tarea que desempeñen. Así pues, dado que todos los módulos han de tener como núcleo central un microcontrolador, se mencionan los requisitos y las funcionalidades mínimas que éste debe cumplir. Finalmente se dan unas nociones básicas acerca de los dos protocolos de comunicaciones presentes en todos los módulos, el protocolo SPI y el protocolo I^2C.

El ***tercer capítulo*** está dedicado en su totalidad al módulo de locomoción. En primer lugar se hace una introducción al módulo explicando por qué es necesario y lo que éste es capaz de hacer. Se continúa con unas breves nociones acerca de los motores de corriente continua para, finalmente, desembocar en el motor concreto que se usará en este PFC. Posteriormente se incluye una introducción a los puentes en H como driver de motores y se analiza cada uno de los que se han probado; a continuación, se argumenta el por qué del uso del TLE4208G como driver de motores y puente en H. En otro apartado posterior se puntualiza acerca de dos formas de controlar la velocidad de los motores, en primer lugar, la reducción de la tensión de alimentación y en segundo lugar, la modulación por ancho de pulso, PWM. Después se detallan, esta vez de forma concreta para este módulo, las características que ha de cumplir el microcontrolador. En los dos siguientes puntos de este capítulo se describen de manera detallada tanto el diseño hardware como el diseño software. En el diseño hardware se analiza cada una de las partes físicas que componen el módulo justificando su existencia, mientras que en el diseño software se explica el modo de usar cada una de las funcionalidades que ofrece. En el último punto se muestra una imagen real tanto de la etapa de prototipado como de la etapa final de construcción del módulo detallando las especificaciones técnicas y el presupuesto del módulo en cuestión.

El ***cuarto capítulo*** está dedicado al módulo US&IR, es decir, el módulo de ultrasonidos e infrarrojos. En primer lugar se hace una introducción al módulo explicando por qué es necesario y lo que éste es capaz de hacer. Se continúa con unas breves nociones acerca de los dos tipos concretos de sensores que se usan en este módulo, el sensor de ultrasonidos SRF04 / SRF05 y el sensor de infrarrojos SHARP GP2D12. Posteriormente se detallan, esta vez de forma concreta para este módulo, las características que ha de cumplir el microcontrolador. En los dos siguientes puntos de este capítulo se describen de manera detallada tanto el diseño hardware como el diseño software. En el diseño hardware se analiza cada una de las partes físicas que componen el módulo justificando su existencia, mientras que en el diseño software se explica el modo de usar cada una de las funcionalidades que ofrece. En el último punto se muestra una imagen real tanto de la etapa de prototipado como de la etapa final de construcción del módulo detallando las especificaciones técnicas y el presupuesto del módulo en cuestión.

El ***quinto capítulo*** está dedicado al módulo de comunicaciones. En primer lugar se hace una introducción al módulo explicando por qué es necesario y lo que éste es capaz de hacer. Se continúa con unas breves nociones acerca de las comunicaciones serie RS-232, las comunicaciones USB y la pantalla LCD que se va a usar en este proyecto. Posteriormente se explica el funcionamiento de dos drivers que se usarán en este módulo, el MAX232 y el FT232BL. Después se detallan, esta vez de forma concreta para este módulo, las características que ha de cumplir el microcontrolador tanto para el modelo A como para el modelo B. A continuación se describen de manera detallada tanto el diseño hardware como el diseño software de cada uno de los modelos del módulo. En el diseño hardware se analiza cada una de las partes físicas que componen ambos modelos justificando su existencia, mientras que en el diseño software se explica el modo de usar cada una de las funcionalidades que ofrece tanto el modelo A como el modelo B del módulo de comunicaciones. Para finalizar se muestran imágenes reales de ambos modelos tanto en la etapa de prototipado como en la etapa final de construcción detallando las especificaciones técnicas de ambos y el presupuesto de ambas variantes del módulo.

En el ***sexto capítulo*** tras justificar la necesidad de diseñar y construir el módulo maestro se muestra al lector la manera de acceder a cada una de las funcionalidades de los módulos esclavo gracias a la API implementada para él. Posteriormente se hace una descripción acerca de la aplicación de ejemplo diseñada mostrando en primer lugar una descripción de la misma para luego detallar tanto el hardware como el software involucrado. Para terminar el capítulo se agregan unas notas acerca del sistema final.

En el ***séptimo capítulo*** se muestran los resultados de una serie de pruebas realizadas con los módulos; una prueba de sensores de ultrasonidos, la captura de una trama de I^2C, una prueba de PWM con distinto ciclo de trabajo y finalmente la prueba de la aplicación de ejemplo diseñada.

En el ***octavo capítulo*** se describen las conclusiones obtenidas de este PFC así como algunas líneas de trabajo futuro que podrían seguirse.

En el ***Anexo A*** se muestran algunas gráficas extraídas de la última revisión del protocolo de I^2C acerca del valor de las resistencias de pull-up necesarias y se definen algunas direcciones que no deben usarse en el bus.

En el ***Anexo B*** se muestra una posible alternativa para reducir el número de canales de PWM necesario para controlar la velocidad y el doble sentido de giro de dos motores de corriente continua.

En el ***Anexo C*** se muestra la probabilidad de error en la comunicación serie a diferentes velocidades al hacer uso de una USART.

En el ***Anexo D*** se presentan los resultados obtenidos del sensor GP2D12 y a partir de los cuales se ha calibrado.

El ***Anexo E*** se informa acerca del software que es necesario instalar para modificar el código tanto del maestro como de los módulos esclavo, además se detallan los archivos que componen cada uno de los módulos y se muestra la manera de cargar el firmware.

En el ***Anexo F*** se muestra el proceso seguido para la construcción física de cada uno de los PCB's diseñados.

El ***Anexo G*** contiene el esquemático y los layouts de todos los PCB's construidos durante el desarrollo de este PFC.

2 Partes comunes a todos los módulos

A pesar de las particularidades propias de cada módulo, existen una serie de elementos comunes a todos ellos. Uno de esos elementos es el núcleo central, cuyas funciones, en este PFC, son llevadas a cabo por el microcontrolador.

Naturalmente, y dado que cada uno de los módulos ha de desempeñar una tarea diferente, los microcontroladores de cada uno de ellos serán distintos, no obstante, todos ellos deben cumplir una serie de requisitos generales que se describen unas líneas más adelante, en el apartado 2.1.

Otro de los elementos comunes a todos los módulos es que todos ellos son capaces de comunicarse mediante el protocolo I^2C. Es por esa razón por lo que en el apartado 2.2 se describirá en detalle el protocolo de comunicaciones citado.

Finalmente, el tercero de los elementos comunes es el que se describe en el apartado 2.3. Se trata del protocolo de comunicaciones SPI, que es usado por todos los módulos tanto para cargar el firmware inicial como para posibles modificaciones del mismo.

2.1 Características genéricas de los microcontroladores

Para realizar el presente PFC es necesario el uso de microcontroladores. Cada módulo controlador así como el módulo central están dotados de "inteligencia" gracias a un microcontrolador.

Cada módulo requiere el uso de un microcontrolador elegido cuidadosamente y de manera específica para realizar las operaciones a las que se va a destinar. No obstante, todos los usados en los módulos que se detallan en este PFC deben a su vez cumplir una serie de requisitos genéricos, que se detallarán más adelante.

En el presente PFC se ha decidido optar por los microcontroladores del fabricante ATMEL(2), debido a la gran variedad de microcontroladores que ofrece, la gran cantidad de información disponible en Internet acerca de ellos, porque permite hacer desde aplicaciones simples hasta aplicaciones de gran complejidad y además porque hay bastantes

[2] www.atmel.com

distribuidores de este fabricante, de modo que es fácil conseguir uno de sus microcontroladores.

A continuación, en la Figura 2-1 se muestra una gráfica que ilustra parte de la gama de microcontroladores del fabricante escogido, Atmel. La gráfica ha sido extraída de su página oficial y muestra una relación entre el tamaño y las prestaciones que ofrecen algunas de las series que tiene en el mercado. Las series de microcontroladores que se manejarán en este PFC son las dos intermedias, tinyAVR y megaAVR.

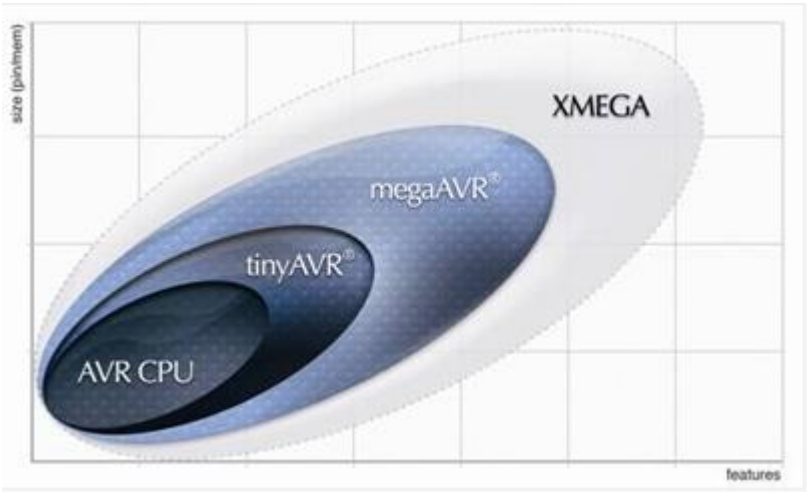


Figura 2-1: Tamaño vs funcionalidad en micros de Atmel (imagen extraída de www.atmel.com)

Finalmente, los microcontroladores escogidos deben cumplir los requisitos siguientes:

- **Económico**

Muchos de los proyectos actuales en el ámbito de la ingeniería como en muchos otros campos no ven la luz por cuestiones económicas.

Por muy eficiente que sea un producto, se pretende que éste sea lo más barato posible, y en la mayoría de las ocasiones el coste está acotado por un máximo; por esta razón, y siguiendo dicha filosofía se busca no sólo un micro barato, sino que cada uno de los componentes del robot sea lo más barato posible sin que esto repercuta en las prestaciones que ofrece.

- **Comunicación I^2C**

Se necesitan también algunos pines que permitan establecer la comunicación I^2C. La mayoría de los micros de un tamaño medio y alto están dotados de dichos pines, pero en

muchos casos, los micros de tamaño más reducido tienen algunos protocolos que son compatibles con I^2C, aunque no son propiamente éste, como por ejemplo el protocolo USI ó el protocolo TWI. Esto puede comprobarse en la Tabla 2-1 (El resto de microcontroladores de mayor gama tienen I^2C).

Microcontrolador	I/O[3]	TWI (I^2C)
ATtiny12	6	NO
ATtiny13	6	NO
ATtiny2313	18	Emulado por USI
ATtiny24/44/84	12	Emulado por USI
ATtiny25/45/85	6	Emulado por USI
ATtiny26	16	Emulado por USI
ATtiny261/461/861	16	Emulado por USI
ATtiny28	11	NO
ATtiny48/88	28	SI
ATmega8/48/88	28	SI

Tabla 2-1: Algunos microcontroladores, I/O e incorporación de I^2C

En el presente PFC se ha decidido usar la comunicación I^2C debido a su muy extendido uso en la mayoría de los aparatos electrónicos, en este sentido, podría usarse cualquier micro que, o bien admitiera el protocolo I^2C, o bien admitiera un protocolo compatible con éste.

- **Pequeño**

¿Para qué un micro de 80 pines si los requisitos se alcanzan con uno de 5? Basándose en este razonamiento se ha buscado el micro más pequeño posible, que permita las funcionalidades que se requieren, simplificando así el tamaño del micro y de cada módulo en particular.

[3] Numero de pines de entrada salida sin incluir los pines de alimentación (VCC) y masa (GND)

- **Programación en lenguaje C**

Los dos lenguajes más usados para la programación de microcontroladores son C(1) y ensamblador. Sería deseable un microcontrolador con un entorno de programación que permitiese la programación en ambos lenguajes. Puesto que el ensamblador es un lenguaje de bajo nivel, permite optimizar y compactar mucho el código, no obstante en ensamblador es más compleja la realización de cambios. Por esta razón, y por la simplicidad y facilidad de programar en lenguaje C, se usará dicho lenguaje para la programación de cada uno de los módulos. De este modo, cualquier persona que no tenga conocimientos de ensamblador podrá hacer las modificaciones que considere oportunas en un lenguaje sencillo y rápido de aprender que permanece invariable ante el paso del tiempo y que cada vez es más usado.

- **No solapamiento**

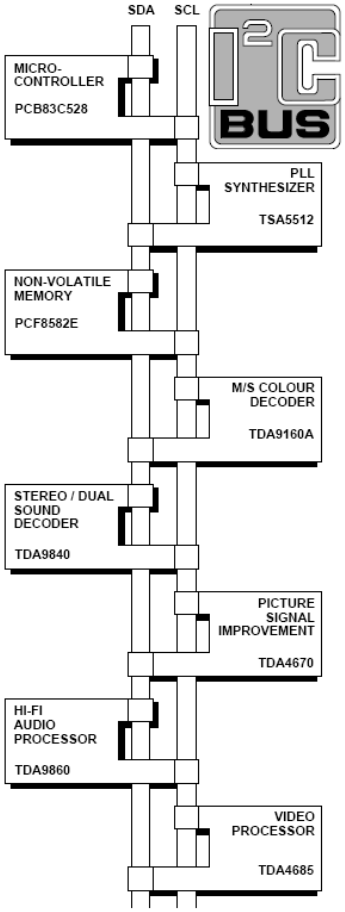
Finalmente, no basta con que se cumplan los requisitos anteriores, y es que los micros pequeños consiguen ser compactos debido a la multiplexación de pines, es decir, por un mismo pin se puede obtener comunicación I^2C y un canal de PWM por ejemplo. Hay que tener especial cuidado con este hecho ya que en ocasiones aunque un microcontrolador cumple los requisitos que se necesitan no es posible su uso por tener algunas funcionalidades multiplexadas. Así pues no debe permitirse solapamiento entre algunos tipos de pines, sin embargo otros pueden solaparse sin causar problemas. A continuación se enumeran los solapamientos permitidos y no permitidos.

- Solapamientos no permitidos
  - Pin de PWM (si se usa) y pines de comunicación I^2C.
  - Pines de detección de interrupciones (si se usa) y pines de comunicación I^2C.
  - Pin de reset con cualquier otro pin.
- Solapamientos permitidos
  - Pines de PWM (en su caso) y pines de SPI con jumper para evitar colisión durante la programación.
  - Pines de SPI y pines de I^2C con jumper para evitar colisión durante la programación.

2.2 Comunicaciones I^2C

- **Concepto**

Se conoce como I^2C a un bus de comunicaciones serie, síncrono y bidireccional de dos hilos[4], inventado por Philips Semiconductors a principios de 1980(2), muy usado en la industria, principalmente para comunicar componentes con cierto nivel de inteligencia como por ejemplo microcontroladores, con circuitos de propósito general como pantallas LCD, memorias, etc. y circuitos aplicados tales como DSP's que residen en un mismo circuito integrado.



El estándar de facto I^2C es un acrónimo de Inter-IC ó Inter Integrated Circuits y Philips Labs[5] lo desarrolló en Holanda inicialmente para comunicar, en una tarjeta de televisión, la CPU con los periféricos de manera que se evitase el ruido electromagnético que se producía al tratar de comunicar muchos periféricos con la CPU mediante un bus de datos de varias líneas impreso en un circuito integrado(3).

El bus I^2C ha sido adoptado en la actualidad por muchos fabricantes de chips entre ellos Xicor, ST Microelectronics, Infineon Technologies, Intel, Texas Instruments, Maxim, Atmel, Analog Devices, etc. Permite velocidades de comunicación de 100 kbits/s, 400 kbits/s y en algunos casos por encima de 1 Mbit/s y de 3,4 Mbits/s[6](4). Además ya no solo se usa para conexiones dentro de un mismo circuito integrado sino que conecta diferentes PCB's mediante cable.

Figura 2-2: Ejemplo de I^2C (extraída de **(5)**)

[4] Un hilo se usa para transmitir el reloj y se conoce como SCL, el otro hilo se usa para el intercambio de datos y recibe el nombre de SDA.

[5] Philips sigue siendo la referencia del protocolo I^2C y en su página web www.nxp.com se publican periódicamente revisiones del protocolo. A fecha de hoy la última revisión es la del 19 de Junio de 2009.

[6] Originalmente se diseñó para 100kbps (modo standard), se permitió 400kbps (modo fast) para algunas aplicaciones que lo requerían y posteriormente está disponible para una velocidad superior a 1 Mbps (modo fast plus) y superior a 3.4Mbps (modo high speed).

Esta flexibilidad y simplicidad hace que I^2C sea una alternativa ciertamente interesante para comunicar los diferentes módulos que componen el robot sobre el que trata este PFC.

Algunos fabricantes, como por ejemplo Atmel usan TWI (Two Wire Interface) que es esencialmente I^2C pero implementado en circuitos de otros fabricantes. El hecho de usar otro nombre es para no tener que pagar la licencia de uso, aunque actualmente I^2C es de uso libre(6).

- **Consideraciones hardware**

Además de las líneas SDA (Serial Data) y SCL (Serial Clock), el estándar I^2C necesita una tercera línea, la masa ó tierra; normalmente no se mencionaba debido a que el protocolo solía usarse dentro de un mismo circuito integrado y por tanto la masa era común a ambos.

En la Figura 2-3 se muestra una imagen que incluye todos los componentes que afectan a la comunicación I^2C; tales son, las resistencias de pull-up (Rp), las resistencias serie (Rs), la capacitancia del bus (Cp) y la capacitancia del crosstalk (Cc).

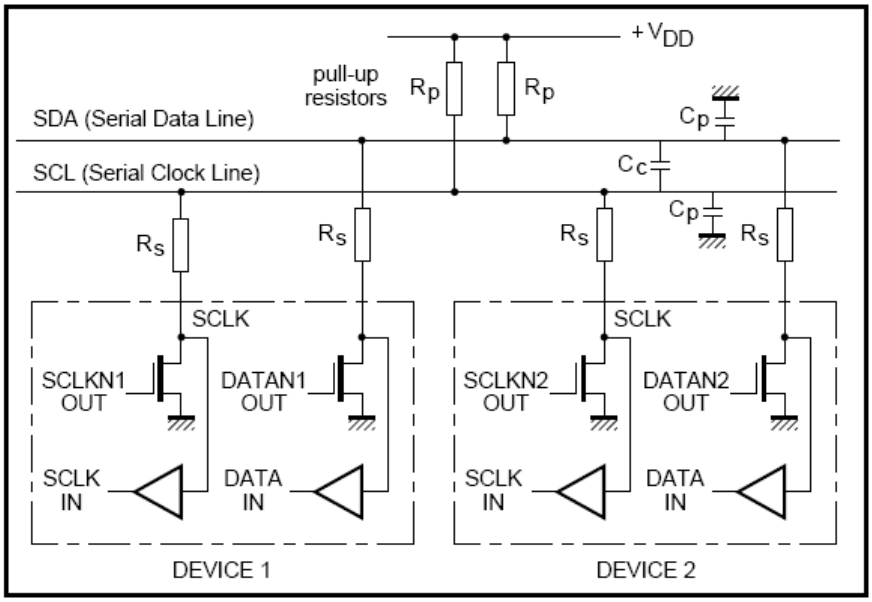


Figura 2-3: Componentes en una comunicación I^2C (extraída de **(5)**)

Para permitir que tanto SDA como SCL puedan actuar de forma bidireccional es necesario que éstas sean líneas de drenaje abierto o colector abierto[7], dependiendo de la

[7] El drenaje abierto es un estado similar al de colector abierto pero asociado a un transistor de efecto de campo (FET).

tecnología usada. Por tanto deben polarizarse en estado alto; eso se consigue conectando los pines a la alimentación mediante resistencias (pull-up). Los voltajes típicos que suelen usarse son +5V. ó +3.3V. aunque se permiten tensiones superiores e inferiores.

El valor de las resistencias de polarización (Rp) varía entre 1k8Ω y 47kΩ aunque los valores más usados son entre 1kΩ y 10kΩ. Con valores pequeños de resistencia se disminuye la sensibilidad al ruido y mejora el tiempo de los flancos de subida y bajada de las señales a base de incrementar el consumo de los integrados.

En I^2C el número máximo de dispositivos que se pueden conectar está limitado (en cuanto al hardware) por la capacitancia máxima del bus (Cp) que es de 400pF., según el estándar, lo que limita la longitud del bus a unos pocos metros. No obstante, como se verá a continuación, pueden usarse buses con una mayor capacitancia a base de disminuir las resistencias Rp, aunque no es aconsejable. Se recomienda que los buses sean lo más cortos posibles para minimizar la capacitancia que depende drásticamente de la longitud de las líneas.

Tanto la capacitancia Cp, como las resistencias Rp, afectan al comportamiento temporal de las señales SDA y SCL.

Los transistores FET usados para forzar la línea a cero suelen consumir bastante más que las resistencias Rp que devuelven la señal al estado alto dando lugar a unas señales con forma de diente de sierra. En cada uno de esos dientes, se puede observar la forma con que se carga la línea (actúa el transistor FET) en el flanco de bajada y la forma con que se descarga (actúa la resistencia Rp) en el flanco de subida.

A continuación se pretende mostrar el efecto que tienen sobre SDA y SCL la capacitancia del bus y el valor de las resistencias de pull-up(7).

En la fila superior de la Figura 2-4, se muestran las señales SDA (verde) y SCL (rojo) con una capacitancia de 300pF. y unos pull-up de 10kΩ.

En la fila inferior izquierda se muestra el efecto de suavizado en el flanco de subida a costa de disminuir los pull-up a un valor de 2kΩ., dejando patente así el hecho de que pueden usarse buses con elevada capacitancia a base de disminuir el valor de los pull-up, aunque no sea aconsejable.

En la fila inferior derecha se muestra el efecto de suavizado en el flanco de subida a costa de reducir la capacitancia a un valor de 150pF. manteniendo los pull-up iniciales de valor 10kΩ.

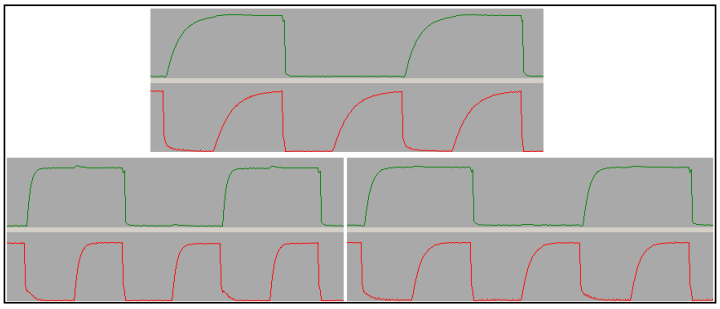


Figura 2-4: SDA y SCL para distintos valores de Cp y Rp (extraída de **(7)**)

También ha de tenerse en cuenta que en algunos circuitos (especialmente en aquellos cuyo bus I^2C es accesible mediante conectores) se deberían añadir unas resistencias en serie (Rs) con SDA y SCL para proteger al dispositivo de sobrecargas. No obstante, Rs afecta, junto con Rp a la tensión existente en nivel bajo según la fórmula:

$$\frac{Rs}{Rs + Rp} \times Vcc \qquad [2.1]$$

Así, si se usan resistencias Rs muy grandes, la tensión en nivel bajo será elevada, corriendo el riesgo de que al enviar un cero lógico por el bus éste sea interpretado como un uno lógico.

Por último, no se debe olvidar el efecto de crosstalk[8] (Cc) que se logra minimizar incrementando el valor de Rs y Rp.

- **Consideraciones software**

Cada dispositivo conectado al bus tiene una dirección única definida por siete bits. Para direccionar un dispositivo se usan sus siete bits además de un bit que indica si la operación que se realizará sobre el dispositivo será de escritura o de lectura. De este modo, todos los dispositivos tienen en realidad dos direcciones, la dirección de sólo lectura y la dirección de sólo escritura dependiendo del octavo bit transmitido. Existe una dirección usada para

[8] Entre dos circuitos o cables se dice que existe crosstalk o diafonía cuando parte de las señales presentes en uno de ellos aparece en el otro.

comunicarse con todos los dispositivos del bus, se trata de la dirección 0x00, conocida como llamada general.

El número máximo de dispositivos que se pueden conectar al bus está limitado (en software) por la cantidad de dispositivos que se pueden direccionar[9], esto es, 128[10].

Los dispositivos conectados al bus se dividen en maestro y esclavos. El maestro es aquel que controla el reloj y envía la dirección del dispositivo con el que desea comunicarse. Los esclavos son los que reciben el reloj y pueden ser direccionados por el dispositivo maestro.

Si en un bus I^2C existe más de un dispositivo maestro, éste se conoce como bus multi-maestro y en estos casos se necesitan mecanismos para evitar la colisión.

Para que un bit enviado por la línea SDA se considere válido debe permanecer estable durante el periodo en el que la línea SCL esté en el estado alto lógico. Pudiendo cambiarse el bit mientras la línea SCL se encuentre en nivel bajo lógico.

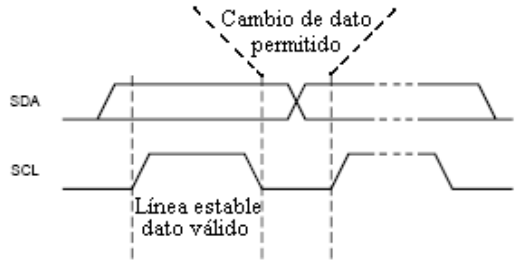


Figura 2-5: Condiciones de validez de datos en un bus I^2C (extraída de **(5)**)

La violación de esta regla de datos válidos da lugar a las tres condiciones especiales(8):

- La condición de START; es indicada mediante la transición de nivel alto lógico a nivel bajo lógico de la línea SDA manteniendo el nivel alto lógico la línea SCL[11].
- La condición de STOP; es indicada mediante la transición de nivel bajo lógico a nivel alto lógico de la línea SDA manteniendo el nivel alto lógico la línea SCL.
- La condición de REPEATED START; se genera igual que la de START pero tiene un uso distinto que se detalla más adelante.

[9] Existen una serie de direcciones reservadas, dichas direcciones aparecen en la Tabla A-1 del anexo A de este PFC.

[10] Al ser direcciones de 7 bits, existen un total de $2^7 = 128$ direcciones posibles.

[11] Debemos recordar que ningún dispositivo puede llevar las líneas SCL ni SDA a un estado alto lógico, en realidad, lo que hace es liberar la línea de modo que la resistencia de pull-up la lleva a nivel alto.

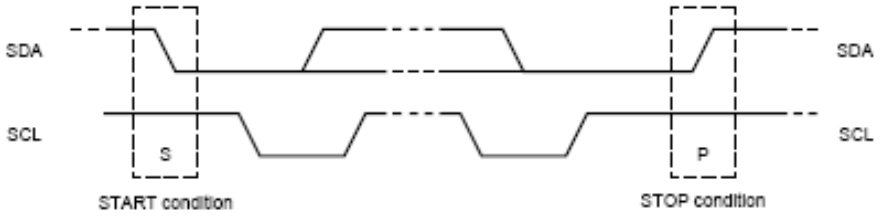


Figura 2-6: Condiciones de START y STOP en un bus I^2C (extraída de **(5)**)

La comunicación según el protocolo I^2C ocurre del siguiente modo(9):

1. El maestro envía un comando de inicio (condición de START).
2. El maestro envía los siete bits de la dirección del esclavo con el que desea establecer la comunicación. A continuación envía un bit de lectura/escritura. Si el bit es un cero lógico expresa el deseo de escribir por parte del maestro, por el contrario, si el bit de lectura/escritura es un uno lógico, entonces el maestro desea leer información del esclavo.
3. El esclavo cuya dirección es la enviada por el maestro responderá con un bit de asentimiento que consiste en poner la línea SDA en el estado bajo lógico.
4. El maestro enviará o recibirá información según el bit de lectura/escritura previamente enviado. La información se envía o recibe por bytes y para cada byte, ocurre un asentimiento por parte del receptor, así que se producen nueve pulsos de reloj por cada ocho bits de datos transmitidos. Tanto la dirección como los bytes de datos son enviados en formato MSBF (primero el bit más significativo).
5. Una vez finalizado el envío o recepción de datos, el maestro enviará un comando de finalización (condición de STOP) o bien otro comando de inicio (condición de REPEATED START) si desea iniciar una nueva lectura/escritura con el mismo esclavo u otro distinto y posteriormente repetir los pasos a partir del número 2.

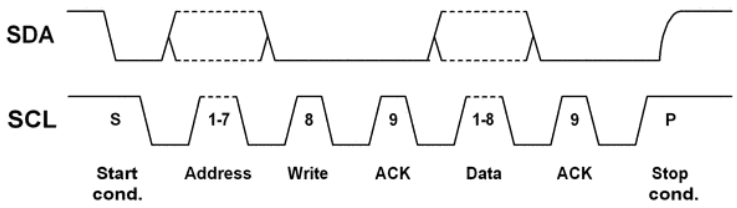


Figura 2-7: Fases del protocolo de comunicaciones I^2C (extraída de (10)**)**

Una característica interesante del bus I^2C es el denominado estiramiento del reloj o clock stretching. Así es como se conoce al hecho de que un dispositivo esclavo, con el que se está estableciendo comunicación, mantenga la línea SCL en el estado bajo lógico. Este fenómeno puede ocurrir tanto durante el bit de asentimiento a la dirección como durante los asentimientos a los datos de información, y se usa para indicar que aún no está listo para procesar más datos. La cuestión es que el maestro deberá liberar la línea SCL y comprobar que ha pasado al estado lógico alto antes de enviar cada bit; si comprueba que la línea SCL se mantiene en estado bajo lógico, interpreta que el esclavo la mantiene así y espera a que SCL pase al estado alto lógico, consiguiendo así "esperar al esclavo".

Algunos dispositivos no permiten el estiramiento del reloj, y se dice que soportan el protocolo TWI y no estrictamente I^2C.

2.3 Comunicaciones SPI

- **Concepto**

El bus serial para interfaz de periféricos, conocido como SPI es un protocolo de comunicaciones serie síncrono, desarrollado por Motorola, que funciona en modo full dúplex y permite la transferencia de información entre circuitos integrados de equipos electrónicos(11).

El protocolo fue diseñado para trabajar con IC's de un mismo PCB puesto que por las altas velocidades a las que opera si las líneas que lo integran fueran demasiado largas se incrementaría mucho su reactancia.

A veces al estándar SPI (Figura 2-8 extraída de (12)) se lo conoce como comunicación de 4 hilos, para enfatizar la diferencia con otras comunicaciones de 3, 2 y 1 hilos respectivamente.

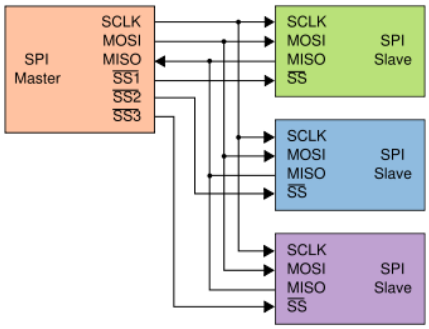


Figura 2-8: Ejemplo de SPI

Los dispositivos que interactúan en el estándar SPI se clasifican en maestros o esclavos. El maestro es quien inicia la trama de red o *data frame* y los esclavos son quienes están conectados a las líneas SS o de *Slave* Select individuales. Las señales que intervienen en la comunicación maestro-esclavo son una línea de reloj, una línea para recepción de datos, una línea para envío de datos y por último una línea para seleccionar el esclavo.

- **Hardware y funcionamiento**

Como se dijo anteriormente, el hardware consiste en cuatro señales;

- SCLK: Serial Clock (salida desde el maestro)
- MOSI: Master Output, Slave Input (salida desde el maestro)
- MISO: Master Input, Slave Output (salida desde el esclavo)
- $\overline{SS}$: Slave Select (salida desde el maestro y activa a nivel bajo)

Dependiendo de los dispositivos en los que se implementa el protocolo, es posible que los nombres de las cuatro señales difieran de las mencionadas aquí.

Cuando en la comunicación sólo intervienen un maestro y un esclavo, se suele fijar a nivel bajo la señal $\overline{SS}$, ya que el maestro se comunicará forzosamente con el único esclavo que hay; no obstante algunos dispositivos requieren que ocurra un flanco de bajada para ser seleccionados de modo que no se recomienda conectar directamente la señal $\overline{SS}$ a tierra.

Cuando existen varios esclavos en la comunicación el maestro deberá poner a nivel bajo la línea $\overline{SS}$ conectada al esclavo con el que desea iniciar la comunicación.

Antes de comenzar la comunicación el maestro fija la velocidad del reloj según la frecuencia máxima del esclavo con el que desea comunicarse. La frecuencia suele estar comprendida entre 1MHz. y 70MHz. Posteriormente fija a nivel bajo la señal $\overline{SS}$ correspondiente y durante cada ciclo de reloj se produce una comunicación full dúplex en la que el maestro envía un bit por la línea MOSI que el esclavo recibe y el esclavo envía un bit por la línea MISO que el maestro recibe (13).

Esto es así debido a la singular conexión del protocolo. En dicha conexión la entrada MISO del maestro está conectada a uno de los extremos de un registro de desplazamiento interno y el otro extremo del registro está conectado a la salida MOSI. Por su parte el esclavo tiene la conexión análoga. De modo que tras, típicamente[12], ocho ciclos de reloj el dato colocado en el registro de desplazamiento del maestro pasa a estar situado en el registro de desplazamiento del esclavo y viceversa, consiguiendo una transferencia de comunicación simultanea La información y la figura adjunta han sido extraídas de (14).

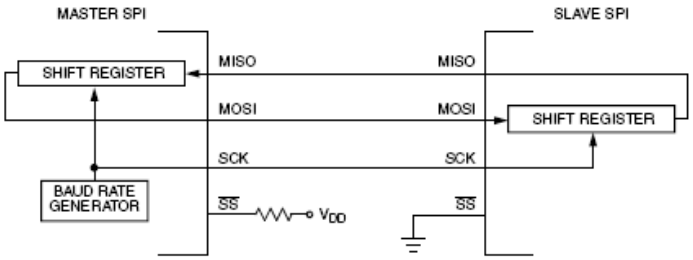


Figura 2-9: Diagrama de boques transferencia SPI Maestro/Esclavo

Además de configurar la frecuencia de reloj es tarea del maestro configurar tanto la polaridad del mismo como su fase. La polaridad del reloj determina si en el estado "idle", es decir, antes de comenzar la transferencia el reloj está en nivel alto lógico o bajo lógico.

[12] Normalmente se transmiten 8 bits de datos pero pueden usarse palabras de mayor longitud aunque siempre se usa el sistema MSBF por lo que se envía primero el bit más significativo.

La fase del reloj es el parámetro que determina si el dato se lee durante el flanco de subida o durante el flanco de bajada. Típicamente esta configuración se define mediante dos bits, CPHA y CPOL. En la Tabla 2-1 se muestra de manera esquemática el uso de ambos bits.

Fase y polaridad de SCLK	CPOL = 0	CPOL = 1
CPHA = 0	En estado "idle" el reloj se encuentra en nivel bajo lógico. El dato se lee en el flanco de subida y se cambian en el flanco de bajada.	En estado "idle" el reloj se encuentra en nivel alto lógico. El dato se lee en el flanco de bajada y se cambian en el flanco de subida.
CPHA = 1	En estado "idle" el reloj se encuentra en nivel bajo lógico. El dato se lee en el flanco de bajada y se cambian en el flanco de subida.	En estado "idle" el reloj se encuentra en nivel alto lógico. El dato se lee en el flanco de subida y se cambian en el flanco de bajada.

Tabla 2-2: Significado del valor de los bits CPHA y CPOL

En algunas notaciones se usan modos para designar a las tuplas CPHA-CPOL. Existen cuatro modos de SPI, los correspondientes a las cuatro combinaciones posibles.

- **Ventajas y desventajas**(15)

Por último se citan algunas ventajas:

- Comunicación Full Duplex
- Mayor velocidad de transmisión que con I²C o SMBus
- No está limitado a la transferencia de bloques de 8 bits
- Consume menos energía que I²C o que SMBus al no poseer resistencias de pull-up
- Los dispositivos esclavos no necesitan su propio reloj
- Usa una única señal específica para cada esclavo ($\overline{SS}$), las demás son compartidas

Y algunas desventajas:

- El direccionamiento se hace mediante líneas específicas (señalización fuera de banda) a diferencia de lo que ocurre en I²C que la dirección de 7 bits se envía por el propio bus
- No hay control de flujo por hardware
- No hay señal de asentimiento. El maestro podría no estar enviando información a nadie
- No permite fácilmente tener varios maestros conectados al bus

En este PFC el protocolo SPI se usará para cargar el firmware implementado en cada uno de los módulos, aprovechando que los microcontroladores de Atmel disponen de SPI.

3 Módulo de locomoción

3.1 Motivación del módulo

La primera funcionalidad de la que se ha querido dotar a la plataforma robótica ha sido de la movilidad. Es deseable que el sistema sea capaz de trasladarse de un lugar a otro para poder desempeñar una tarea, ésto se consigue típicamente mediante motores, que al girar, son capaces de convertir su movimiento radial en un avance longitudinal.

Y eso es precisamente lo que busca el denominado módulo de locomoción. Pretende ser un módulo capaz de proporcionar movimiento a la plataforma robótica[13] y trasladarla de un lugar a otro dependiendo de las órdenes de otro módulo, al que se denominará módulo maestro.

Las órdenes del módulo maestro serán recibidas y correctamente interpretadas por el módulo de locomoción gracias al bus de comunicaciones que interconecta toda la plataforma robótica. Dicho bus de comunicaciones sigue el protocolo estándar I^2C detallado con anterioridad en el apartado 2.2.

Dado que todo el sistema esta interconectado, para que el módulo de locomoción tan solo ejecute las órdenes que van dirigidas a él, es necesario identificarlo de algún modo. Para ello se usan las direcciones I^2C.

La dirección I^2C grabada en el firmware del módulo de locomoción es la 0x04, no obstante, los dos últimos bits de dicha dirección son configurables mediante hardware. De este modo, las posibles direcciones I^2C del módulo de locomoción son 0x04, 0x05, 0x06 y 0x07. La razón de permitir estas cuatro direcciones, es que en el mismo sistema puedan coexistir un total de hasta cuatro clones de éste módulo.

Gracias al hecho de poder configurar por hardware hasta cuatro direcciones I^2C, el sistema es capaz de manejar hasta ocho motores de forma independiente (ya que cada módulo de locomoción maneja dos motores). La configuración hardware de la dirección del módulo se explica en el apartado 3.6.

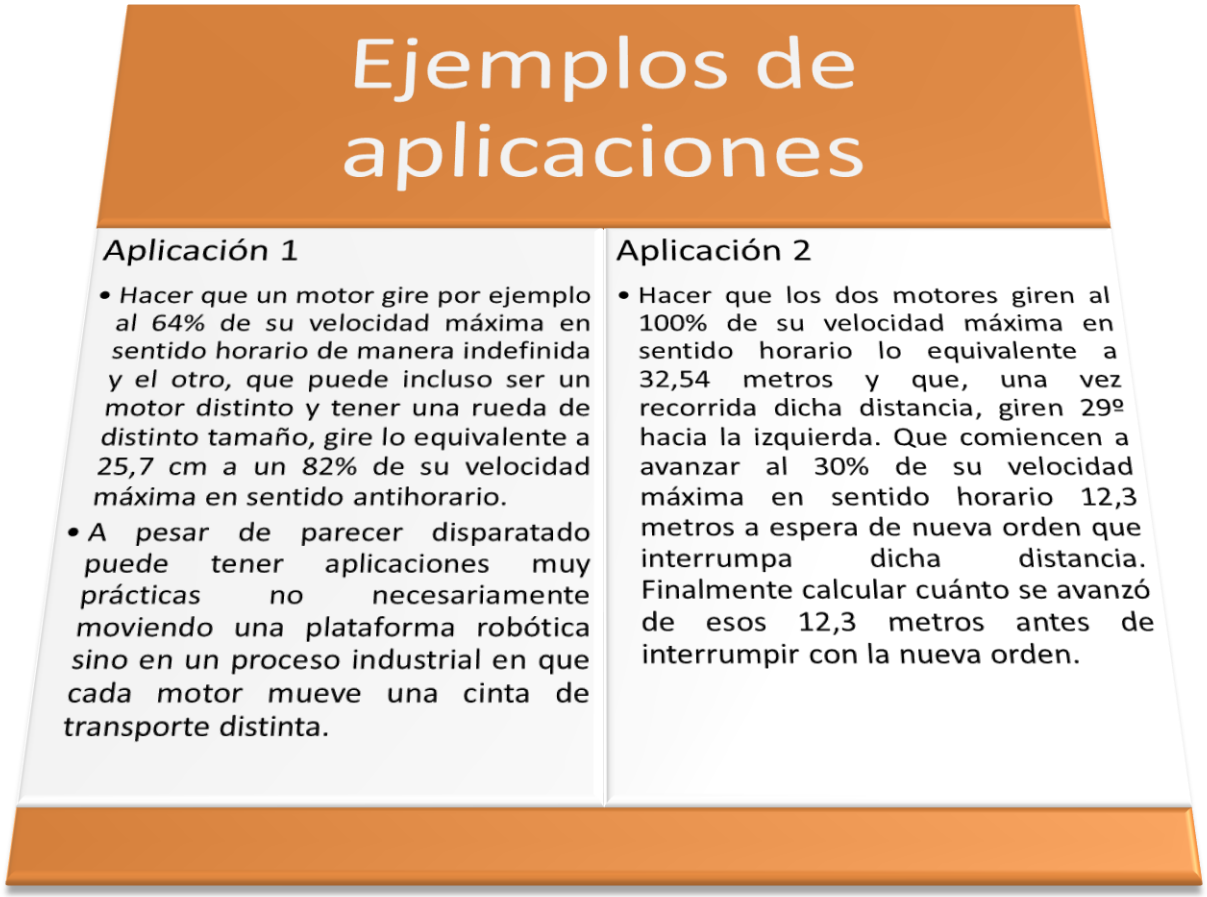
[13] En realidad se ha dotado de tal funcionalidad al módulo, que puede, no solo usarse para mover una plataforma robótica, sino aplicarse a cualquier situación que requiera el dominio de uno o varios motores de manera independiente o conjunta.

3.2 Funcionalidad

Como se dijo en el apartado 3.1, el módulo de locomoción es capaz de interpretar una serie de órdenes recibidas por I^2C. En este apartado pretenden darse a conocer todas las órdenes que es capaz de interpretar el módulo.

1) Girar de manera indefinida en cualquier sentido uno o ambos motores
2) Girar una distancia determinada (min. 1mm. longitudinal) uno o ambos motores
3) Girar a una velocidad concreta (0-100%) uno o ambos motores
4) Provocar un movimiento tal que el módulo gire xº (1-360º) en cualquier sentido
5) Conocer si algún motor está girando y en su caso, cuál
6) Parar uno o ambos motores
7) Saber cuánto han girado uno o ambos motores desde la última orden

Combinando de manera adecuada las órdenes anteriores, se pueden realizar acciones complejas e interesantes tales como las que se muestran a continuación.



3.2.1. Motores de corriente continua

De los cuatro tipos de motores más conocidos (motores de CC, motores de CA, servomotores, y motores paso a paso), en este proyecto usaremos los motores de corriente continua, también llamados motores de CC por sus siglas en español y motores DC por sus siglas en inglés.

- **Concepto**

Son los más usados y los más económicos, se pueden encontrar en diferentes tamaños y con diferentes potencias. Su función es convertir energía eléctrica en energía mecánica, típicamente en movimiento rotatorio.

Figura 3-1: Motor de CC

En la Figura 3-1 se muestra un motor pequeño de corriente continua extraído de un catálogo de componentes.

Hacerlos funcionar es tan simple como aplicar una tensión eléctrica, especificada por el fabricante, entre sus bornes. Si se aplica la tensión inversa entonces el motor girará en sentido inverso.

A diferencia de los servomotores y los motores paso a paso, los motores de corriente continua no pueden posicionarse en una posición concreta, simplemente giran a la máxima velocidad dependiendo de la tensión aplicada y en el sentido que la corriente imponga.

Para tener control acerca de la posición y del número de vueltas que da el motor deben aplicarse otros elementos, como por ejemplo los encoders; existen numerosos tipos y no serán analizados en este PFC aunque se comentará ligeramente su función en el apartado 3.4.1.

- **Constitución y funcionamiento**

Los motores de CC están formados generalmente por las siguientes partes(16):

- Inductor o estator (Arrollamiento de excitación): Es un electroimán formado por un número par de polos. Las bobinas que los arrollan son las encargadas de producir el campo inductor al circular por ellas la corriente de excitación.

- Inducido o rotor (Arrollamiento de inducido): Es una pieza giratoria formada por un núcleo magnético alrededor del cual va el devanado de inducido, sobre el que actúa el campo magnético.
- Colector de delgas: Es un anillo de láminas de cobre llamadas delgas, dispuesto sobre el eje del rotor que sirve para conectar las bobinas del inducido con el circuito exterior a través de las escobillas.
- Escobillas: Son unas piezas de grafito que se colocan sobre el colector de delgas, permitiendo la unión eléctrica de las delgas con los bornes de conexión del inducido.

En la Figura 3-2 pueden verse con más detalle las partes involucradas

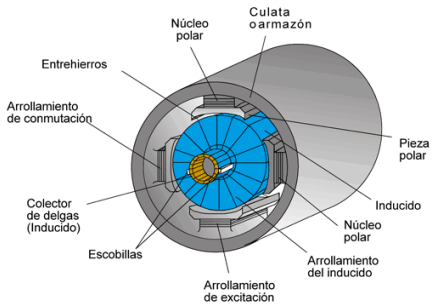


Figura 3-2: Constitución de un motor de CC. extraída de **(16)**

Al girar el rotor, las escobillas van rozando con las delgas, conectando la bobina de inducido correspondiente a cada par de delgas con el circuito exterior.

El funcionamiento de un motor de corriente continua se basa en que cuando una corriente eléctrica pasa a través de un cable conductor inmerso en un campo magnético, la fuerza magnética produce un par el cual provoca el giro del motor. El campo magnético es definido por la ley de Lorentz (17).

Se obtendrá el valor máximo de fuerza cuando el campo magnético sea perpendicular al conductor y se tendrá una fuerza nula cuando el campo sea paralelo al flujo de corriente eléctrica.

El motor que se usará para el módulo de locomoción es el Faulhaber 1624E012S495 tiene una reductora de 141:1 y un encoder magnético de cuadratura, HES201. Puede verse su forma y dimensiones en la Figura 3-3.

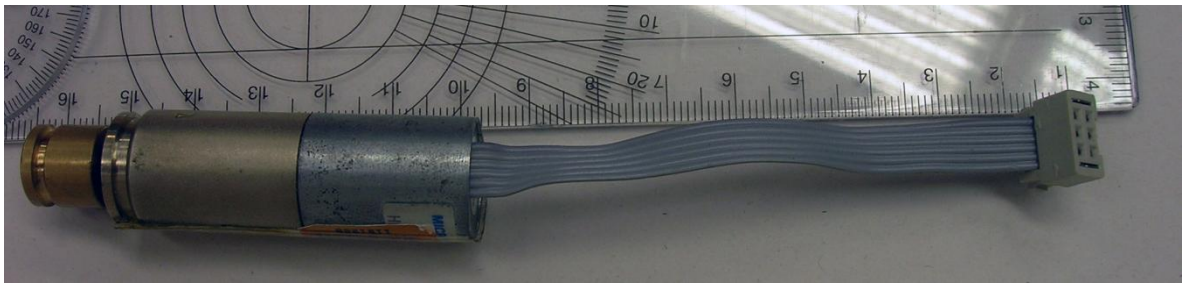


Figura 3-3: Motor usado en el módulo de locomoción, Faulhaber 1624E012S495

Gracias a la Figura 3-4, extraída de Internet, puede deducirse que a una tensión de alimentación de 9,5V. consumirá una corriente de 28mA. y girará a una velocidad máxima de 131 revoluciones por minuto. El conjunto final entrega un par de unos 116mNm., a 12V.

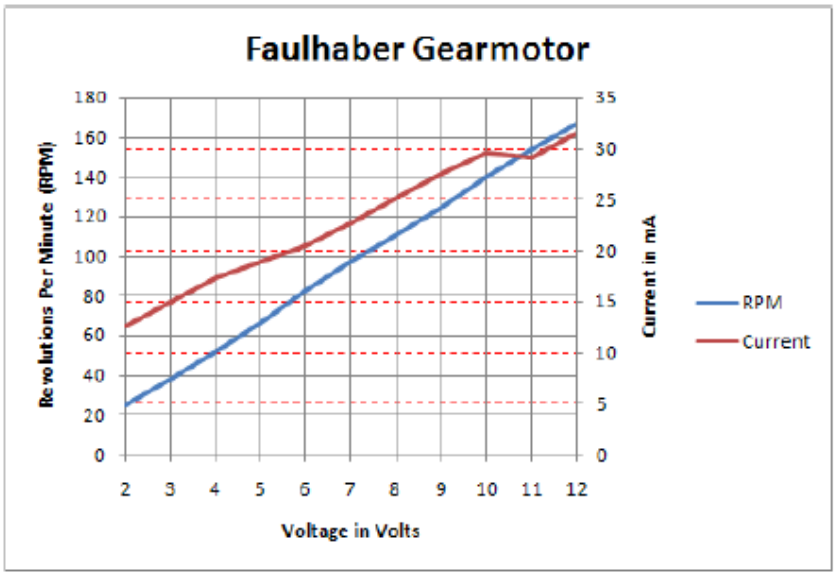


Figura 3-4: RPM y corriente en función de la tensión

La distribución de pines del motor se muestra en la figura Figura 3-5, donde A y B son los dos canales del encoder, 0V. y 5V. son la alimentación del mismo y por último M+ y M- son las entradas de tensión para el motor.

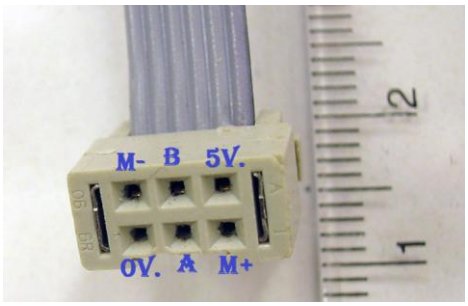


Figura 3-5: Distribución de pines motor Faulhaber 1624E012S495

3.3 Puentes en H y drivers de potencia

Los pines de salida de los microcontroladores son capaces de entregar una corriente mucho menor que la corriente que necesitan los motores para trabajar. Por esta razón, si se conecta directamente un motor a los pines de un microcontrolador, estos posiblemente se quemen. Para evitar esta situación lo que se hace es introducir una etapa intermedia entre ellos de manera que sea esa etapa la que se encargue de proporcionar a los motores la tensión y corriente que necesitan. El elemento que cumple la función de esa etapa intermedia se denomina driver de potencia. En ocasiones los driver de potencia están configurados especialmente para trabajar con motores. Se dice que tienen una configuración de Puente en H y gracias a eso permiten que el motor que gobiernan pueda girar en ambos sentidos. En los apartados siguientes se amplían ambos conceptos.

3.3.1. Teoría de puentes en H

Los Puentes H o Puentes en H son circuitos electrónicos cuya función es la de permitir que un motor eléctrico de corriente continua pueda girar en ambos sentidos, horario y antihorario; en la actualidad, los Puentes en H pueden comprarse en un circuito integrado o bien construirse a partir de componentes discretos (transistores y resistencias).

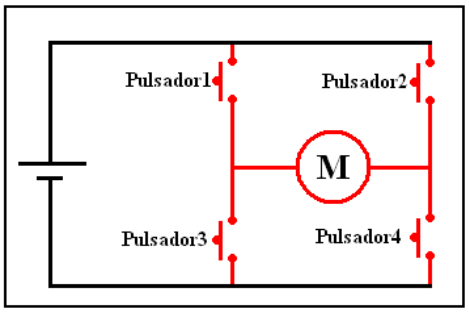


Figura 3-6: Esquema de Puente en H

En la Figura 3-6 se presenta un esquema funcional de un puente en H, obsérvese que el nombre de este circuito proviene de la analogía física del mismo a la letra H (coloreado en rojo para mayor similitud). En dicha figura se muestra un puente H implementado mediante pulsadores, pero en la realidad, los pulsadores se sustituyen por transistores, y deben añadirse además algunas resistencias de modo que se gobiernen mediante señales de control los cuatro pulsadores.

El esquema final de un puente H más realista es el que se muestra en la Figura 3-7. No obstante en dicha figura se han omitido los diodos de protección (por simplicidad). Éstos deben situarse en torno al motor para suprimir los picos de tensión provocados por la inducción de sus bobinas al comenzar a girar o cambiar su sentido de giro.

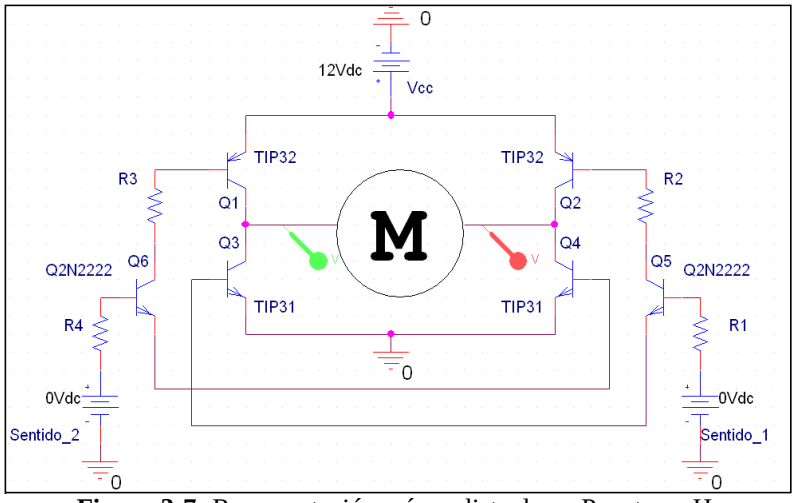


Figura 3-7: Representación más realista de un Puente en H

En la Figura 3-7 se han añadido dos marcadores de tensión (en verde y en rojo) para comprobar cuál es la tensión que hay en los extremos del motor. Para simular las señales de control, se han añadido dos fuentes de tensión de corriente continua (Sentido_1 y Sentido_2). También se ha añadido una fuente de tensión de 12V. de corriente continua que será quien alimente al motor.

Colocando 5V. en Sentido_1 y 0V. en Sentido_2 se consigue que el transistor Q5 conduzca. La corriente (en marrón) de dicho transistor circulará por las bases de Q2 y Q3 creando así un camino de circulación para la corriente (en amarillo) entregada por Vcc, haciendo finalmente girar al motor en un sentido. En la Figura 3-8 izquierda se muestra el camino que seguirá la corriente y en la Figura 3-8 derecha se muestra la tensión que habrá en los extremos del motor (recogidos por los marcadores rojo y verde).

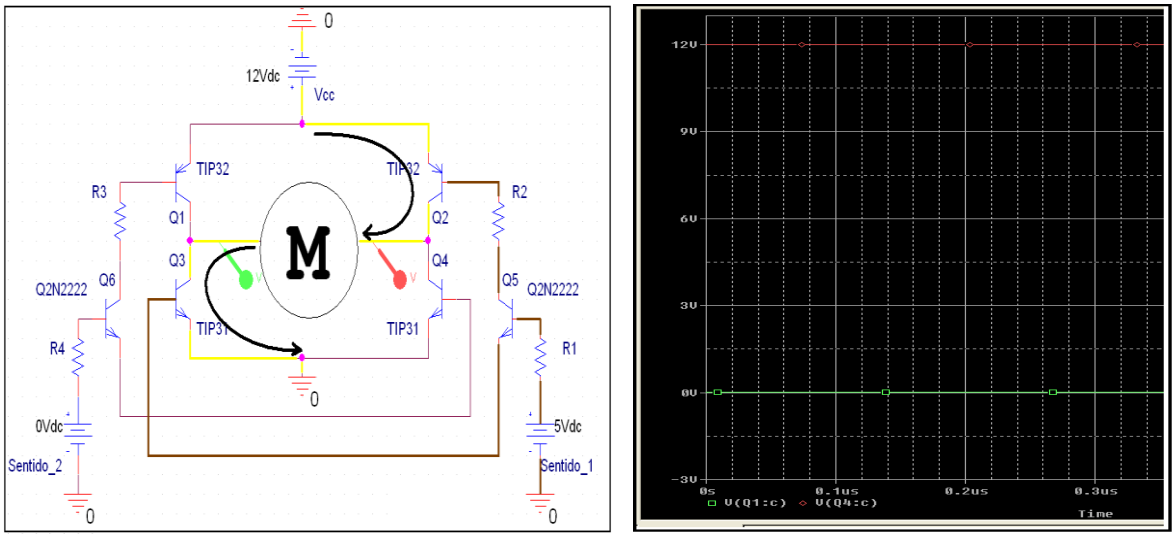


Figura 3-8: Circulación con Sentido_1 en on (izquierda) y tensión en bornas del motor (derecha)

Colocando 5V. en Sentido_2 y 0V. en Sentido_1 se consigue que el transistor Q6 conduzca, la corriente (en marrón) de dicho transistor circulará por las bases de Q1 y Q4 creando así un camino de circulación (en amarillo) para la corriente entregada por Vcc, haciendo finalmente girar al motor en el otro sentido. En la Figura 3-9 izquierda se muestra el camino que seguirá la corriente y en la Figura 3-9 derecha se muestra la tensión que habrá en los extremos del motor (recogidos por los marcadores rojo y verde).

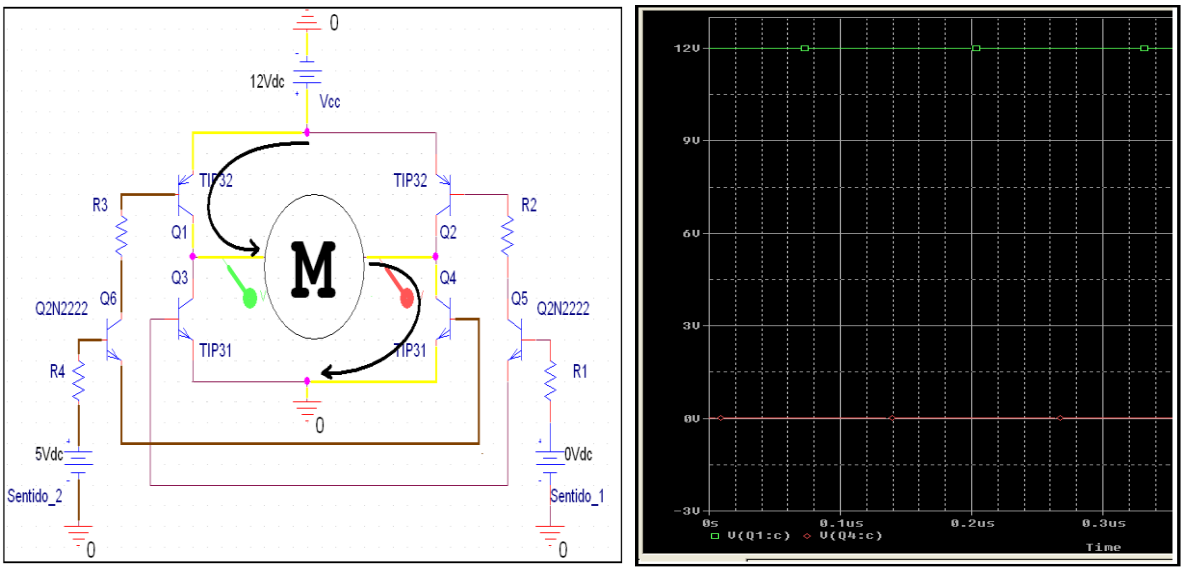


Figura 3-9: Circulación con Sentido_2 en on (izquierda) y tensión en bornas del motor (derecha)

Un hecho que se debe tener muy en cuenta a la hora de trabajar con Puentes en H es que las señales de control (Sentido_1 y Sentido_2) bajo ningún concepto pueden estar en HIGH simultáneamente, ya que haría que los transistores Q1, Q2, Q3 y Q4 tomarán el estado de conducción, habilitando la circulación directa de corriente entre Vcc y GND, cosa que dañaría tanto los transistores como la fuente de alimentación si ésta no tuviera la protección adecuada. Para evitar este hecho, se suelen usar lo que llamamos *circuitos de interlock* (Figura 3-10).

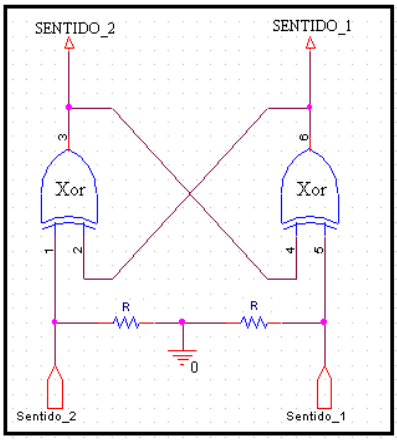


Figura 3-10: Circuito de interlock

Gracias al *circuito de interlock* si se ponen en estado alto lógico simultáneamente tanto la señal Sentido_1 como la señal Sentido_2 entonces las señales SENTIDO_1 y SENTIDO_2 tomarán el estado bajo lógico evitando así que los cuatro transistores conduzcan a la vez.

3.3.2. Puentes en H probados

Para la realización de este PFC se han probado tres circuitos integrados que tenían implementado un Puente en H.

En primer lugar se probó el integrado **L293** de STMicroelectronics[14] que es un driver de cuatro canales capaz de proporcionar una corriente de salida de hasta 1A por canal a una tensión de cómo máximo 36V. Este integrado es ampliamente usado para el control de motores actuando como Puente en H en aplicaciones robóticas muy básicas.

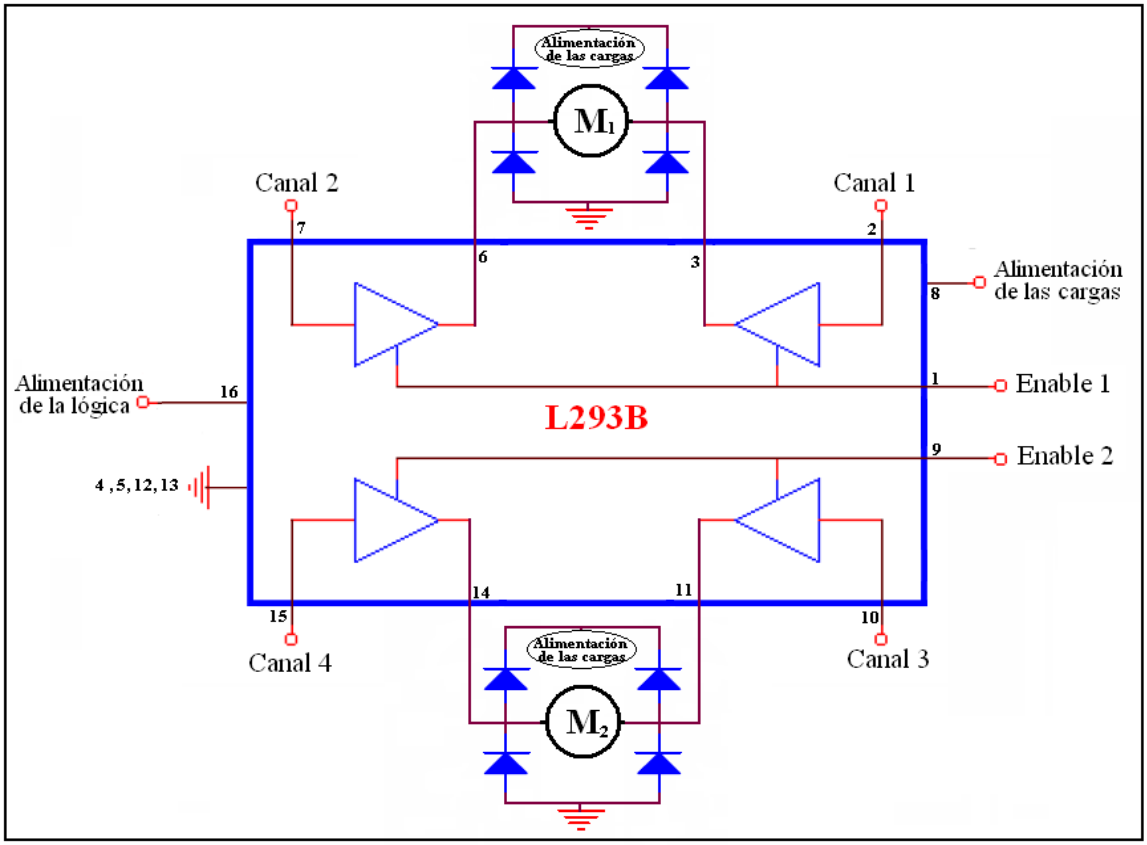


Figura 3-11: Esquema de conexión de un L293B

Dispone de un pin de habilitación para los canales 1 y 2 y otro pin de habilitación para los canales 3 y 4 de modo que se pueda habilitar de forma independiente una u otra pareja de canales para controlar un motor en los dos sentidos (o dos motores en un sentido). Este integrado carece de diodos de protección para proteger al motor ante posibles picos de corriente durante el arranque, así pues, deben añadirse dichos diodos.

[14] Para más información acerca del integrado puede acceder al datasheet del fabricante en http://www.st.com/stonline/

En la Figura 3-11 se mostraba un posible esquema de conexión para un L293B que controla el doble sentido de giro de dos motores.

Cuando el canal uno se pone a nivel alto y el canal dos se pone a nivel bajo, el motor uno gira en sentido antihorario, y si el canal uno se pone a nivel bajo y el canal dos se pone a nivel alto el motor uno gira en sentido horario. El mismo razonamiento puede seguirse con el motor dos y los canales tres y cuatro. Si dos canales asociados a un motor tienen simultáneamente el mismo valor, entonces el motor parará rápidamente, por otro lado, si se pone a nivel bajo el pin de *enable* asociado a un motor, éste girará libremente (por inercia) hasta parar.

En segundo lugar se probó el integrado **UCC37324** de Texas Instruments[15] que es un driver de dos canales capaz de proporcionar una corriente de salida de hasta 4A por canal a una tensión de cómo máximo 16V.

En algunos casos no son necesarios diodos de protección para este integrado ya que los pines de salida están dotados de una impedancia muy baja ante picos de corriente gracias a los diodos del los transistores MOSFET, no obstante sería recomendable usarlos. Dispone de ocho pines, por lo que tan solo puede controlar el doble sentido de giro de un motor o el sentido único de dos motores; así, en cuanto a funcionalidad, equivale a ½ L293.

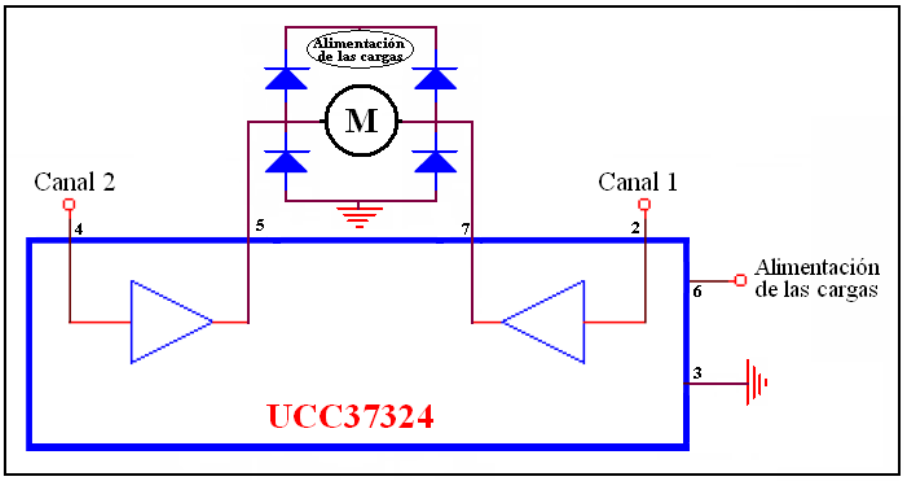


Figura 3-12: Esquema de conexión de un UCC37324

En la Figura 3-12 se muestra un posible esquema de conexión para un UCC37324 que controla el doble sentido de giro de un motor.

[15] Para más información acerca del integrado puede acceder al datasheet del fabricante en http://focus.ti.com/general/docs/prod.tsp

Cuando el canal uno se pone a nivel alto y el canal dos se pone a nivel bajo, el motor gira en sentido antihorario, y si el canal uno se pone a nivel bajo y el canal dos se pone a nivel alto el motor uno gira en sentido horario. Si dos canales asociados a un motor tienen simultáneamente el mismo valor, entonces el motor parará rápidamente.

En tercer lugar se probó el integrado **TLE4207G** de Infineon Technologies[16] que es un Puente en H capaz de proporcionar una corriente de salida de hasta 1A a una tensión de cómo máximo 45V.

Este integrado, al contrario que los dos analizados anteriormente, está dotado de diodos de protección por lo que basta con colocar el motor, cuyo sentido de giro deseamos controlar, entre los terminales de salida del integrado.

Además, el TLE4207G dispone de un pin que informa acerca de posibles sobretensiones o temperaturas excesivas[17] cosa de la que carecían los otros dos integrados analizados anteriormente.

Además, al igual que el L293 tiene un pin de habilitación que nos puede permitir el ahorro de consumo.

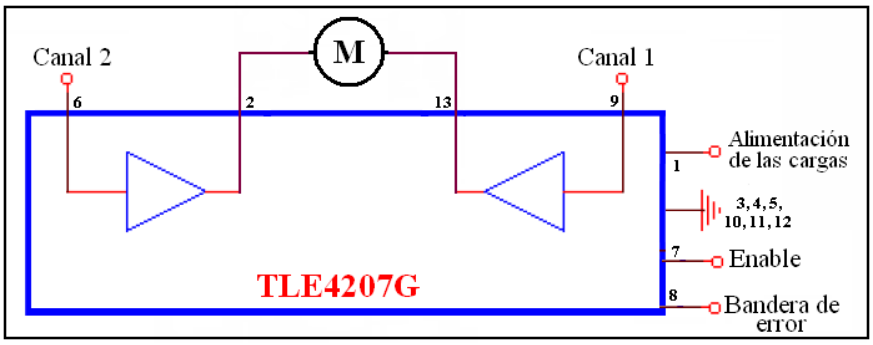


Figura 3-13: Esquema de conexión de un TLE4207G

En la Figura 3-13 se muestra un posible esquema de conexión para el integrado TLE4207G que controla el doble sentido de giro de un motor.

Suponiendo que enable esta a nivel alto y la bandera de error también; cuando el canal uno se pone a nivel alto y el canal dos se pone a nivel bajo, el motor gira en sentido

[16] Para más información acerca del integrado puede acceder al datasheet del fabricante en http://www.infineon.com/cms/en/product/index.html

[17] El pin 8 del TLE4207G que informa de posibles errores es de colector abierto y debe estar polarizado a la alimentación mediante un pull-up de 10 kΩ

antihorario, y cuando el canal uno se pone a nivel bajo y el canal dos se pone a nivel alto el motor uno gira en sentido horario. Si los dos canales tienen simultáneamente el mismo valor, entonces el motor parará rápidamente, por otro lado, si se pone a nivel bajo el pin de habilitación, el motor girará libremente (por inercia) hasta parar.

Este último integrado tiene dos inconvenientes. El primero de ellos es que con un integrado tan solo podría controlarse el doble sentido de giro de un motor, y dado que se necesitan controlar dos motores, sería necesario el uso de dos integrados de este tipo. El segundo inconveniente es que el TLE 4207 G no cumple la directiva RoHS[18].

Ambos inconvenientes han forzado el uso de otro driver para motores, el **TLE 4208 G**, es del mismo fabricante, Infineon Technologies, y tiene las mismas ventajas que el driver analizado anteriormente, pero a diferencia de aquel, este último puede controlar con un solo integrado el doble sentido de giro de dos motores, y además cumple con la normativa RoHs. Es por todo el análisis anterior por lo que como puente H y driver de motores se ha usado el TLE 4208 G cuyo esquema funcional se adjunta en la Figura 3-14.

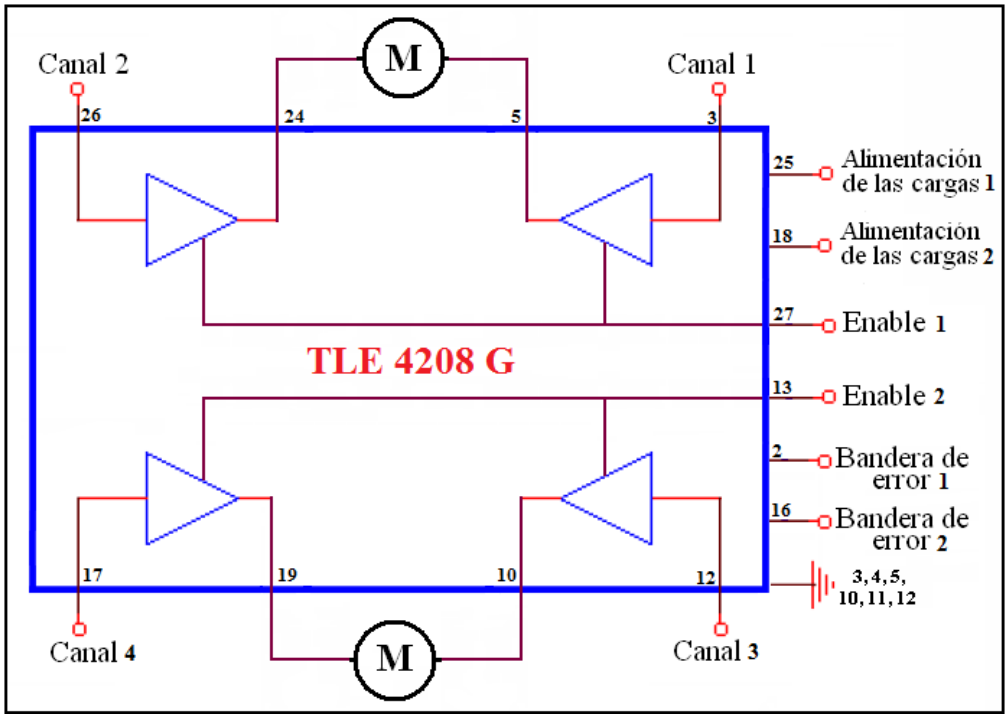


Figura 3-14: Esquema de conexión de un TLE4208G

18 Se trata de la directiva 2050/95/CE de Restricción de ciertas Sustancias Peligrosas en aparatos eléctricos y electrónicos que restringe el uso de plomo, mercurio, cadmio, cromo hexavalente, PBB y PBDE

3.4 Control de velocidad

Gracias al uso de los Puentes en H explicados en el apartado 3.3.1 se consigue que el motor de corriente continua tenga la posibilidad de girar en ambos sentidos, pero si lo que se desea es controlar la velocidad de giro existen dos mecanismos típicos para conseguirlo; bien mediante la reducción de la tensión de alimentación o bien mediante la modulación por ancho de pulso.

3.4.1. Reducción de la tensión de alimentación

La manera más intuitiva de controlar la velocidad de giro de un motor de corriente continua es regulando el voltaje de la fuente de alimentación. De este modo, si se disminuye la tensión entregada al motor, se disminuye también la corriente que circula por él y por tanto se reduce su velocidad de giro, lo contrario ocurrirá si se aumenta la tensión de alimentación (sin rebasar ciertos límites).

Ciertamente se trata de una forma primitiva de conseguir el objetivo, además este tipo de regulación podría no funcionar en algunas situaciones en las que la fuerza que necesite el motor para girar no sea constante; y en el caso de que funcionara, sería muy complicado calcular la tensión que se debe aplicar para girar a una velocidad concreta.

Para poder controlar la velocidad con este método, suele emplearse realimentación sobre el eje del motor; dicho de otro modo, se mide la velocidad a la que gira el eje y en base a eso, se regula la corriente que se aplica al motor. El encargado de medir la velocidad a la que gira el eje del motor se denomina tacómetro.

Existen diversos tipos de tacómetros, con mayor o menor complejidad y que ofrecen más o menos precisión. Uno de los más fáciles de construir y menos preciso es colocar un disco con franjas alternadas de color claro y oscuro, y usar un sistema óptico de lectura, dotado de un transmisor/receptor, que entregue pulsos al haber coincidencia entre las bandas y el fotoreceptor.

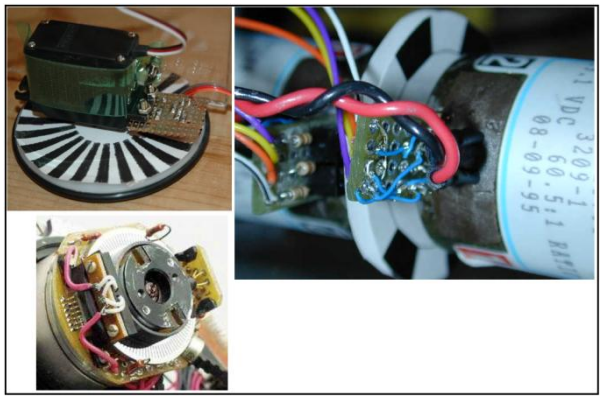


Figura 3-15: Ejemplos de tacómetros ópticos (extraídas de Internet)

En muchas ocasiones lo que se usa es un encoder ó tacómetro de pulsos. Consiste en un sistema capaz de contar digitalmente los pulsos y saber así de forma más precisa la cantidad de vueltas que da el eje de un motor. El inconveniente de los encoder es que requieren un procesamiento mayor de la señal para realizar la realimentación.

3.4.2. Modulación por ancho de pulso, PWM

La técnica de PWM o modulación por ancho de pulso consiste en modificar el ciclo de trabajo de una señal periódica[19] para controlar la cantidad de energía que se envía a un motor y así regular la velocidad de giro de éste[20].

En el método que se propuso en el apartado anterior, al modificar la tensión eléctrica se modificaba también el par motor; pero usando PWM, no se modifica la tensión eléctrica aplicada sobre el motor de modo que el par se mantiene constante.

En algunas ocasiones, lo que se hace es poner una resistencia para disminuir la tensión que le llega al motor, pero se pierde energía en forma de calor, cosa que no ocurre en PWM, que aprovecha toda la energía eléctrica. La Figura 3-16 izquierda muestra una forma rudimentaria de generar una señal PWM.

Cuando se cierra el interruptor, se permite el paso de corriente y aparece una diferencia de tensión en el motor cuyo valor depende de la fuente de alimentación, sin embargo, cuando el interruptor no está pulsado, la diferencia de tensión es nula. Si se alterna la acción de abrir y cerrar el circuito mediante el interruptor y se muestra la diferencia de tensión con respecto al tiempo se consigue generar una onda cuadrada (Figura 3-16 derecha). Esa es la idea básica de una señal PWM.

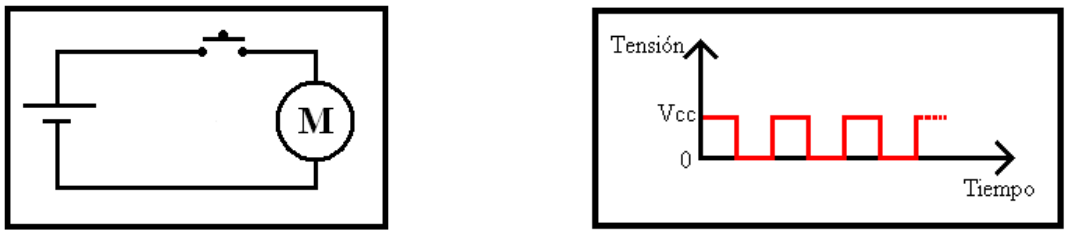


Figura 3-16: Generación manual de PWM (izda.) y señal resultante (dcha.)

19 El ciclo de trabajo o duty cycle, en inglés, es la fracción de tiempo en que una señal está en estado activo alto lógico con respecto a su período y se suele expresar en tanto por ciento.

20 La definición usada para PWM es la relativa a su aplicación para el control de motores, que es el uso que se hace de ella en este PFC. No obstante la técnica de PWM se aplica en numerosas aplicaciones además del control de motores.

En las figura siguiente se muestran señales de PWM con ciclo de trabajo del 90% (Figura 3-17 izquierda), del 75% (Figura 3-17 centro) y del 25% (Figura 3-17 derecha). Obsérvese que la diferencia de potencial media (en verde) que ve el motor es de 0.9Vcc, 0.5Vcc y 0.25Vcc respectivamente, mientras que la diferencia de potencial instantánea (en rojo) sigue siendo de Vcc. Así es como se consigue variar la velocidad de giro sin variar el par motor.

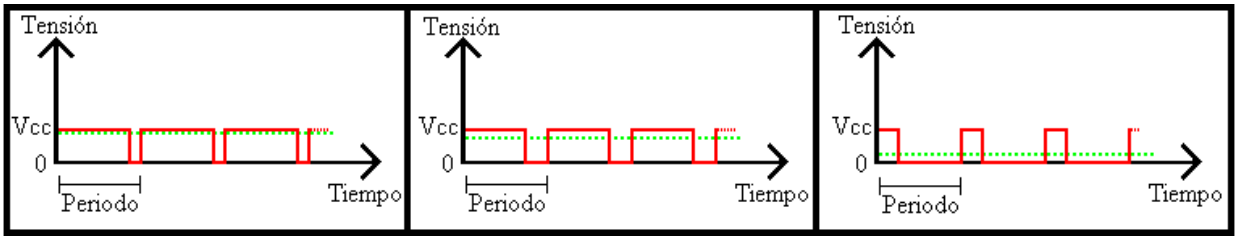


Figura 3-17: Duty cycle del 90% (izquierda), duty cycle del 75% (centro) y duty cycle del 25% (derecha)

La frecuencia de PWM debe ser suficientemente alta como para que: el motor presente un avance continuado (no se mueva de forma pulsante) y no provoque ruido audible. Además, debe ser suficientemente baja como para no presentar ruido eléctrico en el sistema ni pérdidas de eficiencia en el puente en H debidas a la conmutación.

3.5 Características necesarias para el microcontrolador

El módulo de locomoción, además de contener elementos simples como resistencias, condensadores, conectores, etc., contiene un elemento de gran complejidad que se encarga de gestionar todo el módulo, el microcontrolador.

El microcontrolador es el núcleo del módulo y su tarea es interpretar de manera adecuada las órdenes que recibe del módulo maestro por I^2C. Debe convertir dichas órdenes en señales comprensibles para los motores de manera que éstos generen el movimiento que se requirió por parte del maestro.

Para llevar a cabo esta tarea, es necesario que el microcontrolador del módulo de locomoción, además de cumplir los requisitos genéricos definidos en el apartado 2.1 cumpla otra serie de requisitos que a continuación se detallan:

- **Canales de PWM**

Como se explicó en el apartado 3.4.2, para controlar un motor de modo que gire a diferentes velocidades se necesita un canal de PWM, sin embargo, si se quiere que además gire en ambos sentidos se necesita el uso combinado de PWM y un puente H, lo que conlleva dos canales PWM como se muestra en la Figura 3-18.

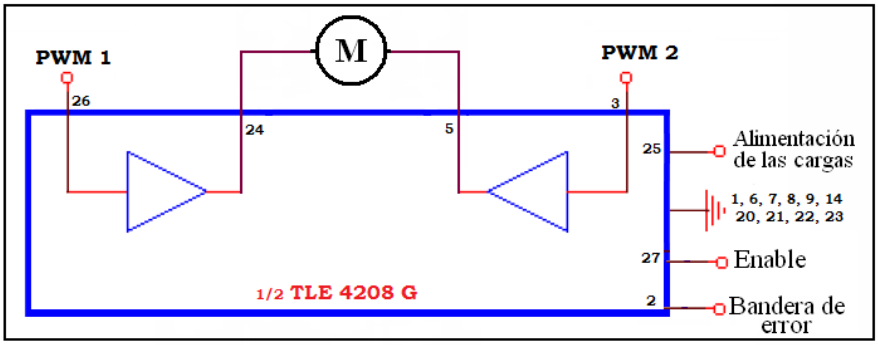


Figura 3-18: Uso de dos canales PWM y un puente en H

Así pues, si lo que se pretende es el control de dos motores, se necesitan un total de cuatro canales de PWM. Si bien es cierto que podría usarse tan solo uno para el control de ambos, no es menos cierto que se debería añadir hardware externo aumentando así no solo la complejidad del diseño sino también el tamaño y el coste del mismo[21]. Es por ello que se ha optado por la opción de no escatimar en canales necesarios y usar cuatro.

[21] En el anexo B de este PFC se presenta una posible opción para hacer el diseño usando tan solo un canal de PWM.

- **Pines de detección de interrupciones**

Los pines de detección de interrupciones son pines que producen una interrupción si ocurre un cambio de nivel en dichos pines. Existen dos tipos de pines de detección de interrupciones, los externos y los internos.

Los pines de detección de interrupciones internos generan una interrupción ante un cambio de nivel del tipo pasar de 0 (bajo lógico) a 1 (alto lógico) o viceversa, mientras los pines de detección externos detectan además de esos cambios, otros del tipo flanco de bajada en un pin o flanco de subida. Normalmente en los microcontroladores suele haber bastantes menos externos que internos.

Para poder saber cuánto ha girado el eje de un motor, se lee la información de su encoder asociado. La información que arroja un encoder es una serie de pulsos por cada vuelta dada, de este modo, contando el número de pulsos proveniente de un encoder podemos saber el número de vueltas que ha dado el eje de un motor. Pues bien, la lectura del encoder se realiza mediante estos pines de detección de interrupciones, de modo que no se necesita estar constantemente contando el número de pulsos sino que se encarga de ello una interrupción hardware. Así, se necesitan, para dos motores, dos pines de detección de interrupciones, sea cual sea su tipo (interno o externo).

- **Frecuencia de operación**

En aplicaciones críticas, como pueden ser las militares, espaciales o médicas, se requiere que el tiempo de respuesta del robot sea realmente pequeño, y que no se le escape ninguna información del exterior por rápido que fuera el evento que ocurrió.

Para dichas aplicaciones se requiere además de una elevada frecuencia de operación un alto número de instrucciones ejecutadas por segundo[(22)]. No obstante, el presente PFC no trata de diseñar un robot para aplicaciones militares y/o médicas sino para aplicaciones de entretenimiento y de trabajo supervisado.

[22] El número de instrucciones ejecutadas por segundo se mide en millones (MIPS) para microprocesadores de 32 o 64 bits; pero para microprocesadores como los que nos ocupan (8 bits) se miden en KIPS (kiloinstrucciones por segundo)

De este modo, la frecuencia no sería algo crítico si no fuera por el hecho de que los canales de PWM tendrán una frecuencia proporcional a la del micro y el motor podría producir un molesto e incesante ruido ante el movimiento del robot.

Se busca pues, una frecuencia tal que nos permita que la frecuencia de PWM sea mayor a la frecuencia audible[23]. Además un factor importante, es que cuanto más elevada sea la frecuencia de operación del PWM menos sufrirán los motores tal como se detalló anteriormente.

3.5.1. Elección del microcontrolador

Para el módulo de locomoción se ha decidido trabajar con el microcontrolador ATtiny2313 de Atmel porque cumple todos los requisitos mencionados anteriormente:

Económico: Cuesta alrededor de 1 euro (al comprar 10 unidades).

Pequeño: El tamaño de los microcontroladores es generalmente proporcional al número de pines de estos, así a menor número de pines menor tamaño del integrado. En este caso dispone de 18 pines de entrada / salida para interactuar con el medio, lo cual hace que el microcontrolador sea a la vez que pequeño funcional. Además está dotado de una memoria flash de 2kbytes, capacidad suficiente para albergar el módulo de locomoción.

Canales de PWM: Como ya se ha comentado, sería bastante recomendable que el microcontrolador tuviera cuatro canales para generar señal PWM; éste dispone de esos cuatro canales de PWM.

Pines de detección de interrupciones: Puede detectar interrupciones de hasta nueve pines distintos, dos de los cuales son capaces de detectar interrupciones externas, ya explicadas.

Frecuencia de operación: Su frecuencia de operación es de hasta 20MHz. con una alimentación de 5V., y por tanto es capaz de generar canales de PWM de casi 40kHz., lo cual está por encima de la frecuencia máxima audible del ser humano (~20kHz.).

[23] Las frecuencias que pueden ser percibidas por un oído sano y joven de un ser humano varían entre 20 Hz y 20 KHz.

Comunicación I^2C: No tiene implementado propiamente el protocolo de I^2C pero se puede emular mediante el uso del protocolo USI.

Programación en lenguaje C: Puede programarse en lenguaje C.

No solapamiento: No existe solapamiento entre los pines que generan la salida PWM y los pines que gestionan la comunicación I^2C. No existe solapamiento entre los pines de detección de interrupciones externas y los pines que generan la salida PWM. Existe solapamiento entre los pines que gestionan la comunicación I^2C y los pines que gestionan la comunicación SPI, pero puesto que ambas comunicaciones no serán simultaneas en condiciones normales de uso, es permisible su coexistencia para la que se han añadido jumpers cuya funcionalidad y detalles se explicarán en el apartado 3.6.

3.6 Diseño hardware

Para explicar con mayor claridad y detalle cada una de las partes hardware que componen el módulo de locomoción se ha representado con ayuda de los programas *Eagle 3D* y *POV Ray* una imagen tridimensional de éste módulo (Figura 3-19).

Aunque en el apartado 3.8.1 se muestra una imagen real del módulo construido, y en el anexo G.1 se muestran tanto el esquemático como el layout, se ha considerado que la imagen en 3D permite una mayor claridad y resolución del hardware que facilita una mejor explicación de cada una de las partes de las que se compone el PCB.

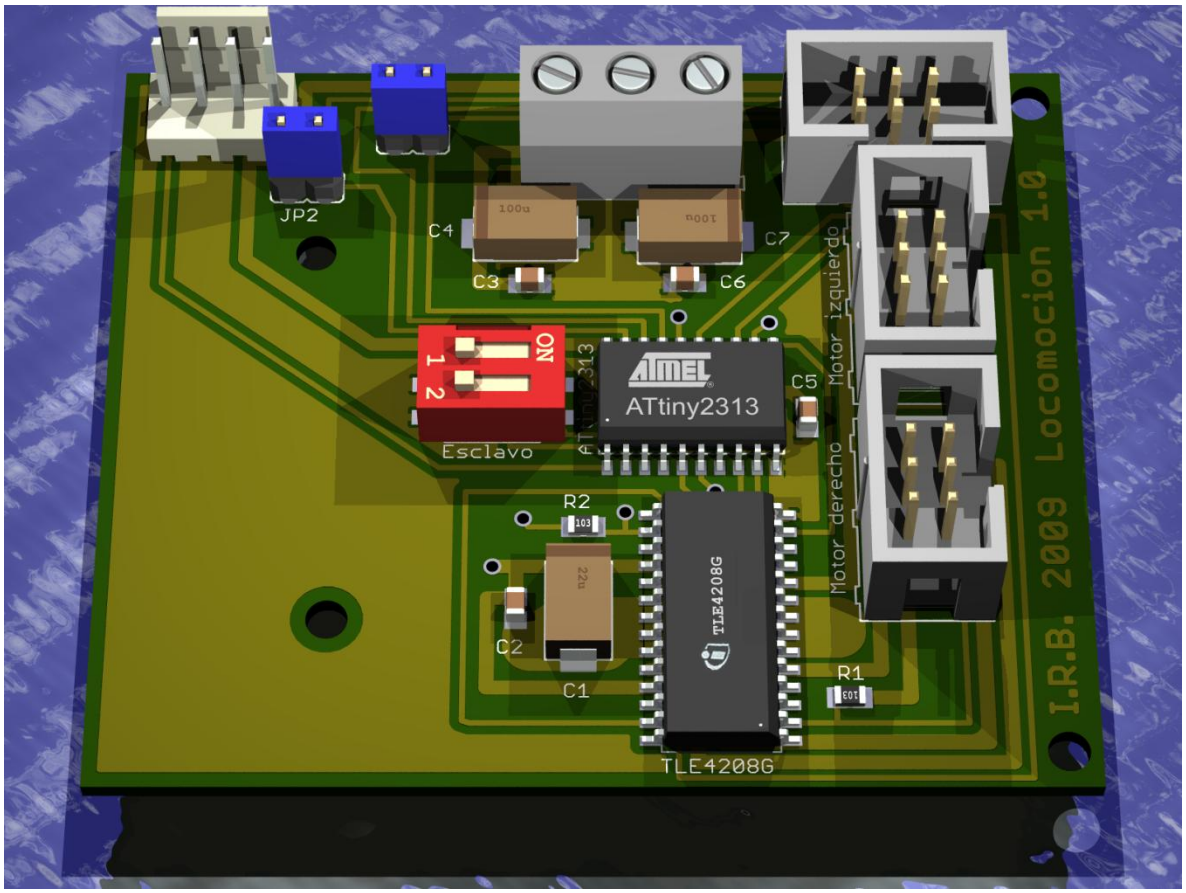


Figura 3-19: Imagen 3D del módulo de locomoción

A continuación, en las páginas siguientes se ampliará y comentará la función de cada una de las partes del módulo representado.

En primer lugar se presenta, en la Figura 3-20, una ampliación de la parte superior en la que se puede observar la circuitería dedicada a la programación y a la comunicación.

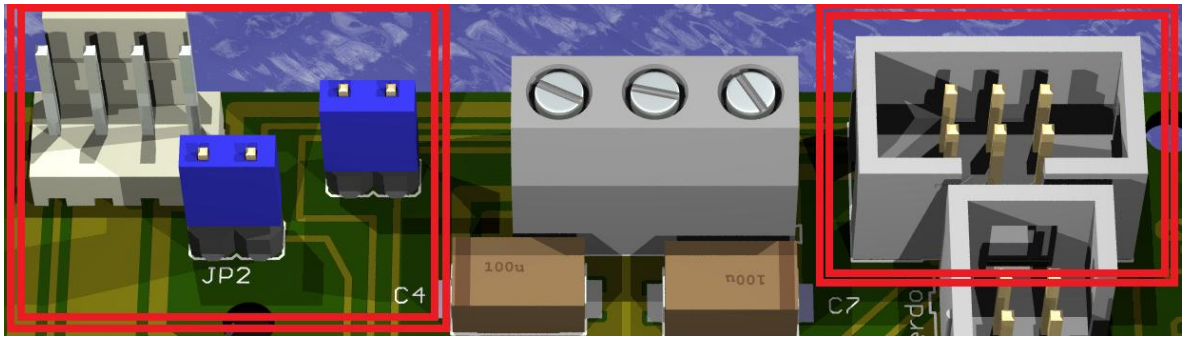


Figura 3-20: SPI e I^{2}C del módulo de locomoción

- Recuadrado en rojo a la derecha aparece el conector encargado de las comunicaciones SPI, protocolo del que se habló en el apartado 2.3 y cuya función es la de establecer comunicación entre el microcontrolador y un ordenador para modificar el firmware del módulo de locomoción. Los pines que lo componen son de arriba abajo y de derecha a izquierda GND, MOSI, +5V., $\overline{RESET}$, SCK y MISO.

- Recuadrado en rojo a la izquierda aparece el conector encargado de las comunicaciones I^{2}C, protocolo del que se hablo en el apartado 2.2 y cuya función es la de establecer comunicación entre el módulo de locomoción y el módulo maestro. Los pines que lo componen son de derecha a izquierda SCL, SDA, GND y $\overline{RESET}$.

  Se ha añadido la señal de $\overline{RESET}$ para posibilitar un reseteo general de todos los módulos de manera simultánea en vez de tener que resetear de manera independiente cada uno de ellos.

  En el microcontrolador ATtiny2313, los pines SDA y SCL del protocolo I^{2}C están solapados con los pines MOSI y SCK del protocolo SPI. Para que este solapamiento no produzca problemas de comunicación, se han añadido los jumpers JP1 y JP2 situados cerca del conector de I^{2}C. Así pues, en condiciones normales de funcionamiento JP1 y JP2 deben estar colocados para que las comunicaciones I^{2}C tengan éxito y deben eliminarse cuando deseen usarse las comunicaciones SPI (actualización del firmware).

En segundo lugar se presenta, en la Figura 3-21, una ampliación de la parte central que alberga la "inteligencia" del módulo.

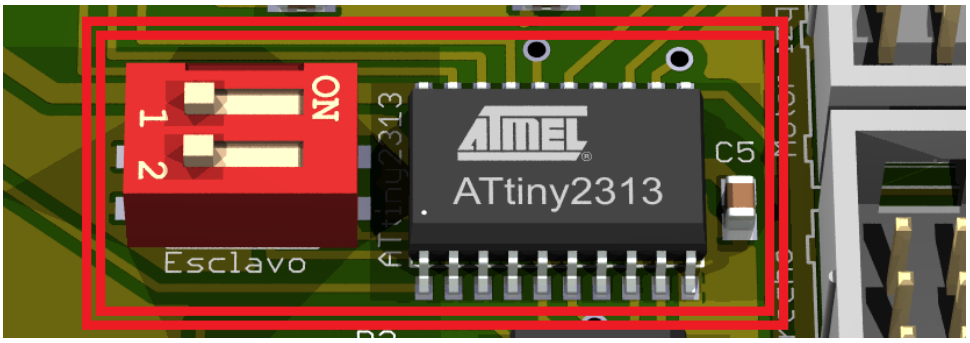


Figura 3-21: Núcleo del módulo de locomoción

- Recuadrado en rojo aparece a la derecha el microcontrolador, en este caso el ATtiny2313 de ATMEL, tal como se justificó en el apartado 3.5.1.

  El microcontrolador es el encargado de recibir las órdenes del maestro por I^2C, interpretarlas y actuar en consecuencia en este caso moviendo uno o dos motores en unos sentidos concretos y a unas velocidades determinadas. Junto a él aparece el condensador de desacoplo C5, de valor 100nF.

- Lo que aparece a la izquierda del microcontrolador es lo que se ha denominado "selector de esclavo" y su función es la de completar la dirección I^2C del módulo.

  Tal como se comentó en el apartado 3.1, los dos últimos bits de la dirección del módulo de locomoción se modifican de acuerdo a la posición de dicho selector, pudiendo obtener de esta manera 4 direcciones diferentes correspondientes a las diferentes posiciones. Si el selector no existiera y quisiéramos manejar 4 motores, pensaríamos en replicar el módulo de locomoción, no obstante, cuando el maestro quisiera mover un solo motor, se movería un motor en cada módulo ya que todos responderían a la misma dirección I^2C; sin embargo, gracias al selector de esclavo se puede dotar a cada PCB de una dirección distinta sin necesidad de cambiar el firmware pudiendo así controlar de manera independiente un total de hasta 8 motores.

  El maestro debe, a la hora de emitir cada orden, especificar el esclavo al que se refiere dentro de un mismo módulo. De modo que una posible orden sería *"mover de manera indefinida el motor derecho del esclavo número dos del módulo de locomoción"*.

En la Tabla 3-1 se muestran la posiciones que deberían tener los switches del selector de esclavo dependiendo del esclavo al que el maestro desee dirigirse.

Número de esclavo	Switch 1	Switch 2	Dirección I^2C
Cero	ON	ON	0x04
Uno	OFF	ON	0x05
Dos	ON	OFF	0x06
Tres	OFF	OFF	0x07

Tabla 3-1: Posiciones de los switches del selector en función del esclavo de locomoción deseado

En tercer lugar se presenta, en la Figura 3-22, una ampliación de la parte inferior en la que se puede observar la etapa de aumento de la tensión para los motores.

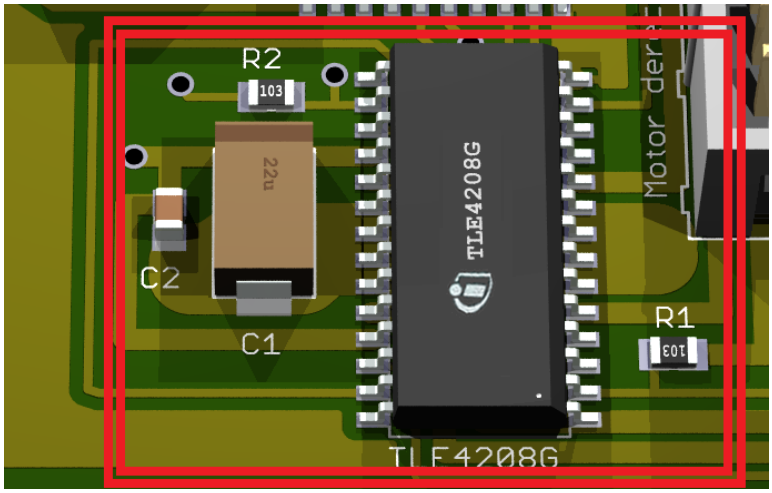


Figura 3-22: Driver y puente H del módulo de locomoción

- Recuadrado en rojo aparece en el centro el puente en H y driver de potencia, en este módulo concreto el integrado TLE4208G de Infineon Technologies que proporciona la tensión y corriente necesarias para los motores que se usan en este PFC y cuyas características ya se mencionaron en el apartado 3.2.1.

  Rodeando al integrado aparece la circuitería necesaria para su correcto funcionamiento. Así pues, dos resistencias de 10kΩ. (R1 y R2) necesarias para los pines de bandera de error (que recordemos eran en colector abierto) y dos condensadores de desacoplo, C1 y C2 de 22µF. y 100nF. respectivamente.

En cuarto lugar se presenta, en la Figura 3-23, una ampliación de la parte lateral en la que se puede observar la etapa de salida del módulo.

- Recuadrado en rojo aparece en la parte inferior el conector para el motor derecho y en la parte superior el motor para el conector izquierdo.

  Ambos conectores son idénticos, de 6 pines cada uno. Están diseñados de manera que encajan con los pines de los motores Faulhaber de los que se habló en el apartado 3.2.1. La disposición de los pines de estos conectores es de arriba debajo y de derecha a izquierda: GND, señal M- del motor, canal 1 encoder, canal 2 encoder, señal M+ del motor, VCC encoder (+5V.).

  No obstante, aunque el conector esta optimizado para esos motores, teniendo en cuenta la disposición de los pines es fácil colocar cualesquiera otros motores de corriente continua que entren dentro de los límites establecidos por el driver TLE4208G que se utiliza en este módulo.

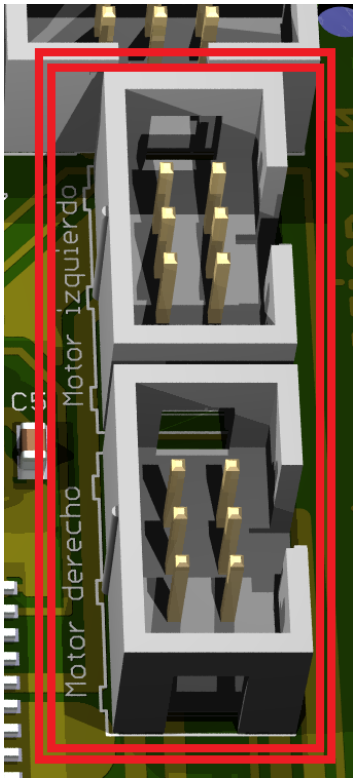


Figura 3-23: Conectores motores

Por último se presenta, en la Figura 3-24, una ampliación de la parte superior central en la que aparece la entrada al módulo.

- Recuadrado en rojo aparece una clema de tres entradas. Se trata de la alimentación del circuito y es por ello que aparecen ahí los condensadores de desacoplo de 100nF. (C3 y C6) y de 100µF. (C4 y C7). Las entradas a la clema son de derecha a izquierda: entrada de 5V., GND y entrada de 9-12V. (según los motores).

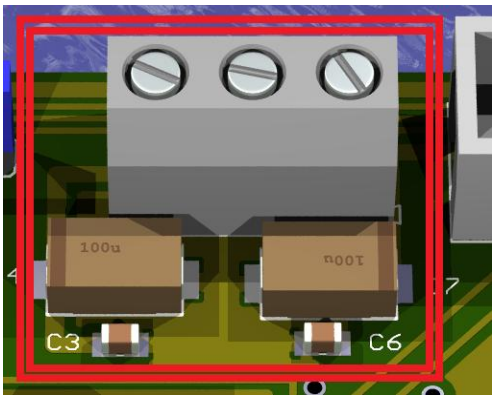


Figura 3-24: Entrada del módulo de locomoción

3.7 Diseño software

En este apartado pretenden mostrarse al lector las funciones que componen el módulo de locomoción. Además del prototipo y de una detallada explicación de cada una de ellas, se explica la manera de ejecutar aquellas que son accesibles desde el exterior.

Recordemos que el módulo de locomoción recibe las órdenes mediante el protocolo de comunicaciones I^2C. De modo que es necesario conocer el orden y la forma de enviar los parámetros de las funciones.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
configurar_pines	void	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del módulo de locomoción, esto hace que sea inaccesible desde el exterior y que por tanto sea el esclavo quien debe hacer una llamada a ésta al inicializarse. En concreto, lo que hace la función es:

1. Habilita como salidas las señales que controlan motores: PB2 (M+ del motor izquierdo; avanzar), PB3 (M- del motor derecho; avanzar), PB4 (M+ del motor derecho; retroceder), y PD5 (M- del motor izquierdo; retroceder). Además configura dichas señales para que generen señales PWM de 31,25kHz.
2. Habilita como salidas las señales que controlan el driver TLE4208G y las pone a nivel alto: PD4 (habilita el motor derecho) y PD6 (habilita el motor izquierdo).
3. Habilita como entradas las señales que provienen del canal A del encoder integrado en el motor Faulhaber: PD2 (encoder motor derecho) y PD3 (encoder motor izquierdo). Se generará una interrupción cada vez que ocurra un flanco de subida en una de esas dos señales para contar los pulsos que emite el encoder.
4. Habilita como entradas las señales que provienen de la bandera de error del driver TLE4208G: PB0 (bandera de error del motor izquierdo) y PB1 (bandera de error del motor derecho). Además establece que se genere una interrupción cada vez que ocurre un cambio de nivel en una de esas dos señales, pudiendo, de este modo, deshabilitar los canales del driver que controlan ese motor evitando sobretensión o temperaturas demasiado altas. Tanto PB0 como PB1 están a nivel alto en condiciones normales.
5. Habilita como entradas las señales que completan la dirección I^2C del módulo, y además activa el pull-up interno: PD0 (da valor al último bit de la dirección) y PD1 (da valor al penúltimo bit de la dirección).

Finalmente, habilita las interrupciones generales.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
avanza_distancia	uint8_t movimiento uint8_t porcentaje uint8_t radio uint8_t motor uint16_t distancia	void	Condición de START Escribir en el esclavo 000001xx Transmitir el comando 0xA1 Transmitir el dato movimiento Transmitir el dato porcentaje Transmitir el dato radio Transmitir el dato motor Transmite MSB de distancia Transmite LSB de distancia Condición de STOP

La tarea de esta función es hacer que la plataforma robótica avance una distancia.

El parámetro movimiento puede tomar el valor cero o el valor uno. En el caso de tomar el valor cero, el movimiento será de avance y en el caso de tomar el valor uno el movimiento será de retroceso. Así pues, uno o ambos motores girarán en sentido horario o antihorario dependiendo del valor 0 ó 1 del parámetro movimiento.

El parámetro porcentaje puede tomar un valor entre cero y cien. Este parámetro indica la velocidad relativa de los motores, así, 100 representa la velocidad máxima, 50 representa el 50% de la máxima velocidad y 0 representa parar el motor. En el caso de que el parámetro tome valor cero, es decir si se desea parar el motor o motores, no importa el valor del parámetro movimiento.

El parámetro radio puede tomar un valor entre 1 y 255. Este parámetro hace referencia al radio de la circunferencia de giro de los motores que soportan la plataforma robótica. El parámetro va indicado en mm. por lo que no podrán usarse ruedas con un radio exterior mayor a 25,5 cm.

El parámetro motor puede tomar el valor cero, uno o dos. Este parámetro indica el motor o motores sobre el que se desea actuar. Así pues, el movimiento de avance o retroceso a una velocidad determinada se aplicará sobre el motor derecho (si el valor de motor es cero), sobre el motor izquierdo (si el valor de motor es uno), o sobre ambos motores (si el valor de motor es dos).

El parámetro distancia puede tomar un valor entre 0 y 65535. Dicho valor indica en mm la distancia lineal que se desea que recorran uno o ambos motores. La distancia lineal que recorren los motores coincidirá con la distancia que avanza la plataforma robótica únicamente si se mueven ambos. La distancia va expresada en mm.

Por ejemplo, para que una plataforma robótica con ruedas de 19,2 cm. avance 4 metros y 34,6 cm a un 87% de su velocidad máxima habría que ejecutar *avanzar_distancia (0, 87, 192, 2, 4346)*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
girar_angulo	uint8_t porcentaje uint8_t angulo uint8_t sentido uint8_t radio uint8_t ancho	void	Condición de START Escribir en el esclavo 000001xx Transmitir el comando 0xA2 Transmitir el dato porcentaje Transmitir el dato angulo Transmitir el dato sentido Transmitir el dato radio Transmitir el dato ancho Condición de STOP

La tarea de esta función es generar un giro en la plataforma robótica basándose en algunos parámetros de referencia pasados como argumentos.

El parámetro porcentaje puede tomar un valor entre uno y cien. Este parámetro indica la velocidad relativa con que se producirá el giro de la plataforma, así, 100 representa la velocidad máxima y 50 representa el 50% de la máxima velocidad.

El parámetro ángulo puede tomar un valor entre 1 y 180. El valor de este parámetro indica el ángulo, en grados, que se desea girar la plataforma robótica. Está limitado a 180º, no obstante, es posible girar cualquier ángulo usando este parámetro en combinación con el parámetro sentido.

El parámetro sentido puede tomar el valor cero o uno. Este parámetro indica el sentido hacia el que la plataforma robótica realizará el giro. Así pues, si se desea que el giro se realice hacia la izquierda, el parámetro sentido debe tomar el valor cero, mientras que si se desea girar a la derecha, debemos poner el valor uno en el parámetro sentido. Obsérvese que si se desea girar un ángulo de, por ejemplo, 230º a la derecha, basta con girar 130º hacia la izquierda.

El parámetro radio puede tomar un valor entre 1 y 255. Este parámetro hace referencia al radio de la circunferencia de giro de los motores que soportan la plataforma robótica. El parámetro va indicado en mm. por lo que no podrán usarse ruedas con un radio exterior mayor a 25,5 cm.

El parámetro ancho puede tomar un valor entre 1 y 255. Este parámetro hace referencia a la anchura de la plataforma robótica. El parámetro va indicado en cm. por lo que no podrán usarse plataformas robóticas con una anchura superior a 2,55 m.

Por ejemplo, para girar 29º a la derecha a un 90% de la velocidad máxima una plataforma robótica de 19 cm. de anchura soportada por unas ruedas de 12,3 cm. habría que ejecutar *girar_angulo (90, 29, 1, 123, 19)*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
calcula_distancia	uint8_t motor uint8_t radio	uint16_t	Condición de START Escribir en el esclavo 000001xx Transmitir el comando 0xA3 Transmitir el dato motor Transmitir el dato radio Condición de REPEATED START Leer del esclavo 000001xx Recibe MSB de distancia Recibe LSB de distancia Condición de STOP

La tarea de esta función es calcular la distancia lineal que ha recorrido uno de los motores o ambos desde la última orden ejecutada.

El parámetro motor puede tomar el valor cero, uno o dos. Este parámetro indica el motor o motores sobre el que se desea obtener la información. Así pues, si desea conocerse la distancia recorrida por el motor derecho, el valor de motor debe ser cero; si desea conocerse la distancia recorrida por el motor izquierdo el valor de motor debe ser uno; si desea conocerse la distancia recorrida por ambos motores el valor de motor debe ser dos.

El parámetro radio puede tomar un valor entre 1 y 255. Este parámetro hace referencia al radio de la circunferencia de giro de los motores que soportan la plataforma robótica. El parámetro va indicado en mm. por lo que no podrán usarse ruedas con un radio exterior mayor a 25,5 cm.

La función retornará un valor entre 0 y 65535. Dicho valor indica en mm la distancia recorrida por la plataforma robótica. Así el máximo valor calculado no superará en ningún caso 65,5350 m.

Por ejemplo, para averiguar la distancia lineal que se ha desplazado el motor izquierdo al realizar un giro de unos 90º con una rueda de 13,9 cm de radio exterior habría que ejecutar *distancia_lineal = calcula_distancia (1, 139)*

Nota: Previamente habría que haber ejecutado la instrucción correspondiente para la realización del giro mencionado.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
mover_motor	uint8_t movimiento uint8_t porcentaje uint8_t motor	void	Condición de START Escribir en el esclavo 000001xx Transmitir el comando 0xA4 Transmitir el dato movimiento Transmitir el dato porcentaje Transmitir el dato motor Condición de STOP

La tarea de esta función es generar el movimiento en uno o varios motores a una velocidad y en un sentido determinado por un período de tiempo indefinido; para ello hace uso de una serie de parámetros.

El parámetro movimiento puede tomar el valor cero o el valor uno. En el caso de tomar el valor cero, el movimiento será de avance y en el caso de tomar el valor uno el movimiento será de retroceso. Así pues, uno o ambos motores girarán en sentido horario o antihorario dependiendo del valor 0 ó 1 del parámetro movimiento.

El parámetro porcentaje puede tomar un valor entre cero y cien. Este parámetro indica la velocidad relativa de los motores, así, 100 representa la velocidad máxima, 50 representa el 50% de la máxima velocidad y 0 representa parar el motor. En el caso de que el parámetro tome valor cero, es decir si se desea parar el motor o motores, no importa el valor del parámetro movimiento.

El parámetro motor puede tomar el valor cero, uno o dos. Este parámetro indica el motor o motores sobre el que se desea actuar. Así pues, el movimiento de avance o retroceso a una velocidad determinada se aplicará sobre el motor derecho (si el valor de motor es cero), sobre el motor izquierdo (si el valor de motor es uno), o sobre ambos motores (si el valor de motor es dos).

Por ejemplo, para mover el motor izquierdo indefinidamente en sentido horario a un 73% de su velocidad máxima habría que ejecutar *mover_motor(0, 73, 1)*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
en_movimento	void	uint8_t	Condición de START Escribir en el esclavo 000001xx Transmitir el comando 0xA5 Condición de REPEATED START Leer del esclavo 000001xx Recibe dato Condición de STOP

La tarea de esta función es la de comprobar si los motores están o no en movimiento, y en caso de estarlo especificar cual o cuales están girando.

Se pueden retornar cuatro posibles valores:

1. Si ambos motores está girando, se retornará el valor 0x03
2. Si el motor derecho está girando, se retornará el valor 0x01
3. Si el motor izquierdo está girando, se retornará el valor 0x02
4. Si ninguno de los dos motores está girando, se retornará el valor 0x00

Por ejemplo, para esperar a que el motor derecho este parado podría ejecutarse la instrucción *while (en_movimiento == 0x01);*

Nombre de la interrupción	Protocolo por parte del maestro
ISR (INT0_vect)	El maestro no puede hacer una llamada a una interrupción propia del esclavo

Se trata de una interrupción propia del módulo de locomoción, esto hace que sea inaccesible desde el exterior.

La interrupción será ejecutada cuando se detecte un flanco de subida en el pin PD2. Su tarea es incrementar el contador que gestiona los pulsos enviados por el encoder del motor derecho con el objetivo de saber cuánto ha girado el eje del motor.

Si hubo una llamada a la función avanza_distancia, entonces será esta interrupción la que se encargue de parar el motor derecho cuando este haya avanzado la distancia deseada.

Nombre de la interrupción	Protocolo por parte del maestro
ISR (INT1_vect)	El maestro no puede hacer una llamada a una interrupción propia del esclavo

Se trata de una interrupción propia del módulo de locomoción, esto hace que sea inaccesible desde el exterior.

La interrupción será ejecutada cuando se detecte un flanco de subida en el pin PD3. Su tarea es incrementar el contador que gestiona los pulsos enviados por el encoder del motor izquierdo con el objetivo de saber cuánto ha girado el eje del motor.

Si hubo una llamada a la función avanza_distancia, entonces será esta interrupción la que se encargue de parar el motor izquierdo cuando este haya avanzado la distancia deseada.

Nombre de la interrupción	Protocolo por parte del maestro
ISR (PCINT_vect)	El maestro no puede hacer una llamada a una interrupción propia del esclavo

Se trata de una interrupción propia del módulo de locomoción, esto hace que sea inaccesible desde el exterior.

La interrupción será ejecutada cuando se detecte un cambio de nivel en los valores de PB0 o PB1. Si alguno de dichos pines tomará el valor cero, el driver estaría informando de un error de sobretensión o sobretemperatura en el motor izquierdo o el motor derecho respectivamente.

En caso de error, la interrupción actúa deshabilitando los canales del driver asociados al motor que produce el fallo.

Una vez detalladas cada una de las funciones, se procede a mostrar en la Figura 3-25 el diagrama de flujo de la función principal del módulo esclavo de locomoción. En él puede verse tanto el modo en que se reciben datos por las comunicaciones I^2C como el modo en que se realizan las diversas llamadas a las funciones correspondientes.

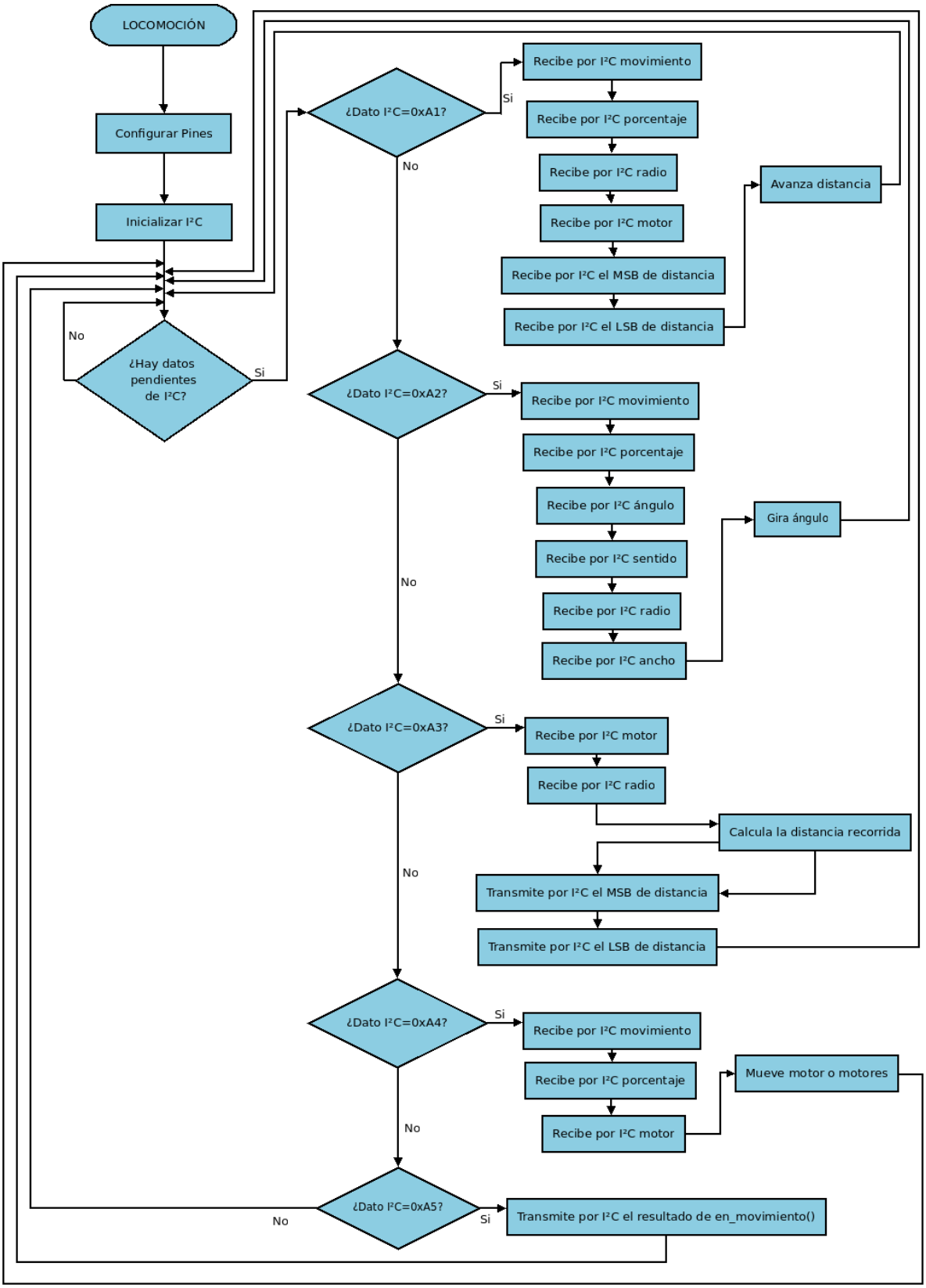


Figura 3-25: Diagrama de flujo del programa principal de locomoción

3.8 Construcción e información técnica

3.8.1. Fase de prototipado del módulo

Antes de construir de manera definitiva el módulo de locomoción, se hizo un prototipo de éste (Figura 3-26), con el objetivo no solo de encontrar posibles errores y corregir problemas de diseño, sino también con el objetivo de probar algunos drivers de motores antes de usar de manera definitiva el TLE-4208-G del que se habló en el apartado 3.3.2. En el prototipo se usaron dos drivers distintos, uno para cada motor, para el motor derecho se hizo uso del driver TLE-4207-G y para el motor izquierdo se usó el UCC37324; ambos analizados en el apartado 3.3.2. Ambos componentes eran SMD de manera que hubo que buscar la forma de interconectarlos con el resto del circuito, por ello, el driver UCC37324 aparece rodeado por pegamento, para asegurar de ese modo la fiabilidad de las soldaduras al cable amarillo que lo une al resto del circuito. Aunque no resultaba muy agradable se concluyó que era aceptable por estar en la fase de prototipado. El cableado está realizado por la parte posterior de la placa usando la técnica de wrapping(18).

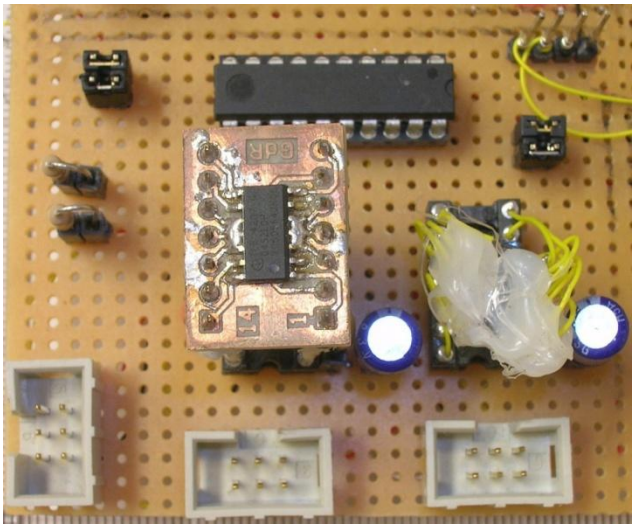


Figura 3-26: Prototipado del módulo de locomoción

3.8.2. Especificaciones, aspecto y presupuesto del módulo

En este apartado se muestra en primer lugar información acerca de las especificaciones técnicas del módulo, necesarias para saber por ejemplo la resolución de avance o la corriente máxima que consume el módulo; dicha información se aprecia en la Tabla 3-2.

Especificaciones técnicas del módulo de locomoción	
Tensión necesaria	5V. para lógica y entre 9V. y 12V. para los motores
Corriente máxima lógica	125,84mA. (con los motores Faulhaber alimentados a 9,5V)
RPM máxima	131 (con los motores Faulhaber alimentados a 9,5V)
Rango de trabajo para avance	Desde 3mm. hasta 65.536mm.
Resolución de avance	1mm.
Error máximo en avance	1cm.
Rango de trabajo para giro	Desde 2^{o} hasta 180^{o} (es posible seleccionar sentido de giro)
Resolución de giro	1^{o}
Error máximo en giro	5^{o}
Dimensiones	55,2450mm. de alto y 66,6750mm. de ancho (36,8346cm^{2}.)

Tabla 3-2: Especificaciones técnicas del módulo de locomoción

En segundo lugar se muestra, en la Figura 3-27, el módulo de locomoción una vez construido en su versión definitiva.

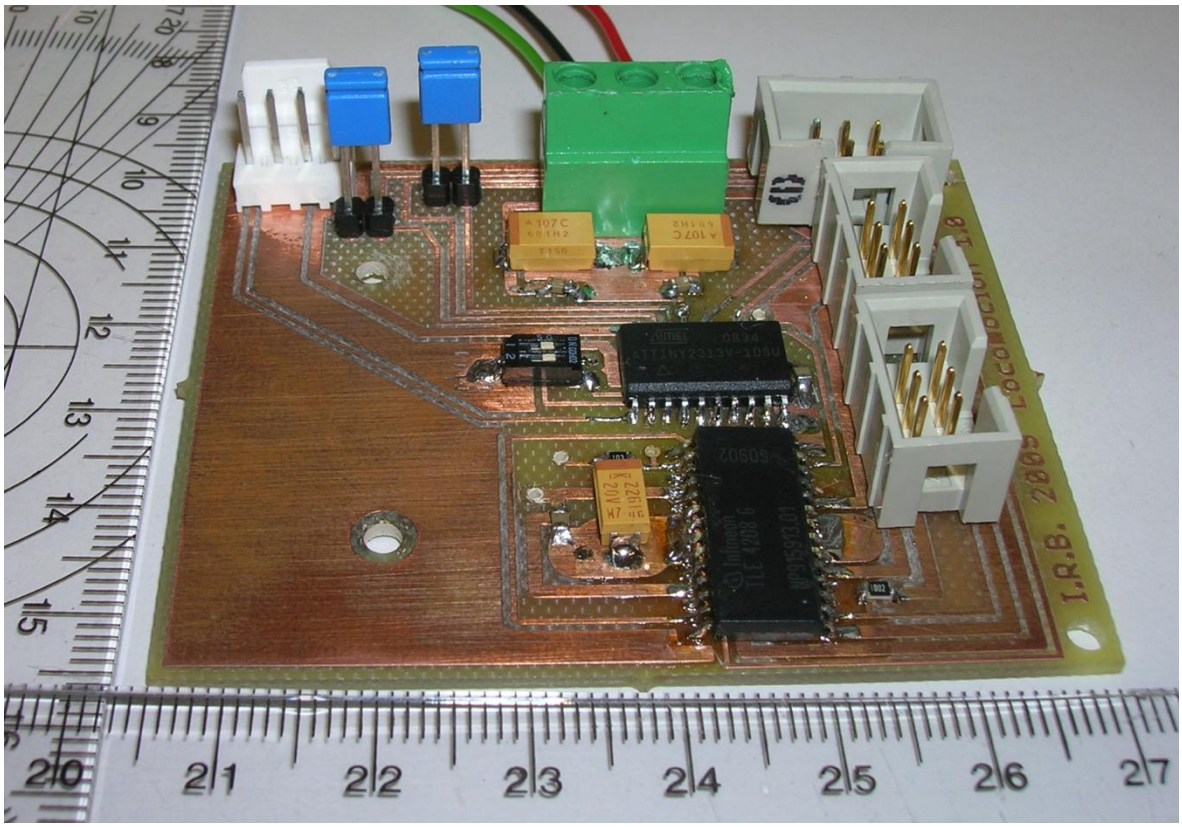


Figura 3-27: Imagen real del módulo de locomoción

En tercer lugar se hace constar el precio que costaría construir el módulo de nuevo. En la Tabla 3-3, se muestra el precio de cada componente necesario para realizar el montaje del módulo de locomoción.

Los precios han sido extraídos de Merchan electrónica y componentes[24], exceptuando algunos de ellos, que por no encontrarse en el distribuidor anterior se han consultado en Farnell[25], estos últimos están en color azul. Los que no se han encontrado en ninguno de los dos anteriores han sido consultados en RS[26], en este caso, han sido coloreados en naranja.

Referencia	Encapsulado	Componente	Valor	Precio
C2, C3, C5, C6	0805	Condensador	100nF	4x0.0116€
C1	7343	Condensador	22µF	1x1.257€
C4, C7	7343	Condensador	100µF	2x1.32€
R1, R2	0805	Resistencia	10kΩ	2x0.008€
ATtiny2313	SO20	Microcontrolador	ATtiny2313	1x1.92€
TLE4208G	DIL	Driver	TLE4208G	1x4.61€
JP1…2		Jumper	Tira min. 4 pines	1x0.22€
Motor izquierdo		Conector	Conector Motor	1x1.02€
Motor derecho		Conector	Conector Motor	1x1.02€
CONECTOR SPI		Conector	SPI	1x1.02€
J1	Molex 2,54mm	Conector	I^2C	1x0.479€
Esclavo	DIP (DIL)	Switch SMD	Selector de esclavo	1x1.28€
X1	Pasante 3 vías	Clema 5mm	Bornes para PCB	1x0.4€
Total				**15,9284€**

Tabla 3-3: Precio de los componentes del módulo de locomoción

Teniendo en cuenta que construir cada $dm.^2$ de PCB cuesta 60 €. Se concluye que al coste de los componentes hay que sumar 60€ por construcción.

[24] Los precios están sujetos a modificaciones y deben comprobarse en http://www.e-merchan.com/

[25] Los precios están sujetos a modificaciones y deben comprobarse en http://es.farnell.com/

[26] Los precios están sujetos a modificaciones y deben comprobarse en http://es.rs-online.com/

4 Módulo US&IR

4.1 Motivación del módulo

Una de las funcionalidades de la que se ha querido dotar a la plataforma robótica ha sido la capacidad para detectar la distancia a la que se encuentra un obstáculo. Es deseable que el sistema sea capaz de determinar la distancia a la que se encuentran los obstáculos con el fin de conocer y controlar más el entorno.

Y esa es precisamente la finalidad del denominado módulo US&IR. Pretende ser un módulo capaz de calcular la distancia a la que se encuentra el obstáculo más cercano al propio módulo. La distancia se calculará mediante sensores de ultrasonidos, mediante sensores de infrarrojos o mediante ambos, dependiendo de las órdenes de otro módulo, al que se denominará módulo maestro.

Al igual que ocurría con el módulo de locomoción, las órdenes del módulo maestro serán recibidas y correctamente interpretadas por el módulo US&IR gracias al bus de comunicaciones I^2C que interconecta la plataforma robótica detallado en el apartado 2.2.

Dado que todo el sistema esta interconectado, para que el módulo US&IR tan solo ejecute las órdenes que van dirigidas a él, es necesario identificarlo de algún modo. Para ello se usan las direcciones I^2C. La dirección I^2C grabada en el firmware del módulo US&IR es la 0x08, no obstante, los dos últimos bits de dicha dirección son configurables mediante hardware. De este modo, las posibles direcciones I^2C del módulo de locomoción son 0x08, 0x09, 0x0A y 0x0B. La razón de permitir estas cuatro direcciones, es que en el mismo sistema puedan coexistir un total de hasta cuatro clones de éste módulo.

Gracias al hecho de poder configurar por hardware hasta cuatro direcciones I^2C, el sistema es capaz de manejar hasta dieciséis sensores de ultrasonidos y veintiocho sensores de infrarrojos de forma independiente (ya que cada módulo US&IR maneja cuatro sensores de ultrasonidos y siete sensores de infrarrojos[27]. La configuración hardware de la dirección del módulo se explica en el apartado 4.4.

[27] No pueden manejarse simultáneamente los cuatro sensores de ultrasonidos y los siete sensores de infrarrojos, ya que existen algunas incompatibilidades, que se detallan en el apartado de funcionalidad 4.2

4.2 Funcionalidad

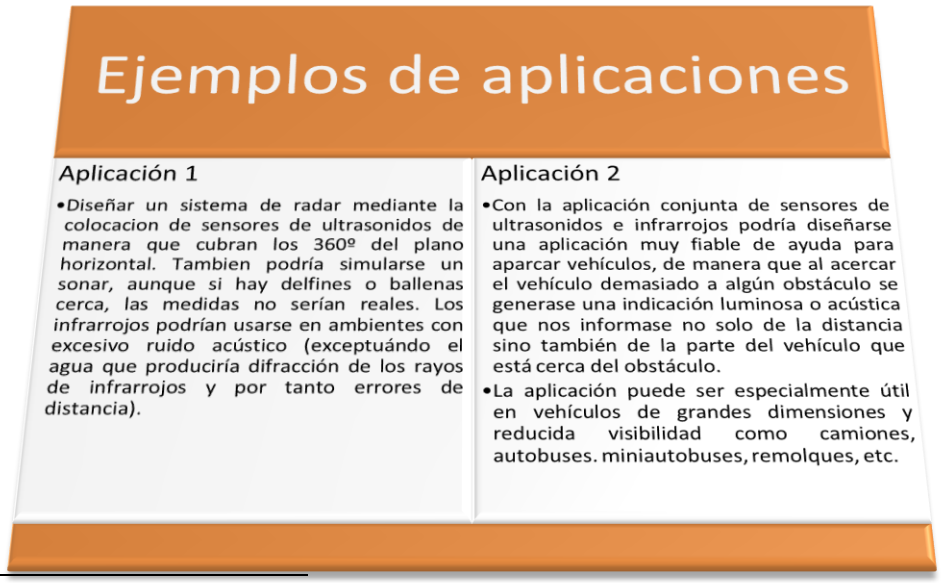
Como se dijo en el apartado 4.1, el módulo US&IR es capaz de interpretar una serie de órdenes recibidas por I^2C. En este apartado pretenden darse a conocer todas las órdenes que es capaz de interpretar el módulo.

1. Calcular la distancia al obstáculo más cercano mediante uno o varios de los cuatro sensores de ultrasonidos que es posible colocar en el módulo.
2. Calcular la distancia al obstáculo más cercano mediante uno o varios de los siete sensores de infrarrojos que es posible colocar en el módulo.

Para poder calcular la distancia desde una gran cantidad de sensores, se ha usado la técnica de la multiplexación de pines; así un mismo pin puede pertenecer a un sensor de ultrasonidos y a un sensor de infrarrojos. Esto trae un inconveniente y es que existen ciertas combinaciones de sensores de ultrasonidos e infrarrojos que no se pueden dar, y son las mostradas a continuación:

1) Sensor de ultrasonidos 1 incompatible[28] con sensores 1 y 4 de infrarrojos.
2) Sensor de ultrasonidos 2 incompatible con sensores 2 y 5 de infrarrojos.
3) Sensor de ultrasonidos 3 incompatible con sensores 3 y 6 de infrarrojos.
4) Sensor de ultrasonidos 4 incompatible con sensores 7 de infrarrojos.

Combinando de manera adecuada las órdenes anteriores, se pueden realizar acciones complejas e interesantes tales como las que se muestran a continuación.



[28] En este contexto, incompatible quiere decir que no pueden estar colocados físicamente a la vez.

4.2.1. Sensores de ultrasonidos SRF04 / SRF05

Para calcular la distancia por ultrasonidos se han empleado, por su muy extendido uso y popularidad, los sensores comerciales SRF04 y SRF05 indistintamente (Figura 4-1).

El módulo US&IR está preparado para usar cualquiera de los dos modelos.

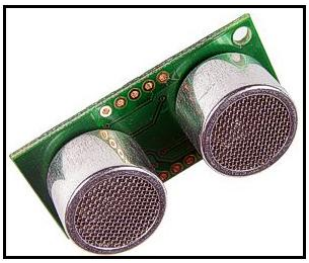


Figura 4-1: Vista frontal SRF04/05

Ambos sensores contienen toda la electrónica encargada de hacer la medición y tienen un funcionamiento muy sencillo. Puede verse una imagen de cada uno de los sensores en la Figura 4-2.

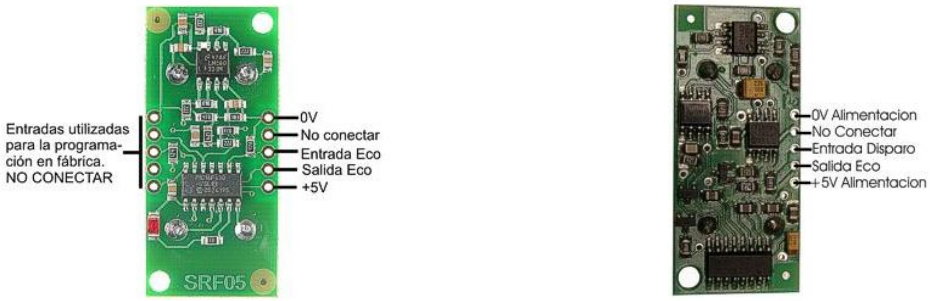


Figura 4-2: Vista trasera del SRF05 (izda.) y vista trasera del SRF04 (dcha.)

En primer lugar el microcontrolador debe emitir un pulso de anchura mínima de 10µs. por el pin "Disparo" del sensor (Entrada Eco en el SRF05 y Entrada Disparo en el SRF04).

A continuación se emite una ráfaga de ocho impulsos ultrasónicos a una frecuencia de 40kHz., dicha ráfaga choca contra el objeto y rebota siendo capturada de nuevo por el sensor.

Finalmente, por el pin "Salida Eco" del sensor aparece un pulso cuya duración es proporcional a la distancia entre el sensor y el objeto.

Puede realizarse de nuevo todo el proceso para tomar otra medida de la distancia teniendo en cuenta que hay que dejar pasar un tiempo mínimo dependiendo de si se usa uno u otro modelo para que el circuito se estabilice y la señal de eco desaparezca completamente.

A continuación, en la Figura 4-3, se muestra un diagrama de tiempos del funcionamiento extraído de la página del fabricante:

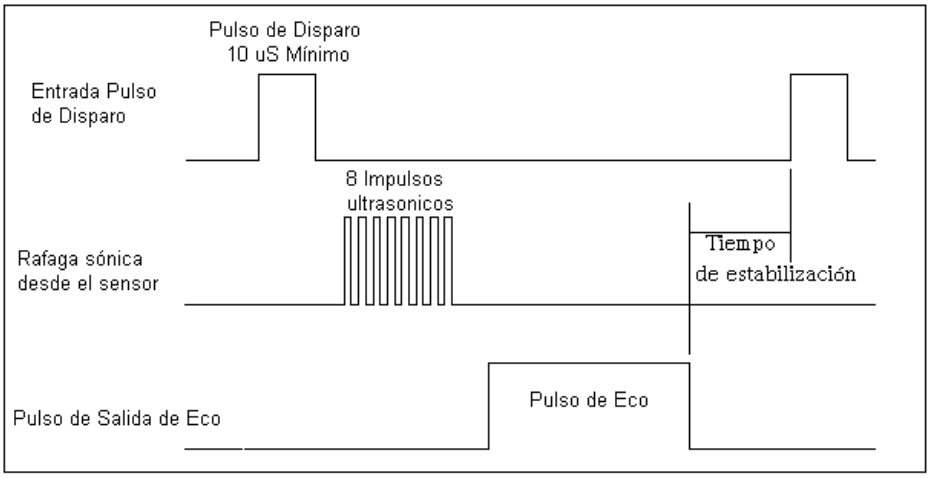


Figura 4-3: Diagrama de tiempos SRF04 / SRF05

Dependiendo del modelo que se use existen algunas particularidades que se muestran en la Tabla 4-1.

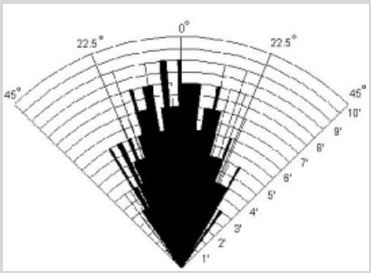
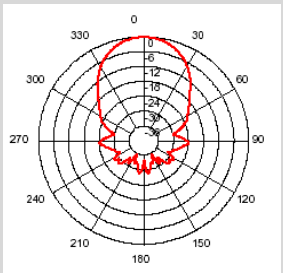
	SRF04	SRF05
Rango de trabajo	De 3 cm. a 3 m.	De 3 cm. a 4 m.
Rango de anchura del eco	De 100µs. a 18ms.	De 100µs. a 25 ms.
Espera entre fin de eco y disparo	10ms.	50ms.
Ancho del eco si no detecta objeto	36ms.	30ms.
Anchura del haz		

Tabla 4-1: Comparativa entre sensores de ultrasonidos SRF04 y SRF05

Como se dijo anteriormente, la anchura del pulso de eco es proporcional a la distancia entre el sensor y el obstáculo. A continuación se pretende calcular ese factor de proporcionalidad.

Sea χ la distancia entre el sensor de ultrasonidos y el objeto más cercano.

Dado que la velocidad, v, se define como el espacio recorrido por unidad de tiempo, τ, el tiempo que tardará la ráfaga ultrasónica en hacer el recorrido sensor - obstáculo es:

$$\boxed{\tau = \frac{\chi}{v}} \qquad [4.1]$$

De modo que en el recorrido de ida y vuelta la ráfaga ultrasónica tardará:

$$\boxed{\tau = \frac{2\,\chi}{v}} \qquad [4.2]$$

La velocidad a la que se transmite la ráfaga, v, coincide con la velocidad del sonido[29], así, el tiempo τ que tardará en recorrer el camino de ida y vuelta resulta:

$$\boxed{\tau \approx \frac{2\,\chi}{0.3442}} \qquad [4.3]$$

Finalmente, despejando, tenemos que el espacio entre el obstáculo y el sensor es:

$$\boxed{\chi \approx 0.1721\tau} \qquad [4.4]$$

Se concluye por tanto que el factor de proporcionalidad entre la distancia a medir en mm. y la anchura del pulso de eco en µs. es aproximadamente 0.1721.

Para conseguir compatibilidad entre los sensores SRF04 y SRF05 se mantiene una espera de 50ms. entre la finalización del eco actual y el comienzo del disparo siguiente y se considera que no hay obstáculo si la anchura del pulso de eco es superior a 30ms.

4.2.2. Sensores de infrarrojos SHARP GP2D12

Para calcular la distancia por infrarrojos se ha empleado el sensor comercial GP2D12 de Sharp (Figura 4-4).

En la parte frontal del sensor se pueden distinguir dos lentes. En la parte interior de la lente de la derecha (según la Figura 4-4) hay un LED infrarrojo. En la parte interior de la

[29] En el aire, a 0 °C, el sonido viaja a una velocidad de 331 m/s y si sube en 1 °C la temperatura, la velocidad del sonido aumenta en 0,6 m/s así, tomando como referencia 22 ºC se tiene una velocidad del sonido de 344.2m/s, es decir, teóricamente 0.3442 mm/µs.

lente de la izquierda (según la Figura 4-4) hay lo que se conoce como PSD[30] o sensor detector de posición.

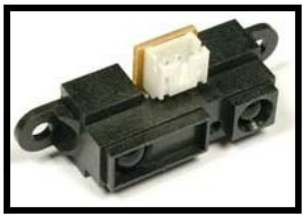


Figura 4-4: Vista frontal GP2D12

El sensor dispone de tres pines; uno de los pines es el de alimentación (4.5V. ~ 5.5V.), otro de los pines es el de tierra (0V.) y el tercero de los pines es el de salida (V_0).

Según el fabricante es necesario situar un condensador de desacoplo entre el pin de alimentación y el pin de masa de valor 10µF. como mínimo para estabilizar la tensión de referencia del sensor. En la página http://www.robotroom.com/DistanceSensor3.html puede verse la influencia de colocar condensadores de diferentes valores.

Las medidas de distancia que pueden tomarse con el GP2D12 varían de 10cm. a 80cm y se determinan en base a la tensión analógica que entrega el pin de salida.

A continuación se muestran dos gráficas, una que ilustra la tensión analógica del pin de salida en función de la distancia a que se encuentra el objeto (Figura 4-5 izquierda extraída de (19)) y otra que ilustra la sensibilidad del sensor con respecto a la temperatura para diferentes distancias (Figura 4-5 derecha extraída de (20)).

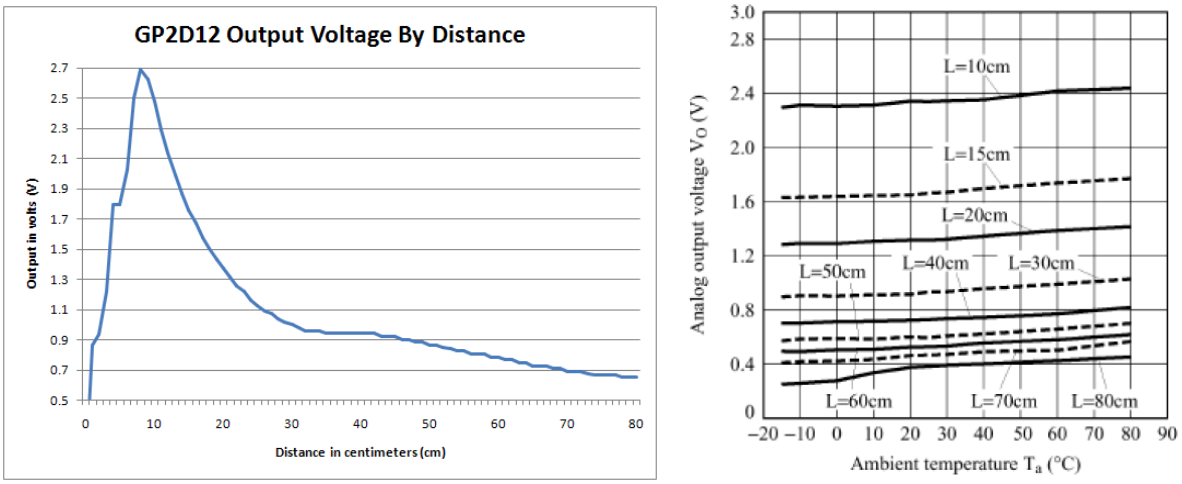


Figura 4-5: V_0 en función de la distancia (izquierda) y sensibilidad V_0 frente a la temperatura (derecha)

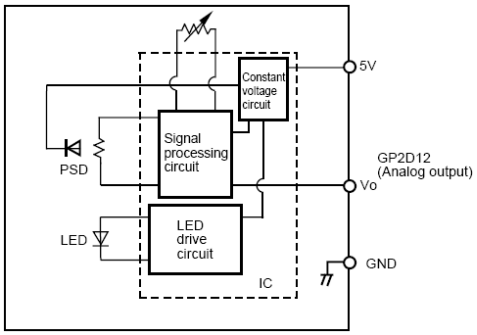
Como se ha podido observar en la Figura 4-5 izquierda, la tensión que entrega a su salida el sensor de infrarrojos hace que este sea un elemento altamente no lineal. Es por ello que para convertir la tensión analógica V_0 a distancia en cm. es necesario el uso de una

[30] Se puede obtener más información acerca de este tipo de elementos en http://en.wikipedia.org/wiki/Position_sensitive_device

tabla de conversión[31]. De la generación de esa tabla se encarga la función genera_tabla.c diseñada y explicada en el apartado dedicado al diseño software. Los valores generados se basan en la calibración realizada experimentalmente y detallada en el anexo D.

De la Figura 4-5 derecha se deduce la influencia de la temperatura sobre la medida entregada por el sensor. Obsérvese que no existe una correlación directa entre el error debido a la temperatura y la distancia que se está midiendo.

Unas lineas más abajo se muestran de nuevo dos gráficas, una que ilustra el diagrama de bloques del sensor y sus conexiones internas del sensor (Figura 4-6 izquierda extraída del datasheet del sensor SHARP GP2D12) y otra que ilustra las conexiones externas necesarias para el funcionamiento del mismo (Figura 4-6 derecha extraída de www.oopic.com/oirrange.htm y modificada posteriormente).



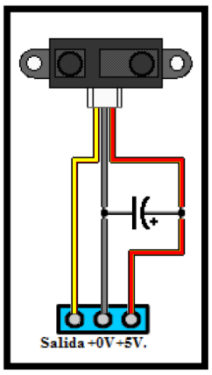


Figura 4-6: Conexiones internas (izquierda) y externas (derecha) del GP2D12

El funcionamiento de este sensor se basa en la emisión de un pulso de luz infrarroja que impacta contra el obstáculo más cercano. Si el objeto está fuera de su campo de visión, entonces el haz de luz se atenúa y, o bien no rebota, o bien no es capaz de llegar hasta el sensor de nuevo. Por el contrario, si el objeto está dentro de su campo de trabajo (0-80cm), el haz de luz infrarroja rebota y es captada por el PSD; al reflectarse la luz infrarroja, se compone un triángulo cuyos vértices quedan definidos por la lente emisora, el sensor detector y el obstáculo en cuestión (Figura 4-7). Teniendo en cuenta que el ángulo calculado por el PSD es tanto menor cuanto más alejado está el obstáculo es posible calcular la distancia en base al ángulo con el que se recibe el reflejo de luz infrarroja.

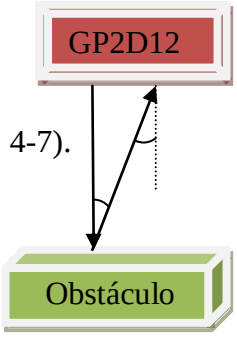


Figura 4-7: Esquema funcionamiento GP2D12

[31] La tabla de conversión que se usa en el módulo de US&IR se encuentra en el archivo tabla_magica.h

4.3 Características necesarias para el microcontrolador

Al igual que el módulo de locomoción, el módulo US&IR contiene además de elementos simples como resistencias, condensadores, conectores, etc., un elemento de gran complejidad que se encarga de gestionar todo el módulo, el microcontrolador.

El microcontrolador es el núcleo del módulo y su tarea es interpretar de manera adecuada las órdenes que recibe del módulo maestro por I^2C. Debe convertir dichas órdenes en señales comprensibles para que los sensores de ultrasonidos e infrarrojos realicen el cálculo de distancia que se requirió por parte del maestro.

Para llevar a cabo esta tarea, es necesario que el microcontrolador del módulo US&IR, además de cumplir los requisitos genéricos definidos en el apartado 2.1 cumpla otra serie de requisitos que a continuación se detallan:

- **Pines de detección de interrupciones**

Para poder calcular la anchura del pulso de eco que arrojan los sensores de ultrasonidos lo que se hace es inicializar y activar un contador al comienzo del pulso y pararlo y ver su valor al final del mismo. Para detectar tanto el comienzo como el final del pulso de eco se usan pines de detección de interrupciones, de modo que no se necesita estar constantemente comprobando si el pulso ha comenzado o finalizado sino que se encarga de ello una interrupción hardware. Así, se necesitan para cuatro sensores de ultrasonidos, cuatro pines de detección de interrupciones, sea cual sea su tipo (interno o externo).

- **Timer**

Para este módulo son necesarios dos *timer* o contadores. Uno de ellos se dedicará a contar la duración del pulso de eco, mientras que la función del segundo *timer* será detectar cuando pasan 50ms. desde la finalización de dicho eco(32).

Además los *timers* necesitan cumplir unas características mínimas. Uno de ellos debe tener un tamaño en bits tal que le permita contar pulsos de eco de duración máxima, en este caso de algo más de 30ms.; el otro timer, sin embargo, debe tener un tamaño en bits tal que le permita generar interrupciones cada 50ms.

[32] Como ya se comento, se ha considerado que entre la finalización de un pulso de eco y el comienzo de un pulso de disparo haya una espera de 50 ms. con el fin de compatibilizar el uso de SRF04 y SRF05.

- **Conversor analógico - digital**

Como se explicó en el apartado 4.2.2, los sensores de infrarrojos SHARP GP2D12 mantienen a su salida una tensión analógica proporcional a la distancia que miden, por esa razón es necesario que el microcontrolador disponga de un conversor analógico – digital, *ADC*, que le permita convertir la información analógica en información digital con el objetivo de procesarla.

- **Frecuencia de operación**

La mayoría de los microcontroladores actuales disponen de un reloj interno que proporciona una frecuencia a veces suficiente. En este caso particular la frecuencia de operación debe ser tal que permita contar pulsos del orden de µs. Esto hace que sea suficiente con frecuencias de 1MHz., no obstante, teniendo en cuenta que el microcontrolador no solo debe contar la duración de los pulsos sino que también ha de realizar operaciones, realizar conversiones de valores analógicos a digitales, establecer comunicación I^2C, etc., con esa frecuencia no sería posible cumplir los objetivos. Así pues para garantizar la detección de pulsos de duración del orden de µs. se ha decidido el uso de una frecuencia de 8MHz.

Muchos microcontroladores permiten operar con frecuencias internas de 8MHz., sin embargo, la frecuencia interna no es del todo exacta y puede generar errores al contar la duración del pulso, por esa razón se decide usar un cristal oscilador externo de 8MHz.

Con todo ello se concluye la necesidad de que el microcontrolador escogido sea capaz de soportar un cristal oscilador de 8MHz.

4.3.1. Elección del microcontrolador

Para el módulo US&IR se ha decidido trabajar con el microcontrolador ATtiny461 de Atmel porque cumple todos los requisitos mencionados anteriormente, en concreto:

Económico: Cuesta alrededor de 1,3 euros (al comprar 10 unidades).

Pequeño: En este caso dispone de 16 pines de entrada / salida para interactuar con el medio, lo cual hace que el micro sea a la vez que pequeño funcional. Además está dotado de una memoria flash de 4kbytes, capacidad suficiente para albergar el módulo US&IR.

Pines de detección de interrupciones: El micro puede detectar interrupciones de hasta quince pines distintos, dos de los cuales son capaces de detectar interrupciones externas, ya explicadas.

Timer: Dispone de dos contadores, uno de hasta 16 bits y otro de hasta 10 bits. Si se usa el contador de 16 bits para contar la duración del pulso de eco, podrían registrarse duraciones de hasta 65ms. teniendo en cuenta que el contador se incrementara cada µs. Si se usa el contador de 10 bits para contar la espera entre la finalización del eco y el inicio del siguiente disparo podrían hacerse esperas de hasta 65,28ms. si se usa un prescaler por 2048 a una frecuencia de 8MHz. Por lo tanto este microcontrolador dispone de *timers* que cumplen los requisitos.

Conversor analógico - digital: Este micro dispone de un ADC de 10 bits con, entre otras cosas, once canales de entrada diferentes. Teniendo en cuenta que se necesita procesar información de 7 sensores de infrarrojos, se dispone de canales suficientes.

Frecuencia de operación: Su frecuencia de operación es de hasta 16MHz. con una alimentación de 5V., y por tanto podremos usar un cristal oscilador de 8MHz. para contar con mayor precisión la duración del pulso de eco.

Comunicación I^2C: No tiene implementado propiamente el protocolo de I^2C pero se puede emular mediante el uso del protocolo USI.

Programación en lenguaje C: Puede programarse en lenguaje C.

No solapamiento: Existe solapamiento entre los pines que se van a dedicar a los sensores de ultrasonidos y los pines que se van a dedicar a los sensores de infrarrojos, para evitarlo solo se permiten una serie de combinaciones concretas ultrasonidos-infrarrojos detalladas en el apartado 4.2. Existe solapamiento entre los pines que gestionan la comunicación I^2C y los pines que gestionan la comunicación SPI, pero puesto que ambas comunicaciones no serán simultaneas en condiciones normales de uso, es permisible su coexistencia para la que se han añadido jumpers cuya funcionalidad y detalles se explicarán en el apartado siguiente 4.4.

4.4 Diseño hardware

Para explicar con mayor claridad y detalle cada una de las partes hardware que componen el módulo US&IR se ha representado con ayuda de los programas *Eagle 3D* y *POV Ray* una imagen tridimensional de éste módulo (Figura 4-8).

Aunque en el apartado 4.6.1 se muestra una imagen real del módulo construido, y en el anexo G.2 se muestran tanto el esquemático como el layout, se ha considerado que la imagen en 3D permite una mayor claridad y resolución del hardware que facilita una mejor explicación de cada una de las partes de las que se compone el PCB.

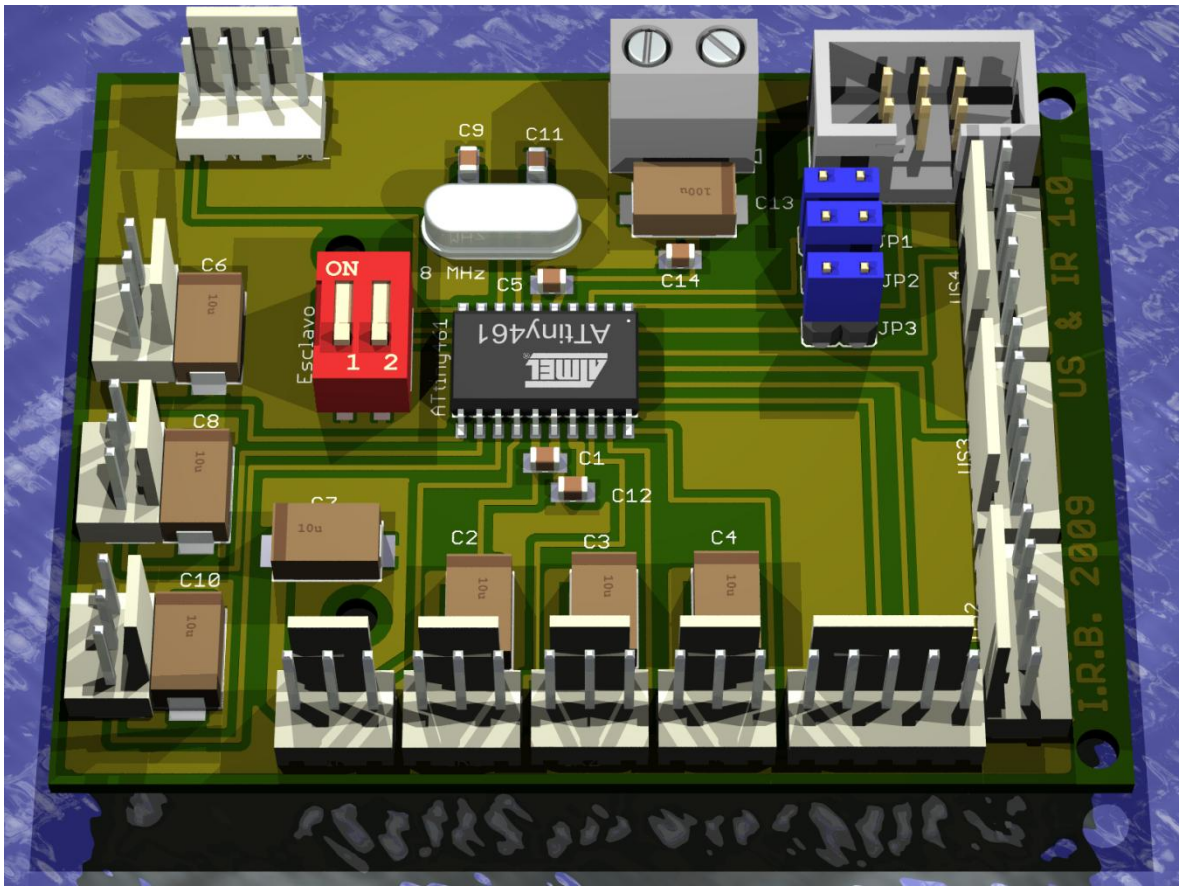


Figura 4-8: Imagen 3D del módulo US&IR

A continuación, en las páginas siguientes se ampliará y comentará la función de cada una de las partes del módulo representado.

En primer lugar se presenta, en la Figura 4-9, una ampliación de la parte superior en la que se puede observar la circuitería dedicada a la programación y a la comunicación.

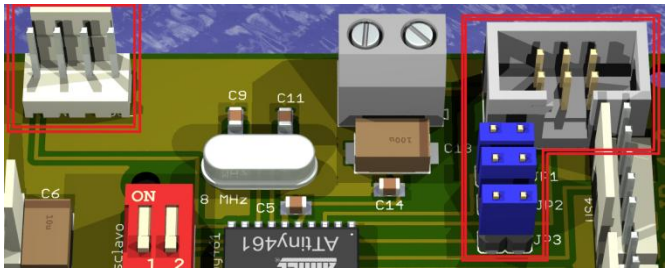


Figura 4-9: SPI e I^{2}C del módulo de US&IR

- Recuadrado en rojo a la derecha aparece el conector encargado de las comunicaciones SPI, protocolo del que se habló en el apartado 2.3 y cuya función es la de establecer comunicación entre el microcontrolador y un ordenador para modificar el firmware del módulo US&IR. Los pines que lo componen son de arriba abajo y de derecha a izquierda GND, MOSI, +5V., $\overline{RESET}$, SCK y MISO.

- Recuadrado en rojo a la izquierda aparece el conector encargado de las comunicaciones I^{2}C, protocolo del que se hablo en el apartado 2.2 y cuya función es la de establecer comunicación entre el módulo US&IR y el módulo maestro. Los pines que lo componen son de derecha a izquierda SCL, SDA, GND y $\overline{RESET}$.

  Se ha añadido la señal de $\overline{RESET}$ para posibilitar un reseteo general de todos los módulos de manera simultánea en vez de resetear de manera independiente cada uno.

  En el microcontrolador ATtiny461, los pines SDA y SCL del protocolo I^{2}C están solapados con los pines MOSI y SCK del protocolo SPI. Para que este solapamiento no produzca problemas de comunicación, se han añadido los jumpers JP1 y JP2 situados cerca del conector SPI. Así pues, en condiciones normales de funcionamiento JP1 y JP2 deben estar colocados para que las comunicaciones I^{2}C tengan éxito y deben eliminarse cuando deseen usarse las comunicaciones SPI (actualización del firmware).

  El pin de disparo del sensor 4 de ultrasonidos está multiplexado con la señal MISO de comunicación SPI, para evitar interferencias se ha conectado el jumper JP3. Este jumper debe estar colocado si desea usarse el sensor número 4 de ultrasonidos, y no debe estar colocado si desea actualizarse el firmware. En el resto de situaciones no importa si está colocado o no.

En segundo lugar se presenta, en la Figura 4-10, una ampliación de la parte central que alberga la "inteligencia" del módulo.

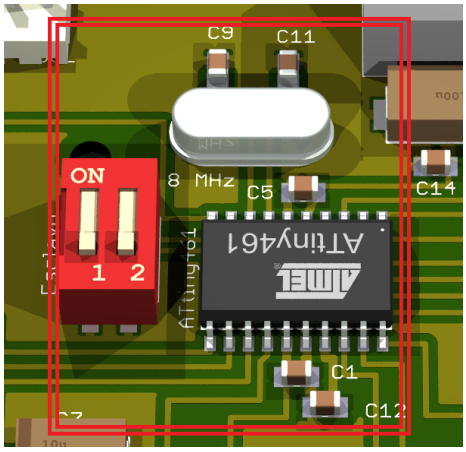


Figura 4-10: Núcleo del módulo US&IR

- Recuadrado en rojo aparece a la derecha el microcontrolador, en este caso el ATtiny461 de ATMEL, tal como se justificó en el apartado 4.3.1.

  El microcontrolador es el encargado de recibir las órdenes del maestro por I^2C, interpretarlas y actuar en consecuencia en este caso tomando medidas de distancia por uno o varios sensores bien de ultrasonidos o bien de infrarrojos. Junto a él aparecen tres condensadores de desacoplo C1, C5 y C12 de valor 100nF. todos ellos. C1 desacopla la tensión analógica, C5 desacopla la tensión digital y C12 desacopla la referencia de tensión para el ADC.

- Lo que aparece a la izquierda del microcontrolador es lo que se ha denominado "selector de esclavo" y su función es la de completar la dirección I^2C del módulo.

  Tal como se comentó en el apartado 4.1, los dos últimos bits de la dirección del módulo US&IR se modifican de acuerdo a la posición de dicho selector, pudiendo obtener de esta manera 4 direcciones diferentes correspondientes a las diferentes posiciones. Si el selector no existiera y quisiéramos manejar 5 sensores de ultrasonidos, pensaríamos en replicar el módulo US&IR, no obstante, cuando el maestro quisiera tomar la medida de uno de ellos, se tomaría la medida de un sensor de cada uno de los módulos ya que todos responderían a la misma dirección I^2C; sin embargo, gracias al selector de esclavo se puede dotar a cada PCB de una dirección distinta sin necesidad de cambiar el firmware pudiendo así controlar de manera independiente un total de hasta 16 sensores de ultrasonidos y 28 sensores de infrarrojos.

El maestro debe, a la hora de emitir cada orden, especificar el esclavo al que se refiere dentro de un mismo módulo. De modo que una posible orden sería *"calcular la*

distancia existente hasta el obstáculo más cercano al sensor 3 del esclavo número dos del módulo US&IR".

En la Tabla 4-2 se muestran la posiciones que deberían tener los switches del selector de esclavo dependiendo del esclavo al que el maestro desee dirigirse.

Número de esclavo	Switch 1	Switch 2	Dirección I^2C
Cero	ON	ON	0x08
Uno	OFF	ON	0x09
Dos	ON	OFF	0x0A
Tres	OFF	OFF	0x0B

Tabla 4-2: Posiciones de los switches del selector en función del esclavo de US&IR deseado

- Sobre el microcontrolador se puede observar el cristal oscilador de 8MHz. que se usa para tener mayor precisión en el cálculo de la duración del pulso de eco. Junto a él se hayan dos condensadores de 22pF. (C9 y C11) necesarios para su correcto funcionamiento.

En tercer lugar se presenta, en la

Figura **4-11**, una ampliación de la parte lateral derecha e inferior en la que se puede observar la etapa de salida del módulo relativa a los ultrasonidos.

- Recuadrado en rojo aparecen los cuatro conectores para los sensores de ultrasonidos SRF04/05 anteriormente explicados. El sensor de ultrasonidos número 1 se ha redondeado en color violeta, el sensor número 2 en color azul, el sensor número 3 en verde y por último el sensor número 4 en amarillo.

  La disposición de los pines de estos conectores es de arriba abajo (en el caso de los sensores 2, 3 y 4) y de derecha a izquierda (en el caso del sensor 1): VCC (+5V.), salida de eco, entrada del disparo, pin sin conectar, GND.

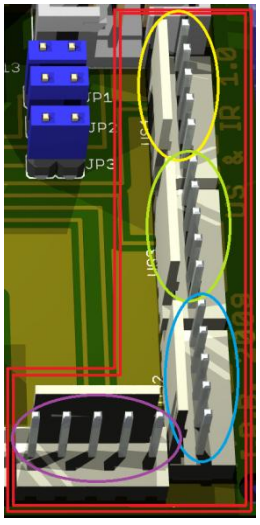


Figura 4-11: Conectores ultrasonidos SRF04/05

En cuarto lugar se presenta, en la al sensor número 5, el hexágono amarillo Figura 4-12, una ampliación de la parte lateral izquierda e inferior en la que se puede observar la etapa de salida del módulo relativa a los infrarrojos.

- En el interior de hexágonos coloreados aparecen los siete conectores para los sensores de infrarrojos GP2D12 anteriormente explicados. El hexágono blanco corresponde al sensor número 1, el hexágono morado corresponde al sensor número 2, el hexágono azul corresponde al sensor número 3, el hexágono verde corresponde al sensor número 4, el hexágono rojo corresponde al sensor número 5, el hexágono amarillo corresponde al sensor 6 y por último, el hexágono naranja corresponde al sensor 7.

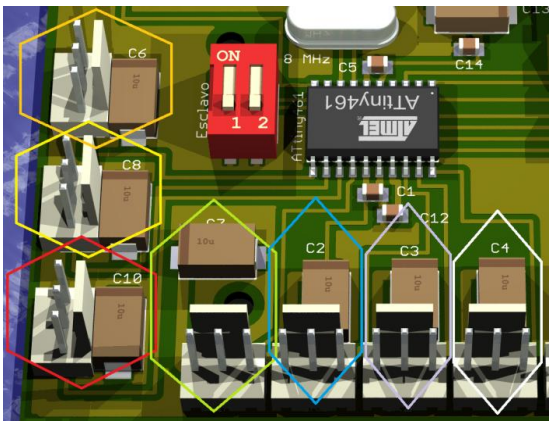


Figura 4-12: Conectores infrarrojos GP2D12

La disposición de los pines de estos conectores es de arriba abajo (en el caso de los sensores 5, 6 y 7) y de izquierda a derecha (en el caso de los sensores 1, 2, 3, 4 y 5): VCC (+5V.), GND, salida V_0 de tensión analógica del sensor.

Acompañando a cada uno de los sensores de infrarrojos puede verse un condensador de valor 10µF. tal como recomendaba el fabricante.

Por último se presenta, en la Figura 4-13, una ampliación de la parte superior central en la que aparece la entrada al módulo.

- Recuadrado en rojo aparece una clema de dos entradas. Se trata de la alimentación del circuito y es por ello que aparecen ahí los condensadores de desacoplo de 100nF. (C14) y de 100µF. (C13). Las entradas a la clema son de derecha a izquierda: entrada de 5V. y GND.

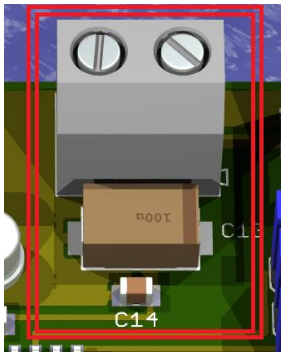


Figura 4-13: Entrada del módulo US&IR

4.5 Diseño software

En este apartado pretenden mostrarse al lector las funciones que componen el módulo US&IR. Además del prototipo y de una detallada explicación de cada una de ellas, se explica la manera de ejecutar aquellas que son accesibles desde el exterior.

Recordemos que el módulo US&IR recibe las órdenes mediante el protocolo de comunicaciones I^2C. De modo que es necesario conocer el orden y la forma de enviar los parámetros de las funciones.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
inicializar_US	void	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del módulo US&IR, esto hace que sea inaccesible desde el exterior y que por tanto sea el esclavo quien debe hacer una llamada a ésta al inicializarse. En concreto, lo que hace la función es:

1. Configura, en el microcontrolador, los registros necesarios para que éste permita el salto a la interrupción PCINT_vect cuando ocurra un cambio de nivel en alguno de los pines PCINT0…7. El pin concreto que podrá generar la interrupción será seleccionado en otra función, genera_pulso().
2. Configura el Timer0 como contador de 16 bits con prescaler por 8. Teniendo en cuenta el uso de un cristal oscilador de 8MHz., se tiene que el contador se incrementará en uno cada µs. ya que:

$$\left(\frac{8 * 10^6 \text{Hz}}{8}\right)^{-1} = 1\mu\text{s}. \qquad [4.5]$$

3. Configura el Timer1 como contador de 8 bits con prescaler por 4096. Teniendo en cuenta el uso de un cristal oscilador de 8MHz., se tiene que el contador se incrementará en uno cada 512µs. ya que:

$$\left(\frac{8 * 10^6 \text{Hz}}{4096}\right)^{-1} = 512\mu\text{s}. \qquad [4.6]$$

   Cuando el valor del contador sea 98 se generará la interrupción Timer1_COMPA_vect, o lo que es lo mismo, cada 98*512µs. ≈ 50ms. se generará la interrupción.

Finalmente habilita las interrupciones generales.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
inicializar_IR	void	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del módulo US&IR, esto hace que sea inaccesible desde el exterior y que por tanto sea el esclavo quien debe hacer una llamada a ésta al inicializarse. En concreto, lo que hace la función es:

1. Selecciona, como tensión de referencia para el conversor A/D, 2,56V. con condensador de bypass conectado a AREF para mejorar la inmunidad al ruido.
2. Activa el preescaler por 64, de modo que la frecuencia de muestreo esté localizada entre 50 y 200kHz., concretamente resulta de 8MHz/64=125kHz.
3. Habilita el ADC y las interrupciones generales.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
genera_pulso	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del módulo US&IR, esto hace que sea inaccesible desde el exterior. En concreto, lo que hace la función es:

1. Configura el microcontrolador para que el pin ECO del sensor indicado como parámetro sea capaz de generar una interrupción al cambiar de nivel.
2. Genera un pulso de duración mínima de 10µs.[(33)] por el sensor indicado como parámetro de la función.

El parámetro sensor puede tomar el valor uno, dos, cuatro u ocho. Este parámetro indica el sensor de ultrasonidos que se usará durante el cálculo de la distancia.

- Si el parámetro toma valor uno, se usará el sensor de ultrasonidos número 1.
- Si el parámetro toma valor dos, se usará el sensor de ultrasonidos número 2.
- Si el parámetro toma valor cuatro, se usará el sensor de ultrasonidos número 3.
- Si el parámetro toma valor ocho, se usará el sensor de ultrasonidos número 4.

Para dar valor al parámetro sensor se puede usar la ecuación $2^{\text{sensor deseado}-1}$, donde *sensor deseado* tomaría valor entre 1 y 4.

[33]Se ha elegido una anchura de pulso de 13 µs para el disparo de los sensores de ultrasonidos de este módulo.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
seleccionar_sensor	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del módulo US&IR, esto hace que sea inaccesible desde el exterior. En concreto, lo que hace la función es:

1. Conectar internamente, mediante un multiplexor, el pin asociado al sensor de infrarrojos correspondiente con el ADC del microcontrolador. Para ello hace uso del parámetro *sensor* cuyo valor ha de ser 1, 2, 4, 8, 16, 32 o 64, y se obtiene de la ecuación: $2^{sensor\ deseado\ -1}$, donde *sensor deseado* tomaría valor entre 1 y 7.

Nombre de la interrupción	Protocolo por parte del maestro
ISR (PCINT_vect)	El maestro no puede hacer una llamada a una interrupción propia del esclavo

Se trata de una interrupción propia del módulo US&IR, esto hace que sea inaccesible desde el exterior.

La interrupción será ejecutada cuando se detecte un cambio de nivel en alguno de los pines conectados a la salida *ECO* de los sensores de ultrasonidos. De este modo se pretende detectar el comienzo y el final de dichos ecos para así, medir su duración.

Nombre de la interrupción	Protocolo por parte del maestro
ISR (TIMER1_COMPA_vect)	El maestro no puede hacer una llamada a una interrupción propia del esclavo

Se trata de una interrupción propia del módulo US&IR, esto hace que sea inaccesible desde el exterior.

La interrupción será ejecutada aproximadamente cada 50ms. tal y como se configuró en la función inicializar_US. Gracias a esta interrupción se consigue garantizar que, entre la finalización de un eco y el comienzo de un disparo, pase un mínimo de 50ms.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
lee_US	uint8_t uint16_t	void	Condición de START Escribir en el esclavo 000010xx Transmitir el comando 0xA1 Transmitir el dato activar Condición de REPEATED START Leer del esclavo 000010xx Tantas veces como 1's del 1er parámetro{ Recibe MSB de distancia Recibe LSB de distancia } Condición de STOP

La tarea de esta función es calcular la distancia existente entre uno o varios sensores de ultrasonidos y el obstáculo u obstáculos más cercanos.

El parámetro activar puede tomar valor entre uno y quince. En este parámetro está codificada la información acerca de los sensores de ultrasonidos que desean usarse. Deberá colocarse un '1' en el bit correspondiente al sensor que desee leerse en formato MSBF. De modo que si desea realizarse la lectura del sensor número uno de ultrasonidos deberá colocarse el bit menos significativo a '1', es decir, habrá que poner en la variable activar 00000001 = 0x01. Sin embargo, si desea realizarse la lectura de los sensores dos y tres de ultrasonidos deberán colocarse a '1' el segundo y tercer bit menos significativo, es decir, habrá que poner en la variable activar 00000110 = 0x06.

El parámetro sensores es un vector de uint16_t. En principio debe contener un máximo de 4 variables de tipo uint16_t. En cada una de esas variables se almacenará la distancia de los sensores de ultrasonidos en formato MSBF. Así pues:

Si la variable activar tuviera valor 0x06, es decir, quisiéramos leer el valor de los sensores de ultrasonidos dos y tres; la distancia calculada por el sensor dos se encontraría en sensores[0] y la distancia calculada por el sensor tres se hallaría en sensores[1]. En ese caso, no sería necesario que esta variable fuera un vector de 4 uint16_t, bastaría con pasar como parámetro un vector de 2 uint16_t.

Si la variable activar tuviera valor 0x04, es decir, quisiéramos leer el valor del sensor de ultrasonidos tres; la distancia calculada por dicho sensor se encontraría en sensores[0]. En ese caso, tampoco sería necesario que esta variable fuera un vector de 4 uint16_t, bastaría con pasar como parámetro un vector de 1 uint16_t.

Por ejemplo, para calcular el resultado de restar la lectura del sensor de ultrasonidos número cuatro menos la lectura del sensor de ultrasonidos número 2 podrían ejecutarse las instrucciones:

lee_US (0x0A, sensores); diferencia = sensores[1] – sensores[0];

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
lee_IR	uint8_t uint16_t	void	Condición de START Escribir en el esclavo 000010xx Transmitir el comando 0xA1 Transmitir el dato activar Condición de REPEATED START Leer del esclavo 000010xx Tantas veces como 1's del 1er parámetro{ Recibe MSB de distancia Recibe LSB de distancia } Condición de STOP

La tarea de esta función es calcular la distancia existente entre uno o varios sensores de infrarrojos y el obstáculo u obstáculos más cercanos.

El parámetro activar puede tomar valor entre uno y quince. En este parámetro está codificada la información acerca de los sensores de infrarrojos que desean usarse. Deberá colocarse un '1' en el bit correspondiente al sensor que desee leerse en formato MSBF. De modo que si desea realizarse la lectura del sensor número uno de infrarrojos deberá colocarse el bit menos significativo a '1', es decir, habrá que poner en la variable activar 00000001 = 0x01. Sin embargo, si desea realizarse la lectura de los sensores dos y tres de infrarrojos deberán colocarse a '1' el segundo y tercer bit menos significativo, es decir, habrá que poner en la variable activar 00000110 = 0x06.

El parámetro sensores es un vector de uint16_t. En principio debe contener un máximo de 7 variables de tipo uint16_t. En cada una de esas variables se almacenará la distancia de los sensores de infrarrojos en formato MSBF. Así pues:

Si la variable activar tuviera valor 0x06, es decir, quisiéramos leer el valor de los sensores de infrarrojos dos y tres; la distancia calculada por el sensor dos se encontraría en sensores[0] y la distancia calculada por el sensor tres se hallaría en sensores[1]. En ese caso, no sería necesario que esta variable fuera un vector de 7 uint16_t, bastaría con pasar como parámetro un vector de 2 uint16_t.

Si la variable activar tuviera valor 0x04, es decir, quisiéramos leer el valor del sensor de infrarrojos tres; la distancia calculada por dicho sensor se encontraría en sensores[0]. En ese caso, tampoco sería necesario que esta variable fuera un vector de 7 uint16_t, bastaría con pasar como parámetro un vector de 1 uint16_t.

Por ejemplo, para calcular el resultado de restar la lectura del sensor de infrarrojos número cuatro menos la lectura del sensor de infrarrojos número 2 podrían ejecutarse las instrucciones:

lee_IR (0x0A, sensores); diferencia = sensores[1] – sensores[0];

Una vez detalladas cada una de las funciones, se procede a mostrar en la Figura 4-14 el diagrama de flujo de la función principal del módulo esclavo US&IR. En él puede verse tanto el modo en que se reciben datos por las comunicaciones I^2C como el modo en que se realizan las diversas llamadas a las funciones correspondientes.

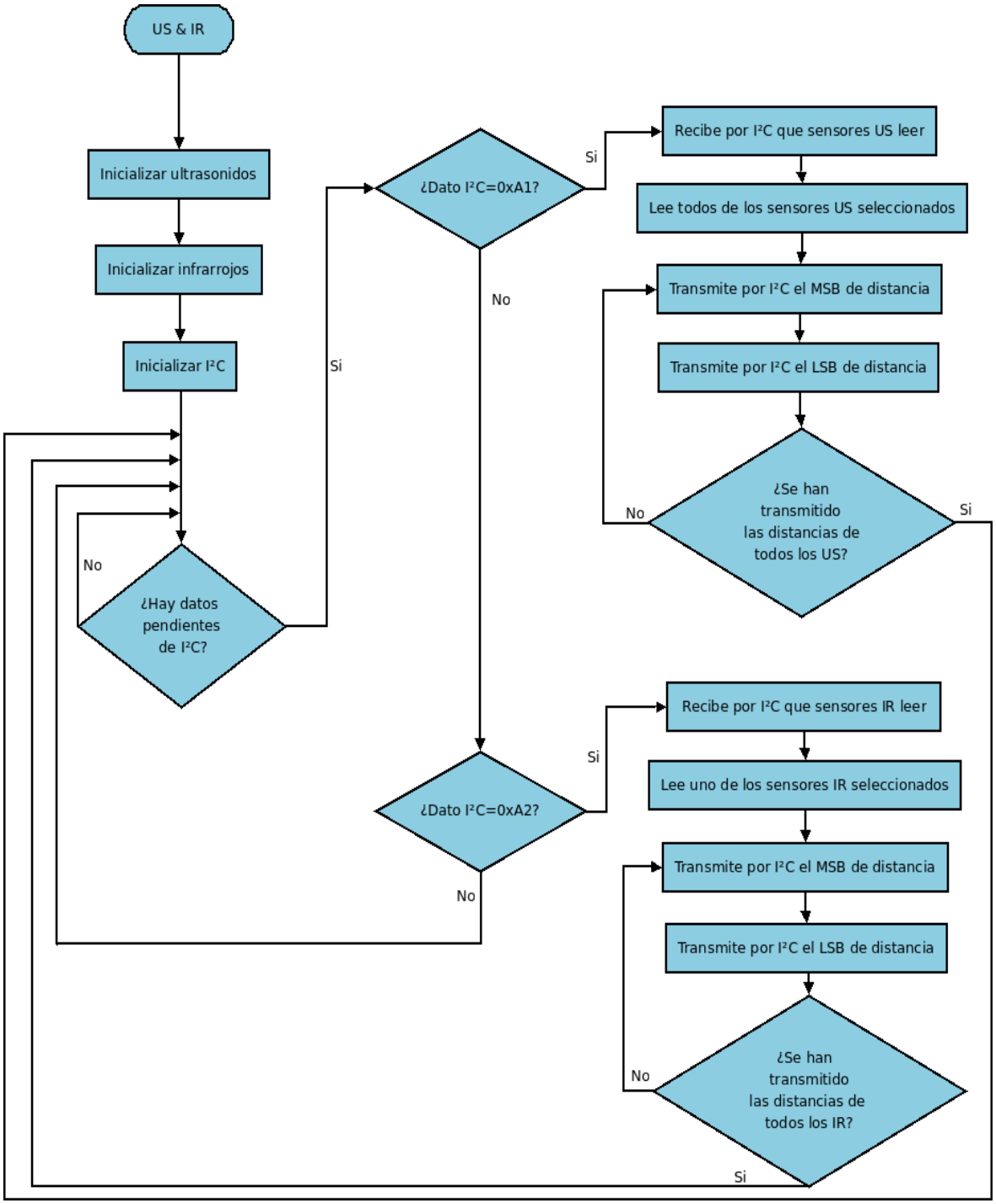


Figura 4-14: Diagrama de flujo del programa principal del módulo US&IR

4.6 Construcción e información técnica

4.6.1. Fase de prototipado del módulo

Antes de construir de manera definitiva el módulo US&IR, se hizo un prototipo de éste (Figura 4-15), con el objetivo de encontrar posibles errores y corregir problemas de diseño. Para el prototipo se consideró suficiente el hecho de colocar únicamente dos conectores para los sensores de ultrasonidos y otros dos conectores para los sensores de infrarrojos.

Con ese primer prototipo se hicieron una serie de calibraciones y medidas de prueba que fueron de gran ayuda para la calibración final de infrarrojos realizada con la versión definitiva del módulo y que se presenta en el anexo D.

Por último resaltar que al igual que el prototipo del módulo de locomoción el cableado se realizó por la parte posterior usando la técnica de wrapping.

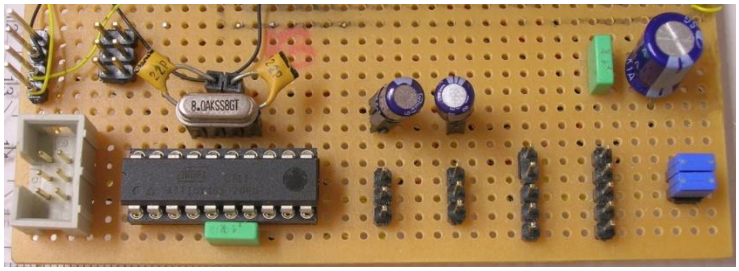


Figura 4-15: Prototipo del módulo US&IR

4.6.2. Especificaciones, aspecto y presupuesto del módulo

En este apartado se muestra en primer lugar información acerca de las especificaciones técnicas del módulo, necesarias para saber por ejemplo la resolución de los ultrasonidos y los infrarrojos; dicha información se aprecia en la Tabla 4-3.

Especificaciones técnicas del módulo US&IR	
Tensión necesaria	5V.
Corriente media con sensores	116,32mA. (tres sensores SRF05 y dos sensores GP2D12)
Corriente máxima	260,64mA. (siete sensores GP2D12)
Nº máximo de ultrasonidos	4 si sólo se usan ultrasonidos
Nº máximo de infrarrojos	7 si sólo se usan infrarrojos
Rango de trabajo ultrasonidos	Desde 3cm. hasta 3m.(SRF04) ó 4m.(SRF05)
Resolución de ultrasonidos	1mm.
Error máximo en ultrasonidos	±1cm.
Rango de trabajo infrarrojos	Desde 10cm. hasta 80cm.
Resolución de infrarrojos	1cm.
Error máximo en infrarrojos	±1cm.
Dimensiones	55,2450mm. de alto y 70,8025mm. de ancho (39,1148cm^2.)

Tabla 4-3: Especificaciones técnicas del módulo US&IR

En segundo lugar se muestra, en la Figura 4-16, el módulo US&IR una vez construido en su versión definitiva.

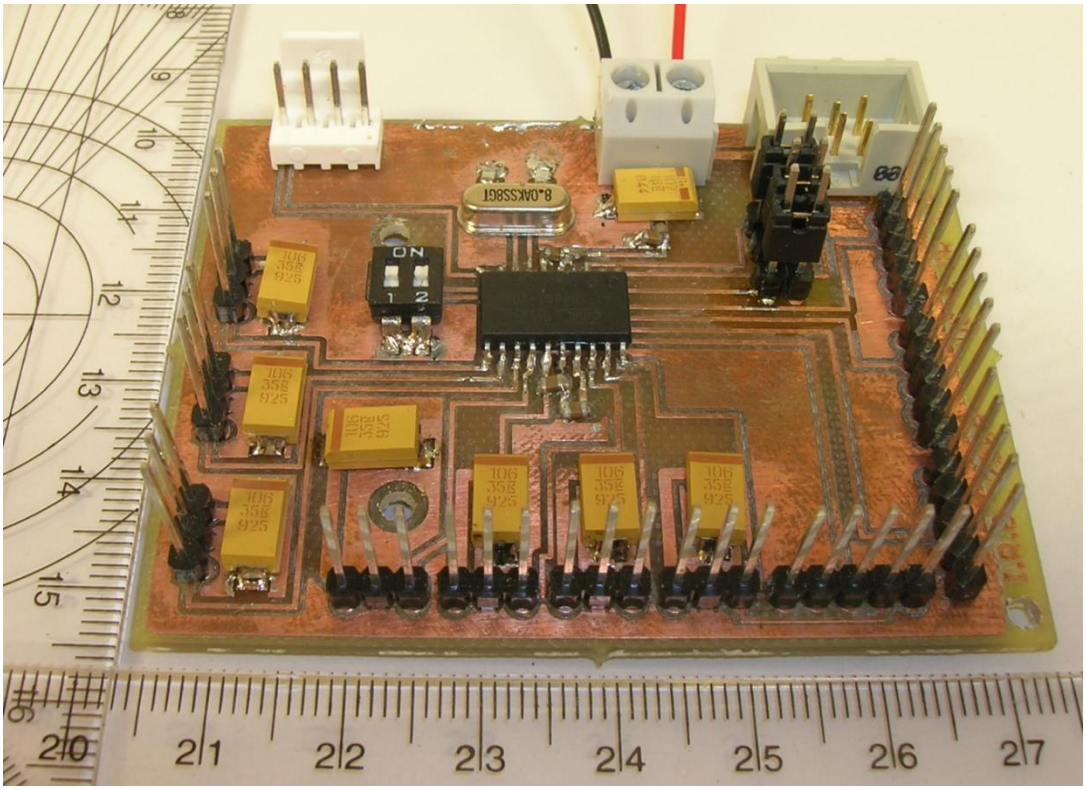


Figura 4-16: Imagen real del módulo US&IR

En tercer lugar se hace constar el precio que costaría construir el módulo de nuevo. En la Tabla 4-4, se muestra el precio de cada componente necesario para realizar el montaje del módulo US&IR. Los precios han sido extraídos de Merchan electrónica y componentes[34], exceptuando algunos de ellos, que por no encontrarse en el distribuidor anterior se han consultado en Farnell[35], estos últimos están en color azul. Los que no se han encontrado en ninguno de los dos anteriores han sido consultados en RS[36], en este caso, han sido coloreados en naranja.

Referencia	Encapsulado	Componente	Valor	Precio
C1, C5, C12, C14	0805	Condensador	100nF	4x0.0116€
C9, C11	0805	Condensador	22pF	2x0.022€
C13	7343	Condensador	100µF	1x1.32€
C2, C3, C4, C6, C7, C8, C10	7343	Condensador	10µF	7x0.33€
ATtiny461	SOIC20	Microcontrolador	ATtiny461	1x2.58€
JP1…3		Jumper	Tira min. 6 pines	1x0.51€
Q1	HC49/S	Cristal oscilador	8MHz	1x1.06€
CONECTOR SPI		Conector	SPI	1x1.02€
J1	Molex 2,54mm	Conector	I^2C	1x0.479€
J2…5	Molex 2,54mm	Conector	Ultrasonidos	4x0.528€
J6…12	Molex 2,54mm	Conector	Infrarrojos	7x0.529€
Esclavo	DIP (DIL)	Switch SMD	Selector de esclavo	1x1.28€
X1	Pasante 2 vías	Clema 5mm	Bornes para PCB	1x0.27€
Total				**16,7344€**

Tabla 4-4: Precio de los componentes del módulo US&IR

Teniendo en cuenta que construir cada $dm.^2$ de PCB cuesta 60€. Se concluye que al coste de los componentes hay que sumar 60€ por construcción.

[34] Los precios están sujetos a modificaciones y deben comprobarse en http://www.e-merchan.com/

[35] Los precios están sujetos a modificaciones y deben comprobarse en http://es.farnell.com/

[36] Los precios están sujetos a modificaciones y deben comprobarse en http://es.rs-online.com/

5 Módulo de comunicaciones

5.1 Motivación del módulo

Una de las funcionalidades de la que se ha querido dotar a la plataforma robótica ha sido la capacidad para intercambiar información con el usuario, o con un PC, tanto recibiendo información del mismo como transmitiéndola. Es deseable que el sistema sea capaz de informar al usuario acerca de algunas situaciones y mostrar datos relativos a los sensores que tiene incorporados con el fin de que haya una interactuación usuario-sistema.

Y esa es precisamente la finalidad del denominado módulo de comunicaciones. La interacción con el usuario puede darse de formas muy diversas y es por ello que se han creado dos modelos diferentes del módulo de comunicaciones, de modo que pueda escogerse el más adecuado a las necesidades de cada aplicación.

En primer lugar, se ha diseñado el modelo A del módulo de comunicaciones. Se trata de un módulo capaz de transmitir y recibir información por los puertos que tiene integrados, que son un puerto USB y un puerto serie RS-232. La información se enviará y/o recibirá por uno de los puertos dependiendo de las órdenes de otro módulo, al que se denominará módulo maestro. Las órdenes del módulo maestro serán recibidas y correctamente interpretadas por el módulo gracias al bus de comunicaciones I^2C que interconecta la plataforma robótica detallado en el apartado 2.2.

En segundo lugar, se ha diseñado el modelo B del módulo de comunicaciones. Se trata de un módulo capaz de realizar las mismas tareas del modelo A pero que además incorpora una pantalla de texto y gráfica LCD. Gracias a la existencia de este modelo no solo se transmite o recibe información por alguno de los puertos USB o serie de los que está dotado, sino que esa información puede mostrarse por la pantalla LCD junto con imágenes explicativas.

Al igual que en el modelo A, el modelo B recibe las órdenes del módulo maestro por I^2C; no obstante, se ha querido dotar a este modelo de mayor valor añadido, de modo que es posible conectarlo directamente al ordenador sin necesidad de usar el módulo maestro y el usuario será capaz de manejar la pantalla LCD directamente desde el puerto USB o serie

con gran facilidad **gracias a la API que se ha creado para tal finalidad tanto para Windows como para Linux** y que puede consultarse en el CD que se acompaña.

Para que cada de los modelos tan solo ejecute las órdenes que van dirigidas a él, cuando están conectados al sistema uno o ambos modelos, es necesario identificarlos de algún modo. Para ello se usan las direcciones I^2C. La dirección I^2C grabada en el firmware del modelo A es la 0x10 y la dirección I^2C grabada en el firmware del modelo B es la 0x0C; no obstante, los dos últimos bits de ambas direcciones son configurables mediante hardware. De este modo, las posibles direcciones I^2C del modelo A son 0x10, 0x11, 0x12 y 0x13, mientras que las del modelo B son 0x0C, 0x0D, 0x0E y 0x0F. La razón de permitir estas cuatro direcciones, es que en el mismo sistema puedan coexistir un total de hasta cuatro clones de cada uno de los modelos de éste módulo.

Gracias al hecho de poder configurar por hardware hasta cuatro direcciones I^2C, el sistema es capaz de manejar hasta ocho puertos USB y ocho puertos serie (ya que cada modelo del módulo de comunicaciones maneja un puerto serie y un puerto USB[37] además de cuatro pantallas LCD (ya que cada modelo B del módulo es capaz de manejar una pantalla LCD).

La configuración hardware de la dirección I^2C del modelo A se explica en el apartado 5.4.2 y la del modelo B en el apartado 5.5.2.

[37] No pueden usarse el puerto USB y el puerto serie a la vez dentro de un mismo modelo del módulo de comunicaciones; la razón de este hecho es que ambos puertos de comunicaciones están multiplexados internamente.

5.2 Funcionalidad

En este apartado se pretenden dar a conocer todas las funcionalidades del módulo de comunicaciones, para una mayor comprensión de las misma se detallarán en primer lugar las referentes al modelo A y en segundo lugar las referentes al modelo B.

Como se dijo en el apartado 5.1, el modelo A del módulo de comunicaciones es capaz de interpretar una serie de órdenes recibidas por I^2C. Dichas órdenes son las siguientes:

3. Recibir y enviar datos por el puerto serie RS-232 (conector DB9 hembra).
4. Recibir y enviar datos por el puerto USB (conector USB tipo B).

En cuanto al modelo B del módulo de comunicaciones, éste tiene dos modos de funcionamiento. El primer modo de funcionamiento consiste en la recepción de órdenes por parte del maestro y su ejecución; en tal modo las órdenes que es capaz de interpretar son las mostradas a continuación:

1. Recibir y enviar datos por el puerto serie RS-232 (conector DB9 hembra).
2. Recibir y enviar datos por el puerto USB (conector USB tipo B).
3. Escribir letras[38] y/o números en cualquier lugar de la pantalla LCD (determinado por una fila y columna de comienzo).
4. Escribir frases y palabras en cualquier lugar de la pantalla LCD sin pensar en el final de línea[39].
5. Mostrar una imagen o un conjunto de ellas[40] por separado o al mismo tiempo en una posición concreta de la pantalla LCD (determinada por fila y columna de comienzo).
6. Encender o apagar un pixel concreto de la pantalla LCD.
7. Encender o apagar la pantalla LCD para ahorrar energía.
8. Borrar todo el contenido de la pantalla LCD sin apagarla.
9. Colorear el fondo de la pantalla LCD de cualquier color según una descripción RGB del mismo.

[38] Las letras pueden ser mayúsculas, minúsculas o una mezcla de ambas. También se contemplan los caracteres '?', '¿', '¡', '!' y '.'

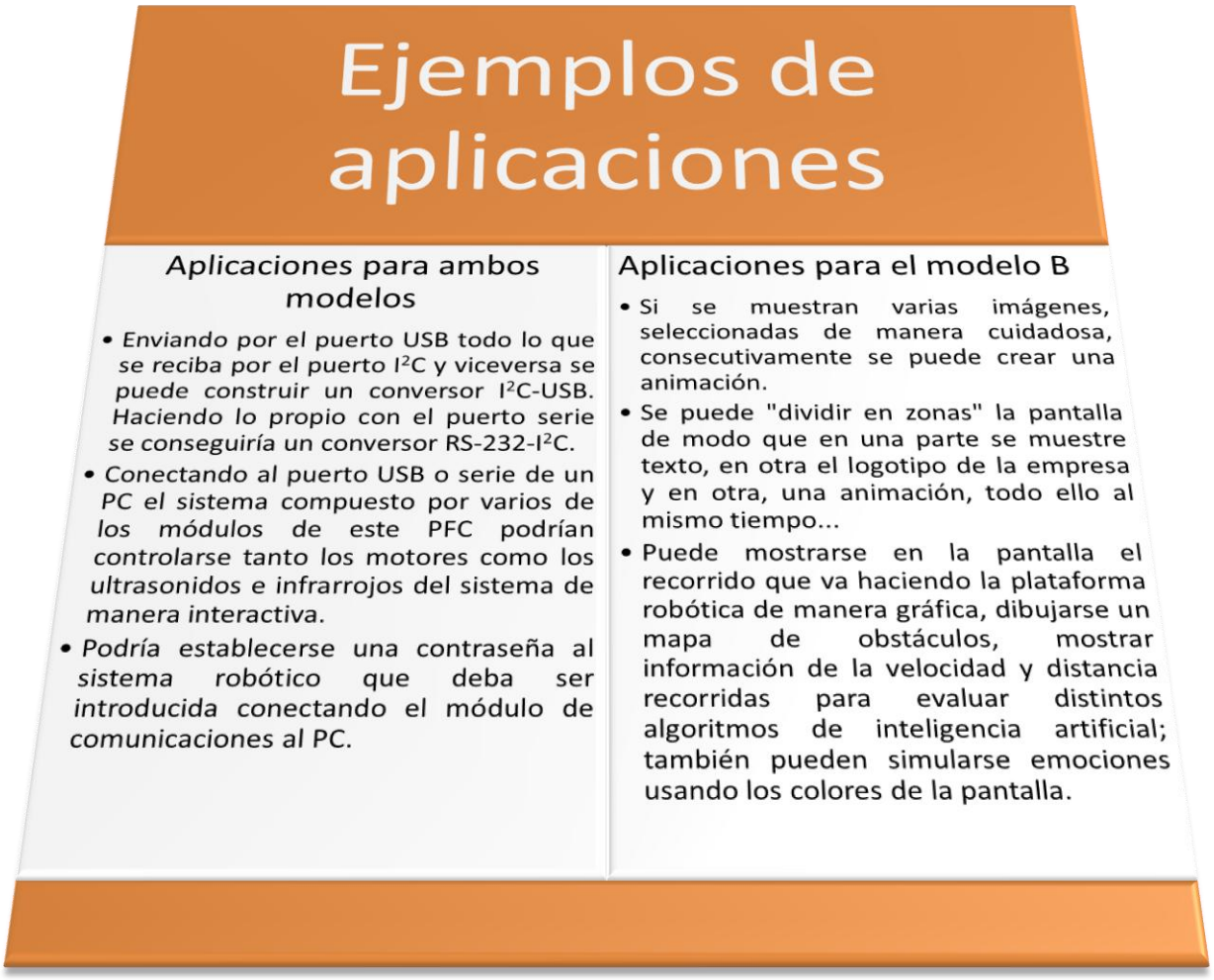
[39] Cuando se alcanza el final de la línea, el resto de la palabra o frase continúa escribiéndose en la línea siguiente.

[40] Pueden mostrarse simultáneamente por pantalla tantas imágenes como quepan en la misma.

El segundo modo de funcionamiento del modelo B del módulo de comunicaciones es el que se conoce como modo autónomo[41], en este modo, se reciben órdenes por parte de otro dispositivo capaz de usar bien el protocolo serie RS-232 o bien el protocolo USB (ambos se detallarán más adelante) prescindiendo así del módulo maestro para funcionar. En este segundo modo, **las funcionalidades son las mismas que en el modo anterior**.

La elección de uno u otro modo de funcionamiento se realiza mediante la colocación o no de un jumper determinado, del que se habla en el apartado el 5.5.2.

Combinando de manera adecuada las órdenes anteriores, se pueden realizar acciones complejas e interesantes tales como las que se muestran a continuación.



[41] En este caso se entiende como autónomo la capacidad de funcionar sin necesidad del módulo maestro de la plataforma robótica, aunque en sentido estricto no trabaje por cuenta propia.

5.2.1. Comunicaciones serie RS-232

- **Introducción**

El RS232-E es un estándar de comunicaciones propuesto por la Asociación de Industrias Electrónicas (EIA).

Se introdujo en la industria alrededor del año 1962 tras haber sido actualizado un total de hasta cinco veces (de ahí la letra E) para adecuarse a las aplicaciones de las comunicaciones seriales. El estándar es muy simple y se usa para conectar dos sistemas entre sí, originalmente un DTE o equipo terminal de datos y un sistema periférico, DCE.

La información es transmitida bit a bit enviando un solo bit a la vez. Tradicionalmente se ha usado y sigue usándose para conectar los PCs a dispositivos como módems; incluso los teclados, ratones y otros periféricos de ordenadores se conectaban siguiendo este estándar. Aunque otros interfaces como USB o Ethernet también transmiten en serie el término *puerto serie* identifica al hardware que sigue el estándar RS-232. Hoy en día, en la mayoría de los periféricos el USB ha reemplazado al puerto serie puesto que es más rápido.

Aunque el puerto serie se considera obsoleto, aún sigue presente en gran parte de los sistemas industriales, es por esta razón que se ha considerado adecuado incluirlo en el módulo de comunicaciones de este PFC.

- **Funcionalidad**

El estándar RS232 es de tipo asíncrono y por lo tanto requiere de bits de inicio y de parada. Tras el bit de inicio se envían una serie de bits de datos que varían entre cinco y nueve. A continuación se envía el bit de paridad (es opcional), que puede indicar paridad par o paridad impar y sirve para detectar fallos de transmisión de datos; finalmente, se envía el bit de parada que puede ser uno o dos o uno y medio (cuando se usan cinco bits de datos se envía un bit de parada con una duración equivalente a un bit y medio). En este caso particular, el módulo de comunicaciones permite al usuario seleccionar tanto el número de bits de parada como el tipo de paridad, par, impar o ninguna. Los dos modelos del módulo de comunicaciones permiten también seleccionar la velocidad en baudios a la que se transmitirá y recibirá la información.

El estándar permite que la comunicación sea simplex, dúplex o full dúplex. En el caso particular del módulo de comunicaciones, la comunicación que se establece es dúplex, esto

quiere decir que se puede tanto enviar como recibir información, pero no hacer ambas cosas de manera simultánea.

El ordenador controla el puerto serie mediante un circuito integrado específico, llamado UART (Transmisor-Receptor-Asíncrono Universal). Del mismo modo, los microcontroladores de cada uno de los modelos del módulo de comunicaciones hacen uso de una USART[42] para el manejo de las comunicaciones según el estándar RS-232.

Una vez que ha comenzado la transmisión de un dato, los bits deben llegar uno detrás de otro a una velocidad constante y en determinados instantes de tiempo. Por eso se dice que el RS-232 es asíncrono por carácter y síncrono por bit.

A continuación, en la Figura 5-1 se muestra la transmisión de la letra J. La imagen ha sido extraída de internet y posteriormente ha sido modificada. En la parte superior puede verse la información tal y como sale de la U(S)ART del microcontrolador y en la parte inferior como debe llegar al conector DB9 según el estándar RS-232. Obsérvese como es necesario además de una adaptación de niveles, la inversión de los niveles lógicos; de esta tarea de adaptación se encargará el driver MAX232 que se analiza en el apartado 5.3.1.

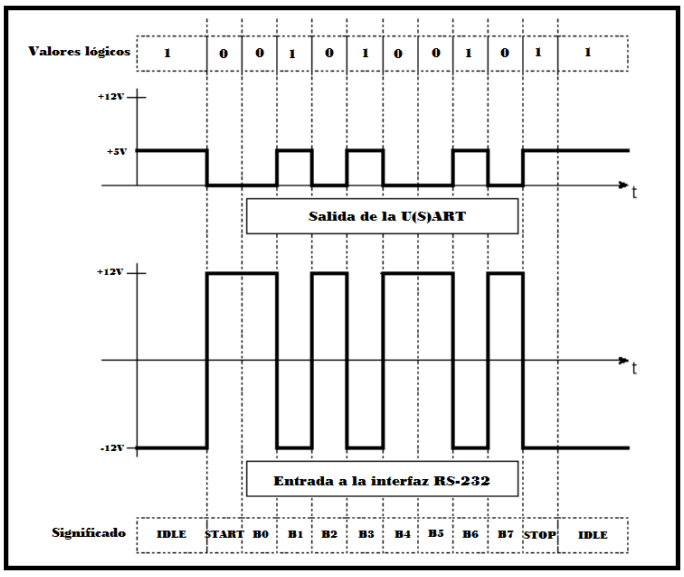


Figura 5-1: Comparación TTL – RS232

Los pines que portan los datos son RxD y TxD. Las demás se encargan de otros trabajos: DTR indica que el ordenador esta encendido, DSR que el aparato conectado a dicho puerto esta encendido, RTS que el ordenador puede recibir datos (porque no está ocupado), CTS que el aparato conectado puede recibir datos, y DCD detecta que existe una

[42] Una USART es una UART pero que además de transmisión-recepción asíncrona permite la posibilidad de usar transmisión-recepción síncrona.

comunicación, presencia de datos. Tanto en el modelo A como en el modelo B del módulo de comunicaciones se usan únicamente los pines RxD, TxD y GND.

- **Consideraciones eléctricas**

Las consideraciones eléctricas de las comunicaciones RS-232 abarcan tres aspectos:

- ***Niveles de tensión***: No usan niveles TTL (definidos entre 0V. y 5V.) sino que usan niveles de tensión entre +5V. y +15V. para el nivel alto y niveles entre -5V. y -15V. para el nivel bajo. No obstante para cumplir con un margen de ruido de 2V. las tensiones quedaron fijadas entre +3V. y +15V. para el cero lógico (*space*) y entre -3V. y -15V. para el uno lógico (*mark*).
- ***Slew rate***: El slew rate, definido como la variación de tensión por unidad de tiempo también queda definida para el RS-232, siendo la máxima permitida de 30V/ms. La motivación de este límite es reducir la diafonía o crosstalk.
- ***Impedancia de la línea***: También queda definida por el estándar la impedancia entre el equipo transmisor y el equipo receptor. La carga vista por el transmisor ha de ser entre 3kΩ. y 7kΩ. Además, la máxima longitud para el cable de conexión quedó fijada originalmente a 15 metros; pero en la revisión D se determinó que la máxima capacitancia del cable debía ser de 2700 pF. y por tanto la longitud máxima quedaba fijada por la capacitancia por unidad de longitud del cable usado.

Para acabar, se muestra en la Tabla 5-1 un resumen de las especificaciones eléctricas extraída de (21).

	RS-232
Cabling	Single-ended
Number of Devices	1 transmit, 1 receive
Communication Mode	Full duplex
Distance (max)	50 feet at 19.2kbps
Data Rate (max)	1Mbps
Signaling	Unbalanced
Mark (data 1)	-5V (min) -15V (max)
Space (data 0)	5V (min) 15V (max)
Input Level (min)	±3V
Output Current	500mA (Note that the driver ICs normally used in PCs are limited to 10mA)
Impedance	5kΩ (Internal)
Bus Architecture	Point-to-Point

Tabla 5-1: Especificaciones eléctricas del estándar RS-232

- **Consideraciones mecánicas**

Para las conexiones por comunicación serie el estándar RS232 recomienda un conector de 25 pines, DB25. El equipo DCE debe tener un conector macho y el cable que se conecte a él un conector hembra. Por otro lado, el equipo DTE debe tener un conector hembra y el cable que se conecte a él un conector macho. A pesar de que el estándar recomendaba el conector de 25 pines, normalmente el conector que se usa es de 9 pines, DB9. Tanto la distribución de los pines DB9 como la distribución de los pines del DB25 se muestran en la Tabla 5-2 extraída de (22).

DB9	DB25	Function
1	8	Data carrier detect
2	3	Receive data
3	2	Transmit data
4	20	Data terminal ready
5	7	Signal ground
6	6	Data set ready
7	4	Request to send
8	5	Clear to send
9	22	Ring indicator

Tabla 5-2: Distribución pines DB9 y DB25

A continuación, se muestran imágenes de la distribución que siguen los pines en un conector DB9. En la Figura 5-2, a la izquierda, se muestra el conector macho DB9 y a la derecha, el conector DB9 hembra. Ambas imágenes han sido extraídas de internet y posteriormente modificadas.

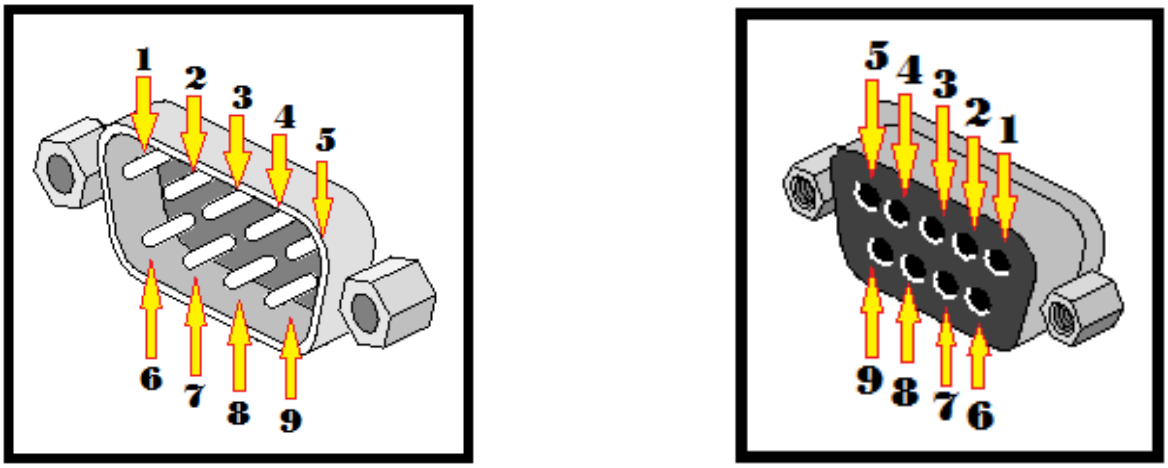


Figura 5-2: Conector DB9 macho (izquierda) y hembra (derecha)

5.2.2. Comunicaciones puerto USB

El objetivo de este apartado es dar unas pinceladas acerca de este protocolo estándar de comunicaciones para poder comprender su importancia en la actualidad y justificar así su existencia dentro del módulo de comunicaciones del presente PFC.

USB es una arquitectura de bus de comunicaciones más que un protocolo. Fue creado en el año 1996 por siete empresas: IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC.

USB se desarrolló con el objetivo de unificar todos los conectores (conectores serie y conectores paralelo) y crear uno más sencillo, con mayores prestaciones y que mejorase las capacidades *plug and play*[43] de los dispositivos permitiendo a estos conectarse y desconectarse sin la necesidad de reiniciar el sistema(23).

Aunque en algunas situaciones el estándar IEEE1394 (Fireware) y el estándar IEEE802.3 (Ethernet) presentan mayores ventajas que el USB en cuanto a velocidad de transferencia y prestaciones, la implementación del protocolo USB es más sencilla (24).

Algunas características del USB es que usa velocidades de 12Mbps (muy por encima de la velocidad de los puertos paralelo y serie) y el funcionamiento del bus se basa en el paso de un testigo.

El controlador envía un testigo por el bus y el dispositivo cuya dirección coincide con la que esta anotada en el testigo puede intercambiar datos con él. Este funcionamiento puede recordar al de las redes *Token Ring*[44].

Existen diversas versiones de USB, la 1.1, la 2.0 y la 3.0. Ésta última es la más actual (a principios de 2010 comenzarán a aparecer los primeros dispositivos compatibles con USB 3.0) y surge con la idea de mejorar las características de la versión 2.0 tanto en cuanto a la velocidad de transferencia (600Mbps.) como a la cantidad de energía que es capaz de entregar (900mA.) (25).

En las versiones anteriores a la 3.0, USB usa cuatro hilos que se transmiten en un cable de par trenzado con impedancia característica de 90 Ω ± 15%, dos de los hilos son de

[43] Plug and play es una tecnología que permite a un dispositivo informático ser conectado a un ordenador sin tener que configurar ni proporcionar parámetros a sus controladores. (29)

[44] Estándar recogido en el IEEE 802.5 y que define una red de área local LAN en configuración de anillo (Ring), con método de paso de testigo (Token) como control de acceso al medio.

alimentación y los otros dos se denominan D+ y D-. Estos, utilizan señalización diferencial en full dúplex para combatir los efectos del ruido electromagnético en enlaces largos. Los niveles de transmisión de la señal varían de 0 a 0'3 V para nivel bajo lógico y de 2'8 a 3'6 V para nivel alto lógico en las versiones 1.0 y 1.1. En las primeras versiones, los alambres de los cables no están conectados a masa, pero en el modo de alta velocidad se tiene una terminación de 45 Ω a tierra o un diferencial de 90 Ω para acoplar la impedancia del cable. Este puerto sólo admite la conexión de dispositivos de bajo consumo, es decir, que tengan un consumo máximo de 100mA en la versión 1.1 y 2.0 (26).

Y por último resaltar que todas las versiones más actuales de USB son compatibles con las menos actuales, de modo que USB 2.0 es compatible con USB 1.1 y USB 3.0 es compatible con las dos anteriores.

Por todo lo anterior se entiende que además de importante es necesario diseñar un módulo que contenga al menos un puerto USB ya que este tipo de puertos no solo son ampliamente usados sino que aún siguen en plena expansión y mejora.

A continuación se presentan en la Tabla 5-3 algunas de las alternativas existentes para el control de un puerto USB sin implementar directamente el protocolo.

Company	Chips	CPU Interface	Bus Speed
Agere Systems	USS-820D	Parallel	Full
FTDI Chip	FT232BM	Asynchronous serial	Full
	FT245BM	Parallel	Ful
National Semiconductor	USBN9603/4	Parallel, Microwire	Full
Philips Semiconductors	PDIUSB12, ISP1181/83	Parallel	Full
	ISP1581	Parallel	Full/High
PLX Technology	NET22272	Parallel	Full/High

Tabla 5-3: Controladores de USB (información de la tabla extraída de **(24)**)

Esa es precisamente la solución escogida para este PFC, concretamente mediante el uso del chip FT232BL del que se hablará con detalle más adelante.

5.2.3. Pantalla LCD

En este apartado se analiza ligeramente la pantalla LCD empleada para el modelo B del módulo de comunicaciones.

Se trata de una pantalla LCD del fabricante Displaytech Ltd(45), concretamente el modelo 64128M que consta de 128x64 puntos y admite tanto texto como gráficos. Está además dotada de 3 LED para iluminar la pantalla; uno rojo, uno verde y uno azul, con el objetivo de generar toda la gama RGB.

Cada uno de los puntos que componen la pantalla es de 0,48mm^2 y es necesario suministrar una tensión de 3V. para el funcionamiento de la misma. El controlador que usa la pantalla es el ST7565R-G.

Tanto las características mecánicas como la distribución de pines de la pantalla usada en este PFC se pueden ver en el datasheet del fabricante.

La pantalla puede controlarse bien mediante la interfaz SPI de cuatro líneas o bien mediante la interfaz paralelo, tanto en la arquitectura 8080 como en la 6800. En el modelo B de comunicaciones se ha optado por escoger la interfaz SPI por requerir menos pines.

Para seleccionar la interfaz serie es necesario poner a valor cero lógico el pin P/S; en ese caso, las señales de la comunicación son $\overline{CS1}$, A0, SCL y SI:

- La señal SCL es la encargada de marcar el período de reloj.
- La señal SI es la encargada de transmitir el dato con la frecuencia que determina la señal SCL.
- La señal $\overline{CS1}$ es una señal de habilitación, y debe estar a valor cero lógico para que se establezca la comunicación.
- La señal A0 es la encargada de diferenciar si los datos que se envían corresponden a un comando (toma valor cero lógico) o por el contrario desea representarse como valor por la pantalla (toma valor uno lógico). Sólo se tiene en cuenta el valor de la señal A0 en el octavo flanco de subida de la señal SCL.

[45] Puede consultarse más información acerca de la pantalla o del fabricante en su página web oficial, que es http://www.displaytech.com.hk

A continuación, en la Figura 5-3 se muestra el funcionamiento del protocolo de comunicaciones que se sigue para el control de la pantalla. Las dos figuras y la tabla mostradas a continuación se han extraído del datasheet del controlador ST7565R.

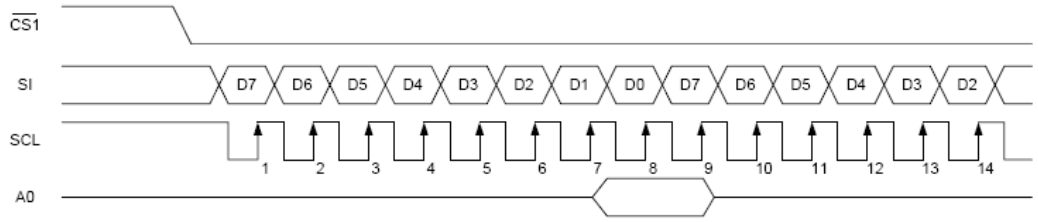


Figura 5-3: Protocolo de comunicaciones serie para la pantalla LCD

Bajo estas líneas se muestra en la Figura 5-4 con mayor detalle el protocolo mostrado anteriormente, pudiendo apreciarse los tiempos que deben cumplirse para controlar correctamente la pantalla mediante la interfaz serie de 4 hilos.

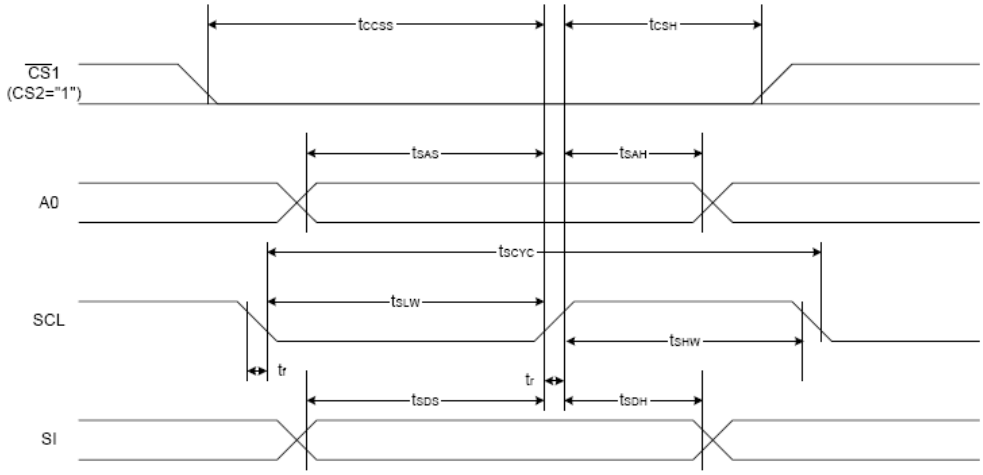


Figura 5-4: Diagrama de tiempos para la comunicación serie

En la Tabla 5-4 se muestran los valores de los tiempos referidos a las señales de la Figura 5-4.

Item	Signal	Symbol	Condition	Rating		Units
				Min.	Max.	
4-line SPI Clock Period	SCL	T_{scyc}		50	—	ns
SCL "H" pulse width		T_{shw}		25	—	
SCL "L" pulse width		T_{SLW}		25	—	
Address setup time	A0	T_{SAS}		20	—	
Address hold time		T_{sah}		10	—	
Data setup time	SI	T_{sds}		20	—	
Data hold time		T_{SDH}		10	—	
CS-SCL time	CS	T_{css}		20	—	
CS-SCL time		T_{csh}		40	—	

Tabla 5-4: Valores de tiempo para las señales de la comunicación serie

El controlador divide a la pantalla LCD en páginas horizontales y en páginas verticales. Cada página horizontal está compuesta por ocho filas y cada página vertical está compuesta por ocho columnas. De este modo, la pantalla queda dividida en 8 páginas horizontales y 16 páginas verticales. Resultando finalmente 8x16 = 128 páginas cada una de 8x8 = 64 píxeles o puntos.

Hay una gran cantidad de comandos que se pueden enviar, desde encender la pantalla, hasta apagarla, pasando por un borrado de todos los píxeles o la fijación de una página y columna determinada; no obstante, y dado que dichos comandos pueden consultarse tanto en el datasheet del controlador como en el fichero *lcd.h* del modelo B del módulo de comunicaciones que se recoge en el cd que se acompña se ha optado por no repetir en este apartado los comandos existentes para el control de la pantalla.

Para acabar se mostrarán todas las letras mayúsculas y minúsculas (Figura 5-5), números y símbolos (Figura 5-6) que se han diseñado para usar en la pantalla del modelo B de comunicaciones. Es posible modificar cualquiera de los caracteres simplemente accediendo al fichero *letras.h* que aparece en el cd que se acompaña.

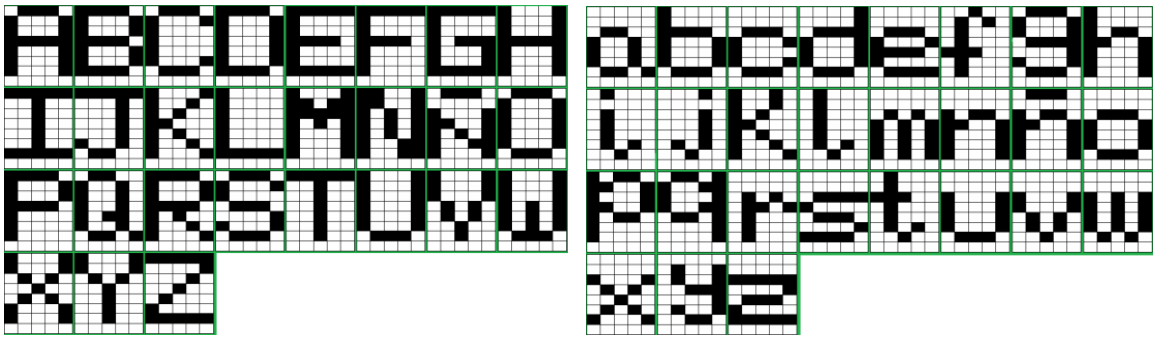


Figura 5-5: Letras mayúsculas y minúsculas diseñadas

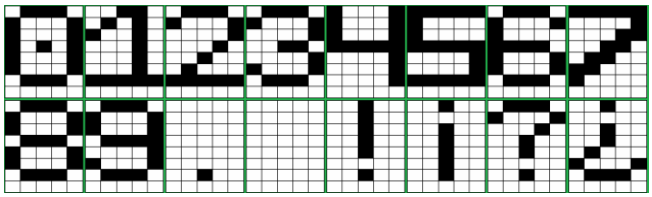


Figura 5-6: Números y símbolos diseñados

Con el objetivo de que el usuario conozca en qué modo se está usando la pantalla, se han diseñado imitaciones de los logotipos registrados USB e I^2C. Ambos se muestran en la Figura 5-7.

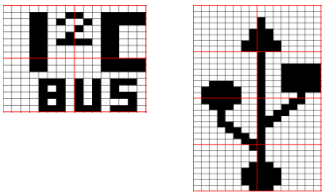


Figura 5-7: Logotipos diseñados

Con el objetivo de hacer funcional la pantalla se ha implementado el software necesario para que si el usuario decide escribir una palabra no aparezcan todas las letras de la palabra juntas sino que exista una pequeña separación natural entre ellas (

Figura 5-8). La separación entre letras es de un píxel. Teniendo en cuenta que cada letra ocupa cinco píxeles y que tras cada una de ellas se deja una columna de píxeles vacía se deduce que en una línea de 128 píxeles caben un total de floor(128/6) = 21 caracteres.

Si bien al no caber un texto completo en la pantalla el propio módulo de comunicaciones lo divide en líneas; es responsabilidad del usuario tener en cuenta que en ocasiones no se pueden separar las letras que componen una palabra y es necesario colocar guiones o pasar la palabra completa a la siguiente línea estableciendo espacios justo antes de la misma. Por ejemplo si deseamos escribir la frase "Esto no es correcto gramaticalmente", se observará que la primera línea de la pantalla acaba con "…correcto g" y la siguiente empieza con "ramaticalmete". El usuario debería preveer esa situación y colocar un doble espacio antes de la palabra "gramaticalmente" para que así aparezca toda ella en la siguiente línea.

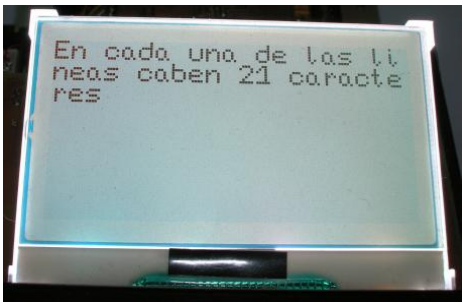


Figura 5-8: Pantalla del módulo de comunicaciones con un texto escrito

Además, si se desea usar el módulo para recibir órdenes del maestro, al inicializar la pantalla aparecerá en ella el logotipo de I^2C; mientras que si se desea usar directamente con otra máquina dotada de puerto USB o serie (por ejemplo un P.C.) aparecerá, al encender el módulo, el logotipo de USB además de un mensaje informativo. Ambos logotipos se muestran en la Figura 5-9.

Figura 5-9: Módulo de comunicaciones con los logotipos dibujados

5.3 Drivers necesarios

Como se ha detallado en el apartado 5.2.1, las comunicaciones serie RS232 usan niveles cerca de ±15V. mientras que el microcontrolador usa niveles TTL de 0V. a 5V. Esto hace que la comunicación por conexión directa entre el microcontrolador y el PC sea inviable. Para poder comunicar ambos elementos es necesario un driver conversor que sea capaz de convertir niveles RS232 en niveles TTL y viceversa tal como se muestra en la Figura 5-10.

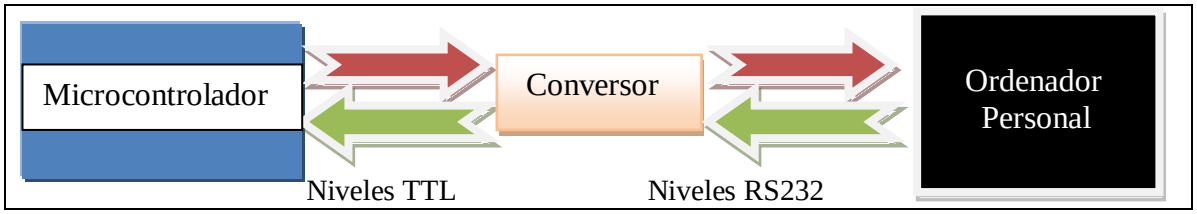


Figura 5-10: Forma correcta de conectar un PC y un microcontrolador

Existen en el mercado numerosos integrados capaces de realizar esta tarea pero el más usado con diferencia es el MAX232 de Dallas Semiconductor-MAXIM(46) que se analizará en el apartado 5.3.1.

Por otro lado, implementar el protocolo USB, analizado en el apartado 5.2.2, en el microcontrolador central del módulo de comunicaciones tiene ciertos inconvenientes, la tarea es costosa y además se dedicaría una carga importante a atender estas comunicaciones restando eficiencia al resto de tareas. Es por esas razones que se ha decidido usar un integrado cuya función fuese hacer una traducción entre comunicaciones serie RS232 y comunicaciones USB. De este modo, el microcontrolador solo necesita interpretar las comunicaciones serie, siendo transparente para él si las comunicaciones extremo a extremo son serie o USB. Para tal fin se ha usado un integrado muy popular en estos casos, el FT232BL de FTDI Chip(47), que se analiza en el apartado 5.3.2.

[46] Para más información acerca del integrado puede acceder al datasheet del fabricante en http://www.maxim-ic.com/products.cfm

[47] Para más información acerca del integrado puede acceder al datasheet del fabricante en http://www.ftdichip.com/FTProducts.htm

5.3.1. MAX-232

El MAX232[48] es un IC capaz de convertir los valores que se manejan en las comunicaciones serie (±12V.) a niveles TTL. En la Figura 5-11 se muestra a la izquierda la distribución de pines del integrado y a la derecha un esquema de conexión del mismo.

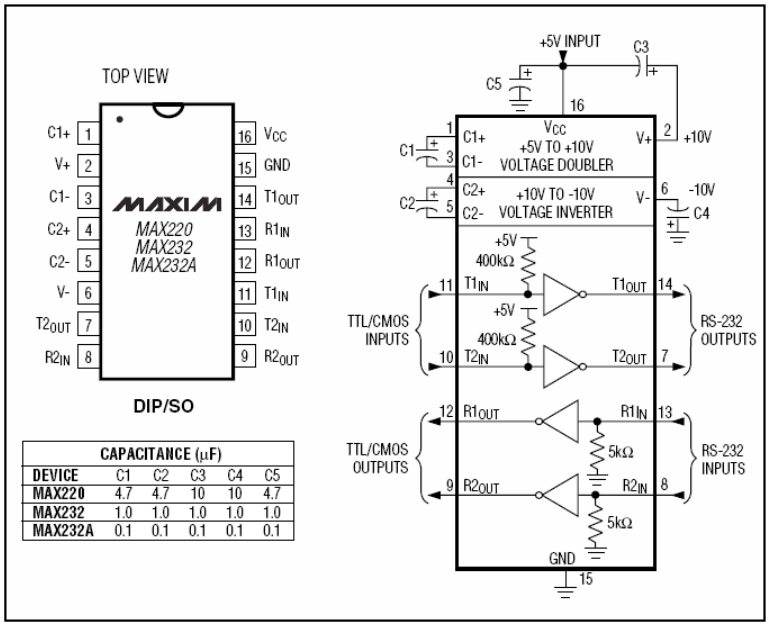


Figura 5-11: Esquema funcional del MAX232 (imagen del datasheet)

El MAX232 permite la conversión simultánea de dos comunicaciones serie de dos hilos, ya que está dotado de 4 canales. En el módulo de comunicaciones, tan solo se usarán dos de esos canales; concretamente el canal de conversión entre los pines 11 y 14 para envío de información por parte del microcontrolador y el canal de conversión entre los pines 13 y 12 para recepción de información por parte del microcontrolador. En la Figura 5-12 se muestran las conexiones de manera más visual.

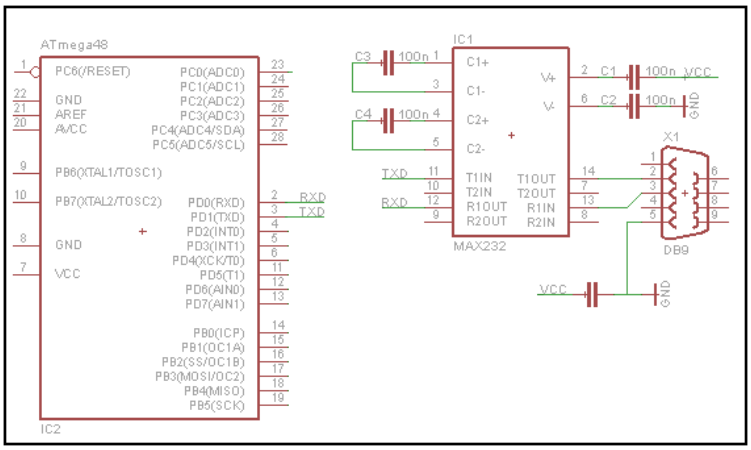


Figura 5-12: Algunas conexiones del módulo de comunicaciones

[48] Además de MAX232 pueden usarse otros integrados idénticos en cuanto a funcionalidad tales como el ST232 que también fue probado en este PFC.

Como se puede ver, el conexionado es muy simple. Tan solo es necesario:

1. Alimentar el integrado, conectando 5V. al pin 16 y 0V. al pin15.
2. Conectar cuatro condensadores de 100nF. como se muestra en la imagen.
3. Hacer el conexionado por parte del microcontrolador y por parte del PC o maestro de comunicaciones serie. Concretamente,
    a. Conectar el pin RXD del microcontrolador al pin 12 del MAX232.
    b. Conectar el pin TXD del microcontrolador al pin 11 del MAX232.
    c. Conectar el pin 14 del MAX232 a la patilla 2 del conector DB9 hembra.
    d. Conectar el pin 13 del MAX232 a la patilla 3 del conector DB9 hembra.
    e. Conectar la patilla 5 del conector DB9 hembra a 0V.

5.3.2. FT232BL

Como se comentó anteriormente, la implementación del protocolo USB en el microcontrolador restaría eficiencia al módulo. No obstante, es necesario usar este tipo de comunicaciones.

Imagine por un momento que el módulo de comunicaciones solo tuviera puerto serie. En la actualidad, muchos ordenadores ya no disponen de tal puerto, de modo que no podríamos conectarlo al PC. Sin embargo, todos los ordenadores modernos disponen de cómo mínimo un puerto USB.

Sería muy interesante poder establecer una comunicación serie mediante el estándar RS232 pero a través del puerto USB. Esto es precisamente lo que permite el integrado FT232BL; es capaz de crear un puerto serie virtual conectado al puerto USB del ordenador. De este modo, el microcontrolador se comunica con el PC como si éste tuviera un puerto serie, aunque en realidad no tenga y esté conectado físicamente al puerto USB.

El FT232BL es capaz de crear dichos puertos virtuales en tres S.O distintos: Windows, Linux y Machintosh, lo que permite que el módulo de comunicaciones pueda conectarse a un ordenador con cualquiera de los tres sistemas operativos mencionados. Además permite establecer comunicaciones serie con siete u ocho bits de datos, uno o dos bits de parada y paridad par, impar o ninguna de ellas.

Una gran ventaja de este integrado es que puede trabajar con microcontroladores a 3.3V., lo cual es muy interesante para el módulo de comunicaciones ya que recuerde que la pantalla trabajaba a 3V.

A continuación, en la Figura 5-13 se presenta un diagrama de bloques funcional del FT232BL extraído de la hoja de datos del fabricante.

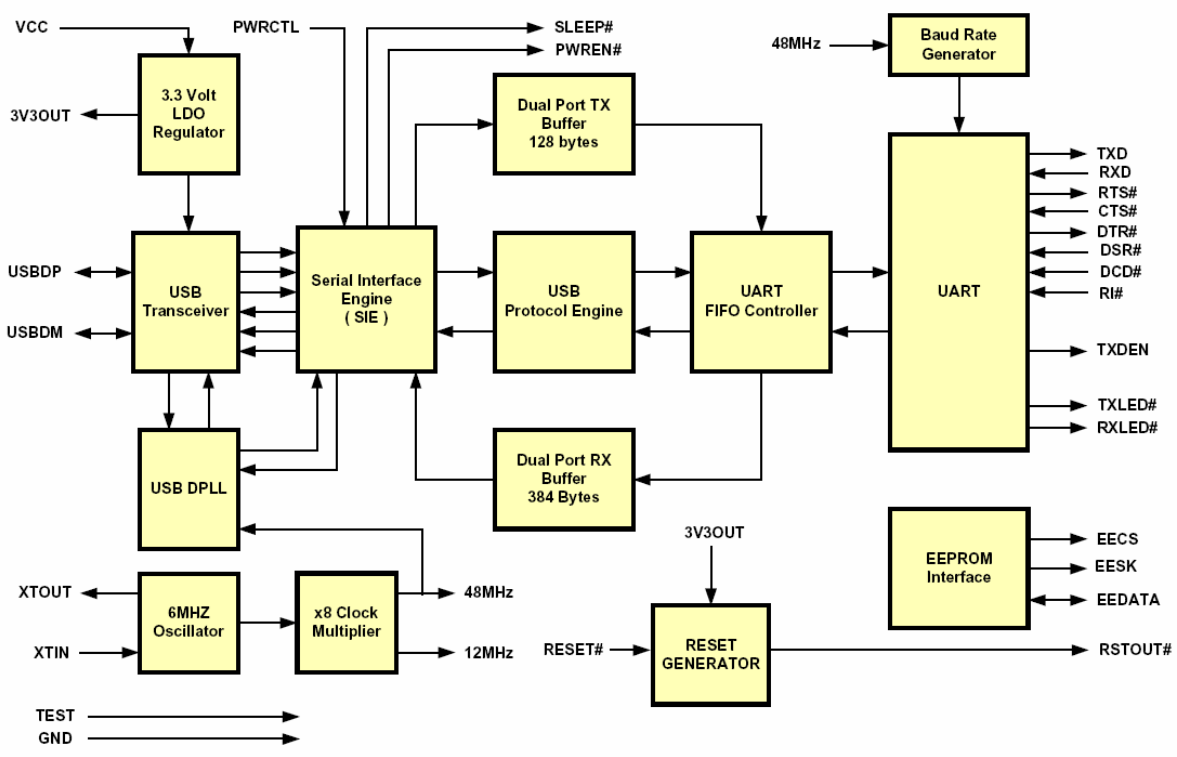


Figura 5-13: Diagrama funcional del FT232BL

Este integrado es bastante más complejo a la hora de hacer las conexiones que el MAX232. Requiere algunos componentes externos más. No obstante, se conecta de manera muy simple al microcontrolador. Basta con conectar el pin 25 del IC al pin RXD del microcontrolador y el pin 24 del IC al pin TXD del microcontrolador.

El FT232BL necesita de un cristal oscilador de 6MHz. para funcionar correctamente y tiene la capacidad de informar al usuario acerca de la transmisión o recepción de información si se le añaden dos LEDs; el resto de componentes necesarios dependerá de la configuración en la que se desee usar el integrado.

Existen cuatro configuraciones básicas para usar este integrado como conversor RS232- USB y se muestran a continuación (*las imágenes han sido extraídas del datasheet*).

En la Figura 5-14 se muestra la configuración USB Bus Powered, en la que la tensión se toma del propio bus USB. Este tipo de configuraciones no se deben usar si se va a consumir más de 100mA. en estado activo o más de 500µA. en estado suspendido.

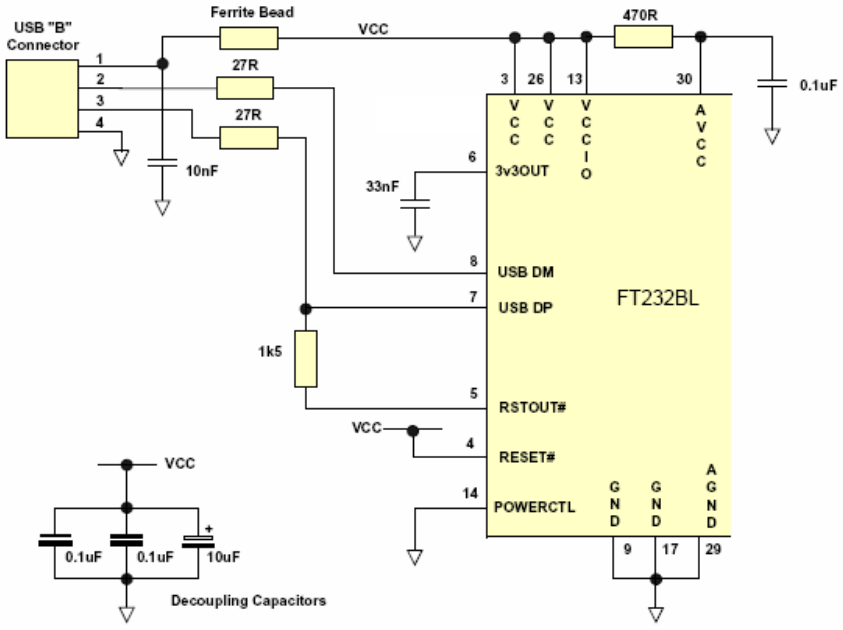


Figura 5-14: USB Bus Powered Configuration

En la Figura 5-15 se muestra la configuración USB Self Powered, en la que la tensión no se toma del bus USB sino de otra fuente de alimentación de la que debe disponerse. Se recomienda esta configuración si se desea disponer de más corriente que la que el puerto USB puede entregar.

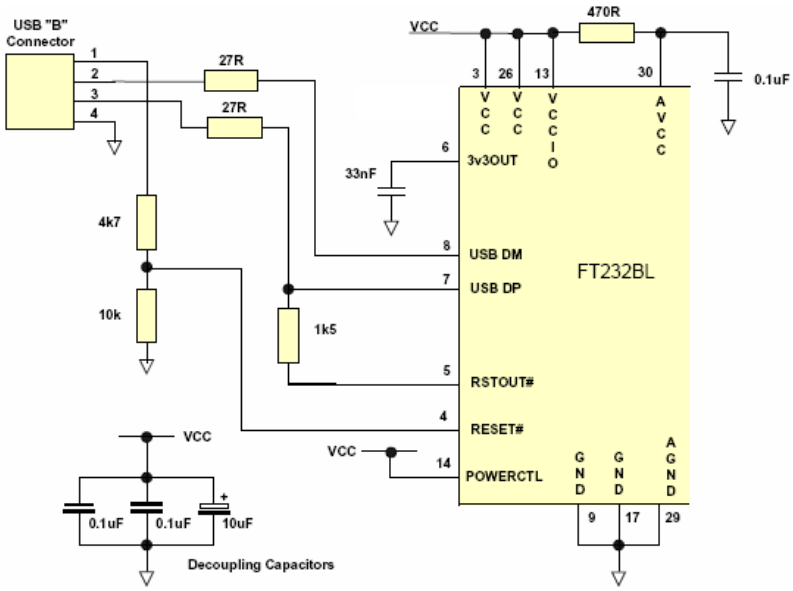


Figura 5-15: USB Self Powered Configuration

En la Figura 5-16 se muestra la configuración USB Bus Powered con interfaz de 3,3V. En este tipo de configuraciones se toma la tensión del propio bus USB pero con la particularidad de que el integrado FT232BL funcionará con una interfaz de 3,3V. esto es, sus pines tendrán una tensión de 3,3V. en estado alto lógico.

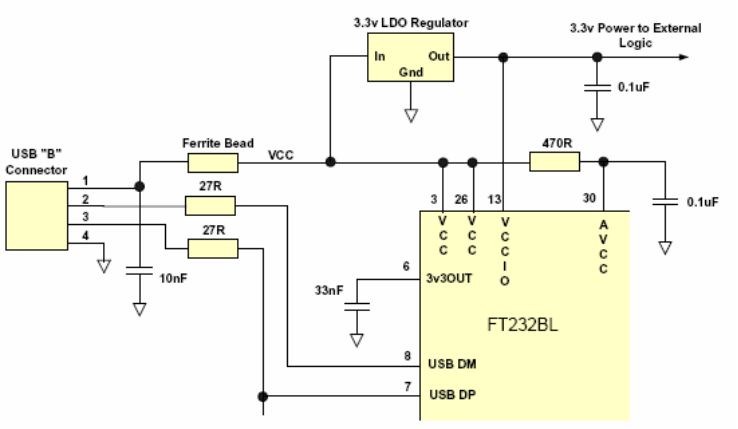


Figura 5-16: USB Bus Powered with 3.3V. logic drive Configuration

En la Figura 5-17 se muestra la configuración USB Self Powered con interfaz de 3,3V. En este tipo de configuraciones no se toma la tensión del bus USB. El integrado FT232BL funcionará con una interfaz de 3,3V. como en el caso anterior, pero esta vez necesita una fuente de 5V. y otra de 3V. para funcionar correctamente.

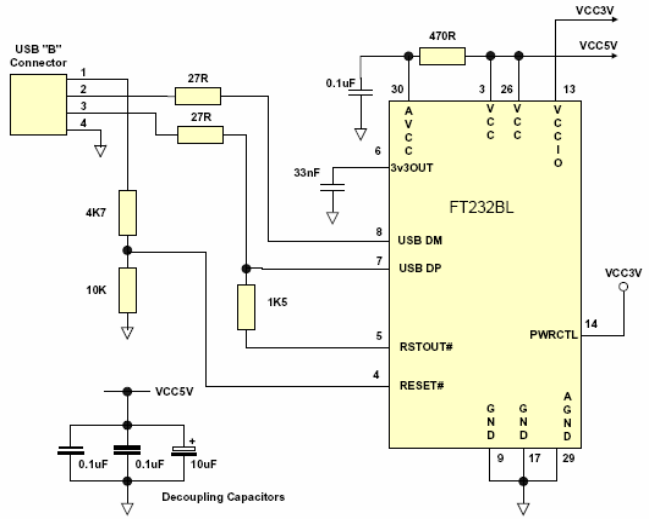


Figura 5-17: USB Self Powered with 3.3V. logic drive Configuration

Se usará la configuración USB Self Powered con interfaz de 3.3V., en el modelo B (para que la tensión de las comunicaciones sea compatible con la de la pantalla) y la configuración USB Self Powered en el modelo A del módulo de comunicaciones.

5.4 Módulo de comunicaciones modelo A

5.4.1. Características necesarias para el microcontrolador

Al igual que el módulo de locomoción y el módulo US&IR, el modelo A del módulo de comunicaciones contiene además de elementos simples como resistencias, condensadores, conectores, etc., un elemento de gran complejidad que se encarga de gestionar todo el módulo, el microcontrolador.

El microcontrolador es el núcleo del módulo y su tarea es interpretar de manera adecuada las órdenes que recibe del módulo maestro por I^2C. Debe convertir dichas órdenes en señales comprensibles para que se transmita y reciba de manera adecuada la información, por uno de los tres puertos posibles (serie, USB, I^2C), que se requirió por parte del maestro.

Para llevar a cabo esta tarea, es necesario que el microcontrolador del modelo A del módulo de comunicaciones, además de cumplir los requisitos genéricos definidos en el apartado 2.1 cumpla otra serie de requisitos que a continuación se detallan:

- **USART**

Tanto para el uso de las comunicaciones serie como para el uso de las comunicaciones por USB es deseable usar un módulo USART ya que simplificaría enormemente todo el proceso de comunicación permitiendo que el microcontrolador pudiera dedicar sus recursos de manera más eficiente. Así pues, se considera un requisito necesario que el microcontrolador disponga de al menos un módulo USART.

- **Frecuencia de operación**

En el caso de las comunicaciones serie mediante el módulo USART, la frecuencia del microcontrolador está directamente relacionada con los errores que se pueden producir en la transmisión. Es por esta razón por lo que necesitamos usar una frecuencia concreta. En este caso, se ha decidido usar una frecuencia de 7,3728MHz.[49] de modo que es necesario que el microcontrolador escogido sea capaz de soportar un cristal oscilador de tal frecuencia.

[49] Se trata de una frecuencia no demasiado baja (> 3,684MHz. es la menor frecuencia que produce error nulo) ni demasiado alta (< 20MHz. es la máxima frecuencia a la que trabajan muchos microcontroladores de baja gama) que produce error nulo en la comunicación a frecuencias de transmisión menores de 250kbps tal como se muestra en el anexo C.

5.4.1.1. Elección del microcontrolador

Para el modelo A del módulo de comunicaciones se ha decidido trabajar con el microcontrolador ATtiny2313 de Atmel porque cumple todos los requisitos mencionados anteriormente, en concreto:

Económico: Cuesta alrededor de 1 euro (al comprar 10 unidades).

Pequeño: En este caso dispone de 18 pines de entrada / salida para interactuar con el medio, lo cual hace que el micro sea a la vez que pequeño funcional. Además está dotado de una memoria flash de 2kbytes, suficiente para albergar el modelo A de comunicaciones.

USART: El microcontrolador dispone de un módulo USART capaz de transmitir tanto en modo síncrono como asíncrono. Permite seleccionar el tipo de paridad, el número de bits de datos y el número de bits de parada. Está dotado además de filtrado de ruido y detección de errores. Además se trata del microcontrolador con USART disponible más pequeño.

Frecuencia de operación: Su frecuencia de operación es de hasta 20MHz. con una alimentación de 5V., y por tanto podremos usar un cristal oscilador de 7.3728MHz. para retransmitir y recibir información de manera fiable.

Comunicación I^2C: No tiene implementado propiamente el protocolo de I^2C pero se puede emular mediante el uso del protocolo USI.

Programación en lenguaje C: Puede programarse en lenguaje C.

No solapamiento: Existe solapamiento entre los pines que gestionan la comunicación I^2C y los pines que gestionan la comunicación SPI, pero puesto que ambas comunicaciones no serán simultaneas en condiciones normales de uso, es permisible su coexistencia para la que se han añadido jumpers cuya funcionalidad y detalles se explicarán en el apartado 5.4.2.

5.4.2. Diseño hardware

Para explicar con mayor claridad y detalle cada una de las partes hardware que componen el modelo A del módulo de comunicaciones se ha representado con ayuda de los programas *Eagle 3D* y *POV Ray* una imagen tridimensional de éste módulo (Figura 5-18).

Aunque en el apartado 5.4.4.1 se muestra una imagen real del módulo construido, y en el anexo G.3 se muestran tanto el esquemático como el layout, se ha considerado que la imagen en 3D permite una mayor claridad y resolución del hardware que facilita una mejor explicación de cada una de las partes de las que se compone el PCB.

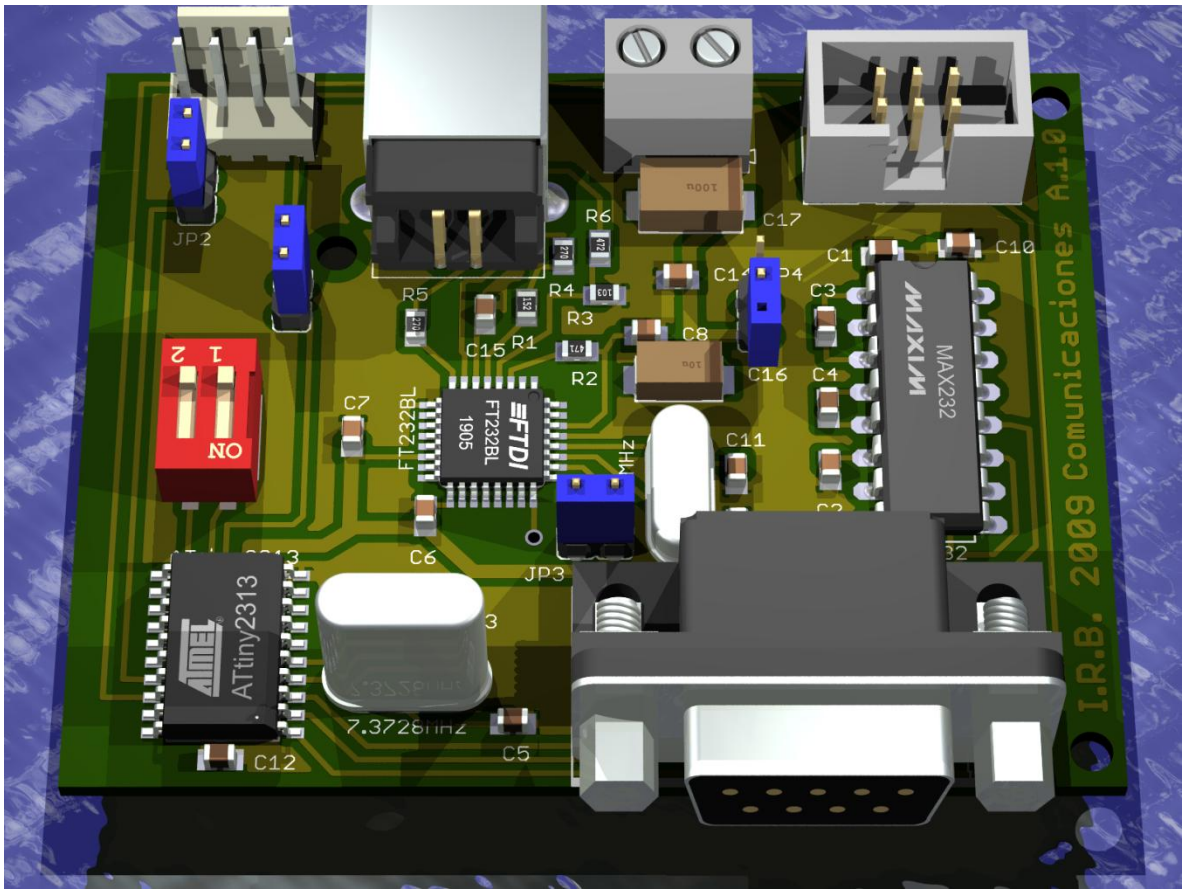


Figura 5-18: Imagen 3D del modelo A del módulo de comunicaciones

A continuación, en las páginas siguientes se ampliará y comentará la función de cada una de las partes del módulo representado.

En primer lugar, en la Figura 5-19, una ampliación de la parte superior en la que se puede observar la circuitería dedicada a la programación y comunicación.

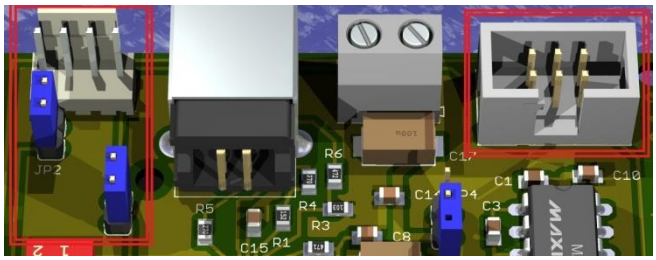


Figura 5-19: SPI e I^2C del modelo A del módulo de comunicaciones

- Recuadrado en rojo a la derecha aparece el conector encargado de las comunicaciones SPI, protocolo del que se habló en el apartado 2.3 y cuya función es la de establecer comunicación entre el microcontrolador y un ordenador para modificar el firmware del modelo A del módulo de comunicaciones. Los pines que lo componen son de arriba abajo y de derecha a izquierda GND, MOSI, +5V., $\overline{RESET}$, SCK y MISO.

- Recuadrado en rojo a la izquierda aparece el conector encargado de las comunicaciones I^2C, protocolo del que se hablo en el apartado 2.2 y cuya función es la de establecer comunicación entre el modelo A del módulo de comunicaciones y el módulo maestro. Los pines que lo componen son de derecha a izquierda SCL, SDA, GND y $\overline{RESET}$.

  Se ha añadido la señal de $\overline{RESET}$ para posibilitar un reseteo general de todos los módulos de manera simultánea en vez de resetear de manera independiente cada uno.

  En el microcontrolador ATtiny2313, los pines SDA y SCL del protocolo I^2C están solapados con los pines MOSI y SCK del protocolo SPI. Para que este solapamiento no produzca problemas de comunicación, se han añadido los jumpers JP1 y JP2 situados cerca del conector I^2C. Así pues, en condiciones normales de funcionamiento JP1 y JP2 deben estar colocados para que las comunicaciones I^2C tengan éxito y deben eliminarse cuando deseen usarse las comunicaciones SPI (actualización del firmware).

En segundo lugar se presenta en la Figura 5-20 una ampliación de todo el PCB en la que se observan dos de los puertos de comunicaciones del módulo.

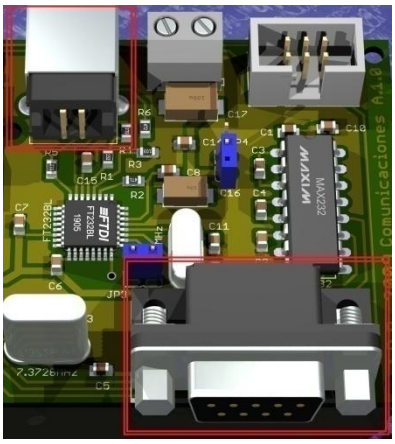


Figura 5-20: Conectores del modelo A

- Aparecen aquí dos conectores hembra que permiten al módulo comunicarse con el exterior. En la parte superior aparece el conector USB tipo B y en la parte inferior el conector DB9 para comunicaciones serie RS-232. Los contactos del conector USB son de arriba abajo y de derecha a izquierda VCC, Data-, GND y Data+. Los contactos del conector DB9 son de arriba abajo y de derecha a izquierda 1.DCD, 2.RxD, 3.TxD, 4.DTR, 5.GND, 6.DSR, 7.RTS, 8.CTS y 9.RI; aunque en este módulo sólo se tienen en cuenta 3.TXD, 2.RXD y 5.GND.

En tercer lugar se presenta en la Figura 5-21 de nuevo otra ampliación de la parte superior en la que se ve la entrada al módulo.

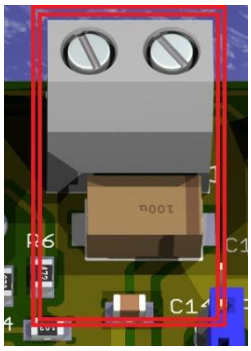


Figura 5-21: Entrada al módulo

- Lo que se aprecia aquí es una clema de dos entradas. Se trata de la alimentación del circuito y es por ello que aparecen ahí los condensadores de desacoplo de 100nF. y 100µF. Las entradas a la clema son de derecha a izquierda: entrada de 5V. y GND.

En cuarto lugar se presenta en la Figura 5-22 una ampliación de la parte lateral derecha del módulo, dedicada a las comunicaciones serie.

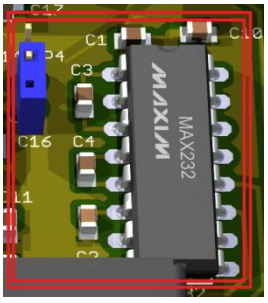


Figura 5-22: IC MAX232

- Aquí se aprecia el integrado MAX232 de MAXIM. Como ya se explicó en el apartado 5.3.1 es necesario un integrado que convierta los niveles TTL[50] con los que trabajamos, a los niveles de tensión del protocolo serie. Los condensadores que aparecen de 100nF. son los recomendados por el fabricante.

[50] No se trata de niveles TTL como tales, ya que estamos trabajando en este caso particular a 3V forzados por los requerimientos de la pantalla LCD.

En la esquina superior izquierda aparece el jumper JP4; su función es dar o quitar la alimentación al MAX232. Esto es muy útil, ya que no sólo ahorramos energía al no alimentarlo mientras no es usado, sino que conseguimos que no interfiera en el caso de que se esté estableciendo comunicación por el puerto USB[51]. Así pues, JP4 debe estar colocado si se desea establecer comunicación por el puerto RS-232 y no colocado en cualquier otro caso.

En quinto lugar se presenta en la Figura 5-23 una ampliación de la parte central del módulo, dedicada a las comunicaciones USB.

- Lo que se muestra aquí es el driver encargado de la gestión de las comunicaciones USB, el FT232BL que ya se comentó en el apartado 5.3.2. Este driver se está usando en modo USB Self Powered de 5V. Las resistencias y condensadores que lo rodean son necesarios para un correcto funcionamiento del integrado. Además también necesita un cristal oscilador de 6MHz. para gestionar correctamente las comunicaciones y es el que se muestra en la parte derecha.

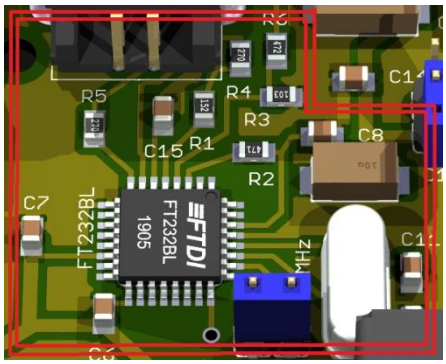


Figura 5-23: FT232BL de FTDI

- También puede observarse un jumper, cuya función es, al igual que ocurría con el MAX232, dar o quitar la alimentación al integrado de manera que se consiga por una parte no interferir con las comunicaciones serie y por otra ahorrar energía. Así pues, en el caso de que se usen las comunicaciones por el puerto USB el jumper JP3 que se muestra debe estar colocado. Dicho jumper no debe estar colocado en ningún otro caso.

[51] La interferencia ocurre en nuestro caso particular ya que no trabajamos con un puerto USB real sino un puerto virtual, que requiere el uso de las señales RxD y TxD del puerto serie. Al estar el puerto serie y el puerto USB multiplexados en nuestro módulo de comunicaciones se producen interferencias si se usan al mismo tiempo ambos puertos.

En sexto y último lugar se presenta, en la Figura 5-24, una ampliación de la parte central que alberga la "inteligencia" del módulo.

- Recuadrado en rojo aparece, en el centro, el microcontrolador, en este caso el ATtiny2313 de ATMEL, tal como se justificó en el apartado 5.4.1.1.

  El microcontrolador es el encargado de recibir las órdenes del maestro por I^2C y actuar en consecuencia en este caso transmitiendo / recibiendo información. Junto a él aparece un condensador de desacoplo, C12, de 100nF.

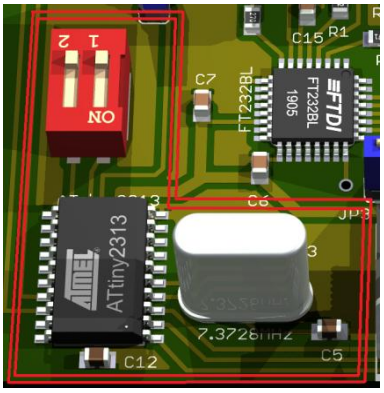


Figura 5-24: Núcleo del módulo

- Lo que aparece sobre el microcontrolador es lo que se ha denominado "selector de esclavo" y su función es la de completar la dirección I^2C del módulo.

  Tal como se comentó en el apartado 5.1, los dos últimos bits de la dirección del módulo de comunicaciones se modifican de acuerdo a la posición de dicho selector, pudiendo obtener de esta manera 4 direcciones diferentes correspondientes a las diferentes posiciones. Si el selector no existiera y quisiéramos usar más de un puerto USB, pensaríamos en replicar el módulo de comunicaciones, no obstante, cuando el maestro quisiera transmitir información por uno solo de los puertos, se estaría transmitiendo la información por todos ellos ya que todos responderían a la misma dirección I^2C; sin embargo, gracias al selector de esclavo podríamos dotar a cada PCB de una dirección distinta sin necesidad de cambiar el firmware pudiendo así usar varios puertos USB/serie de manera independiente.

  El maestro debe, a la hora de emitir cada orden, especificar el esclavo al que se refiere dentro de un mismo módulo. De modo que una posible orden sería *"transmitir el carácter 'a' por el puerto USB del esclavo número dos del modelo A del módulo de comunicaciones"*.

  En la Tabla 5-5 se muestran la posiciones que deberían tener los switches del selector de esclavo dependiendo del esclavo al que el maestro desee dirigirse.

Número de esclavo	Switch 1	Switch 2	Dirección I^2C
Cero	ON	ON	0x10
Uno	OFF	ON	0x11
Dos	ON	OFF	0x12
Tres	OFF	OFF	0x13

Tabla 5-5: Posiciones de los switches del selector en función del esclavo de comunicaciones deseado

- Cerca del microcontrolador se puede observar el cristal oscilador de 7.3728MHz. que se usa para que las comunicaciones serie y USB se transmitan a ciertas velocidades con un 0,0% de error tal como se deduce de las tablas del anexo C. Junto a él se hayan dos condensadores de 22pF. (C5 y C13) necesarios para su correcto funcionamiento.

5.4.3. Diseño software

En este apartado pretenden mostrarse al lector las funciones que componen el modelo A del módulo de comunicaciones. Además del prototipo y de una detallada explicación de cada una de ellas, se explica la manera de acceder a ellas desde el exterior.

Recordemos que el modelo A del módulo de comunicaciones recibe las órdenes mediante el protocolo de comunicaciones I^2C. De modo que es necesario conocer el orden y la forma de enviar los parámetros de las funciones.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
inicializa_USART	uint16_t uint8_t uint8_t	void	Condición de START Escribir en el esclavo 000100xx Transmitir el comando 0xA1 Transmite MSB de bps Transmite LSB de bps Transmitir el dato paridad Transmitir el dato stop Condición de STOP

La tarea de esta función es configurar la USART de acuerdo a ciertos parámetros que puede decidir el usuario, tales como la velocidad en baudios, el tipo de paridad o el número de bits de parada.

El parámetro bps puede tomar un valor entre 0 y 65535. Dicho valor indica la velocidad en baudios a la que se desea configurar las comunicaciones de la USART. Los valores típicos son: 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200 y 230400.

El parámetro paridad puede tomar el valor 0x00, 0x20 ó 0x30. En el caso de tomar el valor 0x00 no se usará paridad en la transmisión, en el caso de tomar el valor 0x20 se usará paridad par en la transmisión y en el caso de tomar el valor 0x30 se usará paridad impar en la transmisión.

El parámetro stop puede tomar el valor 0x00 ó 0x08. Este parámetro indica el número de bits de parada que se usarán para configurar la transmisión. En el caso de que el parámetro tome el valor 0x00 se usará un bit de parada, en el caso de que el parámetro tome el valor 0x08 se usarán dos bits de parada.

Por ejemplo, para configurar una comunicación a 57600 baudios con paridad par y dos bits de parada habría que ejecutar *inicializa_USART (57600, 0x20, 0x08)*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
transmite_USART	uint8_t	void	Condición de START Escribir en el esclavo 000100xx Transmitir el comando 0xA2 Transmitir el dato caracter Condición de REPEATED START Leer del esclavo 000100xx Recibe 0x00 Condición de STOP

La tarea de esta función es transmitir un byte por la USART, se trata de un método bloqueante, de modo que no se iniciara una transferencia hasta que la anterior haya acabado, quedándose a la espera la propia función en caso necesario.

El parámetro carácter puede tomar un valor entre 0 y 255. Dicho valor representa el código ASCII del carácter que se desea transmitir por la USART.

Por ejemplo, para transmitir la letra Ñ habría que ejecutar *transmite_USART (‘Ñ’)*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
recibe_USART	void	uint8_t	Condición de START Escribir en el esclavo 000100xx Transmitir el comando 0xA3 Condición de REPEATED START Leer del esclavo 000100xx Recibe carácter Condición de STOP

La tarea de esta función es recibir un byte por la USART, se trata de un método bloqueante, de modo que no se saldrá de la función hasta que un byte haya sido recibido.

La función retornará un byte cuyo valor corresponde al carácter recibido por la USART.

Por ejemplo, para escribir un mensaje al detectar la primera ‘a’ de una transmisión habría que ejecutar *while (recibe_USART () != ‘a’); printf (“Se ha detectado la ‘a’”);*

Una vez detalladas cada una de las funciones, se procede a mostrar en la Figura 5-25 el diagrama de flujo de la función principal del modelo A del módulo esclavo de comunicaciones. En él puede verse tanto el modo en que se reciben datos por las comunicaciones I^2C como el modo en que se realizan las diversas llamadas a las funciones correspondientes.

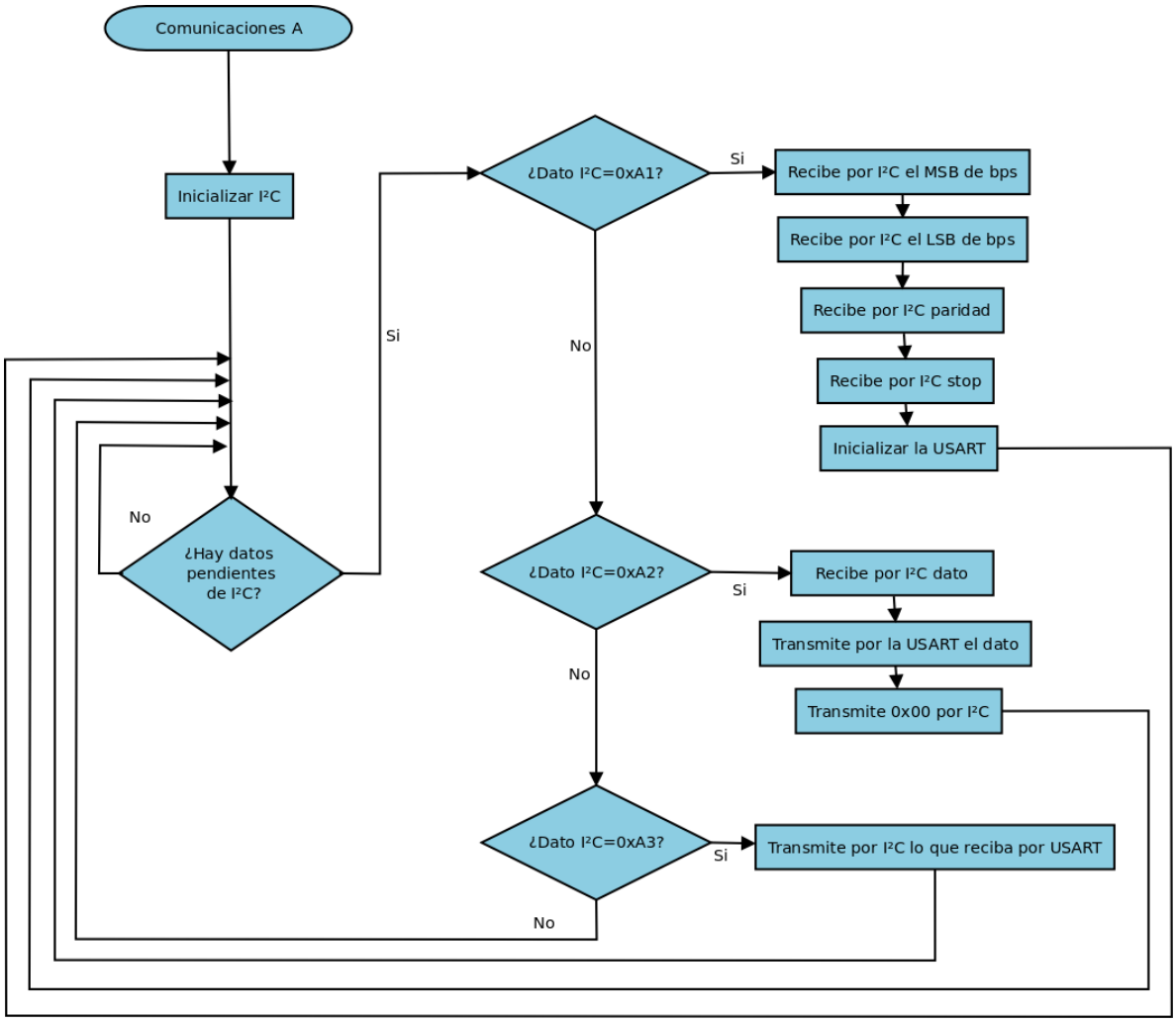


Figura 5-25: Diagrama de flujo del programa principal del modelo A del módulo de comunicaciones

5.4.4. Construcción e información técnica

5.4.4.1. Fase de prototipado del módulo

Antes de construir de manera definitiva el modelo A del módulo de comunicaciones, se hizo un prototipo de éste (Figura 5-26), con el objetivo de encontrar posibles errores y corregir problemas de diseño. Para el prototipo se usó el chip FT232BL del que se habló en el apartado 5.3.2 en encapsulado de montaje superficial y es por ello que aparece rodeado de pegamento y con cables soldados a él. Al igual que en los dos prototipos construidos anteriormente se ha empleado la técnica de wrapping para la unión de los componentes.

Una vez probado el correcto funcionamiento del prototipo se formalizo el diseño dando lugar a la construcción del módulo definitivo.

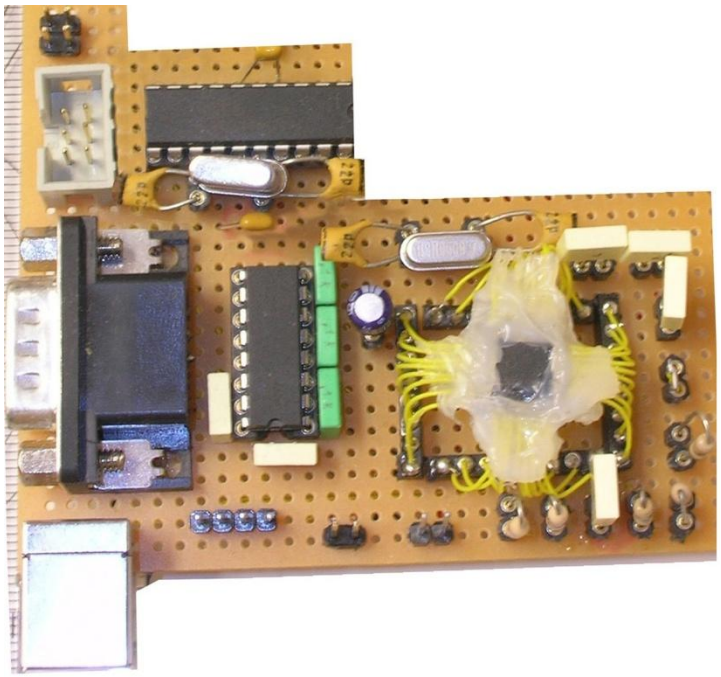


Figura 5-26 Prototipo del modelo A del módulo de comunicaciones

5.4.4.2. Especificaciones, aspecto y presupuesto del módulo

En este apartado se muestra en primer lugar información acerca de las especificaciones técnicas del módulo, necesarias para saber por ejemplo que tipo de puertos contiene o la corriente máxima que consume el módulo; dicha información se aprecia en la Tabla 5-6.

Especificaciones técnicas del modelo A del módulo de comunicaciones	
Tensión necesaria	5V.
Corriente máxima	43,85mA.
Puertos de comunicaciones	RS-232 hembra, USB tipo B hembra e I^2C
Dimensiones	55,2450mm. de alto y 69,5325mm. de ancho (38,4132cm^2.)

Tabla 5-6: Especificaciones técnicas del módulo de comunicaciones A

En segundo lugar se muestra, en la Figura 5-27, el modelo A del módulo de comunicaciones una vez construido en su versión definitiva.

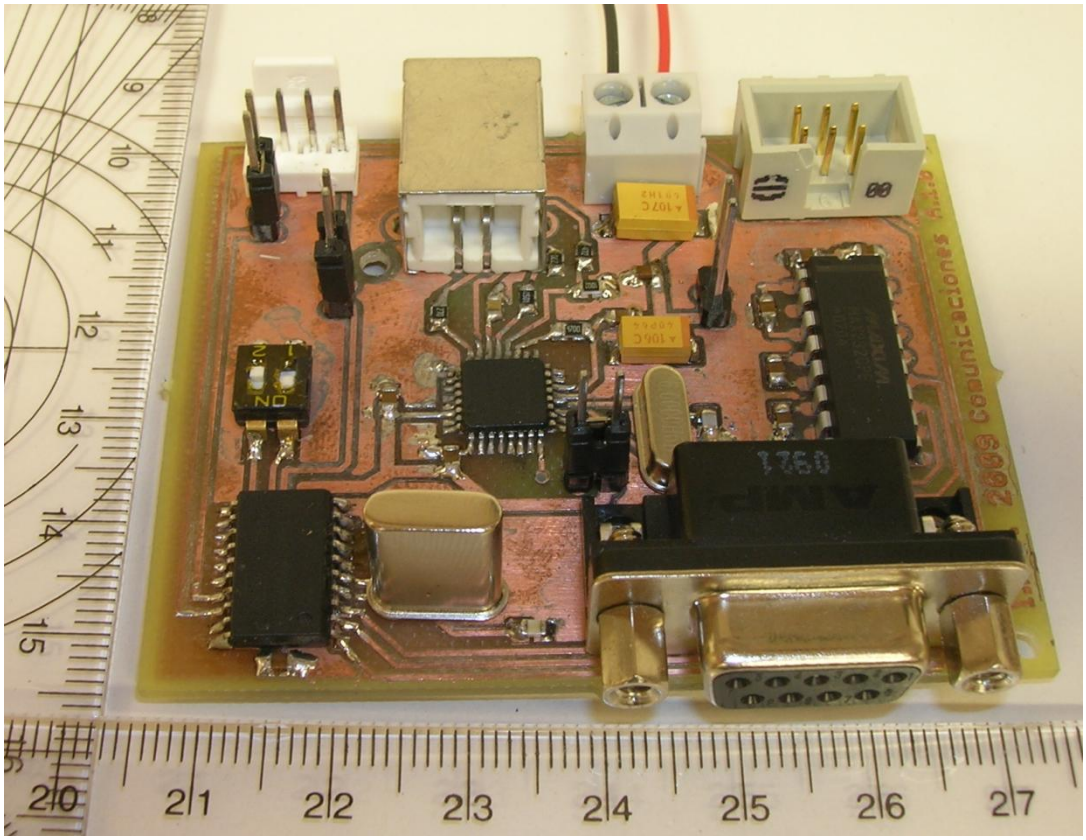


Figura 5-27: Imagen real del modelo A del módulo de comunicaciones

En tercer lugar se hace constar el precio que costaría construir el módulo de nuevo. En la Tabla 5-7, se muestra el precio de cada componente necesario para realizar el montaje del modelo A del módulo de comunicaciones. Los precios han sido extraídos de Merchan electrónica y componentes[52], exceptuando algunos de ellos, que por no encontrarse en el distribuidor anterior se han consultado en Farnell[53], estos últimos están en color azul. Los que no se han encontrado en ninguno de los dos anteriores han sido consultados en RS[54], en este caso, han sido coloreados en naranja.

Referencia	Encapsulado	Componente	Valor	Precio
C1…4, C6...8, C10, C12, C14	0805	Condensador	100nF	10x0.0116€
C5, C13	0805	Condensador	22pF	2x0.022€
C9, C11	0805	Condensador	27pF	2x0.016€
C15	0805	Condensador	33nF	1x0.024€
C16	6032	Condensador	10µF	1x0.33€
C17	7343	Condensador	100µF	1x1.32€
R2	0805	Resistencia	470Ω	1x0.0038€
R3	0805	Resistencia	10kΩ	1x0.008€
R6	0805	Resistencia	4k7Ω	1x0.006€
R1	0805	Resistencia	1k5Ω	1x0.0028€
R4, R5	0805	Resistencia	27Ω	2x0.006€
ATtiny2313	SO20	Microcontrolador	ATtiny2313	1x2.27€
MAX232	DIL16	Driver	MAX232	1x1.1€
FT232BL	QFP32	Driver	FT232BL	1x7.2€
JP1…4		Jumper	Tira min. 8 pines	1x0.55€
Q1	HC49/S	Cristal oscilador	7,3728MHz	1x1.06€
Q2	HC49/S	Cristal oscilador	6MHz	1x1.10€
DB9	DB9	Conector	Serie Hembra	1x1.05€
USB		Conector	USB tipo B Hembra	1x0.48€
CONECTOR SPI		Conector	SPI	1x1.02€
J1	Molex 2,54mm	Conector	I^2C	1x0.479€
Esclavo	DIP (DIL)	Switch SMD	Selector de esclavo	1x1.28€
X2	Pasante 2 vías	Clema 5mm	Bornes para PCB	1x0.27€
Total				**19,7576€**

Tabla 5-7: Precio de los componentes del módulo de comunicaciones A

Teniendo en cuenta que construir cada $dm.^2$ de PCB cuesta 60€. Se concluye que al coste de los componentes hay que sumar 60€ por construcción.

[52] Los precios están sujetos a modificaciones y deben comprobarse en http://www.e-merchan.com/

[53] Los precios están sujetos a modificaciones y deben comprobarse en http://es.farnell.com/

[54] Los precios están sujetos a modificaciones y deben comprobarse en http://es.rs-online.com/

5.5 Módulo de comunicaciones modelo B

5.5.1. Características necesarias para el microcontrolador

Al igual que todos los módulos anteriores, el modelo B del módulo de comunicaciones contiene además de elementos simples como resistencias, condensadores, conectores, etc., un elemento de gran complejidad que se encarga de gestionar todo el módulo, el microcontrolador.

El microcontrolador es el núcleo del módulo y tiene, en este caso, la tarea de interpretar de manera adecuada las órdenes que recibe del módulo maestro por I^2C, convirtiendo dichas órdenes en señales comprensibles bien para que se transmita y reciba de manera adecuada la información, por uno de los tres puertos posibles (serie, USB ó I^2C) o bien para interactuar con la pantalla LCD.

A diferencia del resto de los módulos el microcontrolador debe ser capaz también de manejar la pantalla LCD sin necesidad del módulo maestro, sino recibiendo directamente órdenes desde un PC.

Para llevar a cabo esta tarea, es necesario que el microcontrolador del modelo B del módulo de comunicaciones, además de cumplir los requisitos genéricos definidos en el apartado 2.1 cumpla otra serie de requisitos que a continuación se detallan:

- **USART**

Tanto para el uso de las comunicaciones serie como para el uso de las comunicaciones por USB es deseable usar un módulo USART ya que simplificaría enormemente todo el proceso de comunicación permitiendo que el microcontrolador pudiera dedicar sus recursos de manera más eficiente. Así pues, se considera un requisito necesario que el microcontrolador disponga de al menos un módulo USART.

- **Canales de PWM**

Como se explico en el apartado 5.2.3, para poder generar en la pantalla la gama de colores RGB se necesitan tres canales de PWM, un canal para controlar la cantidad de color rojo, otro para controlar la cantidad de verde y un tercero para controlar la cantidad de azul.

- **Frecuencia de operación**

En el caso de las comunicaciones serie mediante el módulo USART, la frecuencia del microcontrolador está directamente relacionada con los errores que se pueden producir en la transmisión. Es por esta razón por lo que necesitamos usar una frecuencia concreta. En este caso, se ha decidido usar una frecuencia de 7,3728MHz.(55) de modo que necesitamos que el microcontrolador escogido sea capaz de soportar un cristal oscilador de tal frecuencia.

5.5.1.1. Elección del microcontrolador

Para el modelo B del módulo de comunicaciones se ha decidido trabajar con el microcontrolador ATmega48 de Atmel porque cumple todos los requisitos mencionados anteriormente, en concreto:

Económico: Cuesta alrededor de 1,5 euros (al comprar 10 unidades).

Pequeño: El tamaño de los microcontroladores es generalmente proporcional al número de pines de estos, así a menor número de pines menor tamaño del integrado. En este caso dispone de 23 pines de entrada / salida para interactuar con el medio, lo cual hace que el micro sea a la vez que pequeño funcional(56). Además está dotado de una memoria flash de 4kbytes, capacidad suficiente para albergar el módulo de comunicaciones.

USART: El microcontrolador dispone de un módulo USART capaz de transmitir tanto en modo síncrono como asíncrono. Permite seleccionar el tipo de paridad, el número de bits de datos y el número de bits de parada. Está dotado además de filtrado de ruido y detección de errores.

Canales de PWM: Este micro dispone de seis canales de PWM.

Frecuencia de operación: Su frecuencia de operación es de hasta 10MHz. con una alimentación de 3V., y por tanto podremos usar un cristal oscilador de 7,3728MHz. para enviar y recibir datos mediante la USART a numerosas velocidades con un error nulo.

---

[55] Se trata de una frecuencia no demasiado baja (> 3,684MHz. es la menor frecuencia que produce error nulo) ni demasiado alta (< 10/16MHz. es la máxima frecuencia a la que trabajan muchos microcontroladores de baja gama a tensiones de 3V.) que produce error nulo en la comunicación a frecuencias de transmisión menores de 250kbps. tal como se muestra en el anexo C.

[56] Es el microcontrolador con menor número de patas con capacidad de más de 2kbytes de memoria, dotado de USART y de I^2C.

Comunicación I^2C: No tiene implementado propiamente el protocolo de I^2C pero se puede emular mediante el uso del protocolo TWI.

Programación en lenguaje C: Puede programarse en lenguaje C.

No solapamiento: No existe solapamiento entre los pines que gestionan la USART y los pines que gestionan la comunicación I^2C. No existe solapamiento entre los pines que gestionan la comunicación I^2C y los pines que gestionan la comunicación SPI.

5.5.2. Diseño hardware

Para explicar con mayor claridad y detalle cada una de las partes hardware que componen el modelo B del módulo de comunicaciones se ha representado con ayuda de los programas *Eagle 3D* y *POV Ray* una imagen tridimensional de éste módulo (Figura 5-28).

Aunque en el apartado 5.5.4.1 se muestra una imagen real del módulo construido, y en el anexo G.4 se muestran tanto el esquemático como el layout, se ha considerado que la imagen en 3D permite una mayor claridad y resolución del hardware que facilita una mejor explicación de cada una de las partes de las que se compone el PCB.

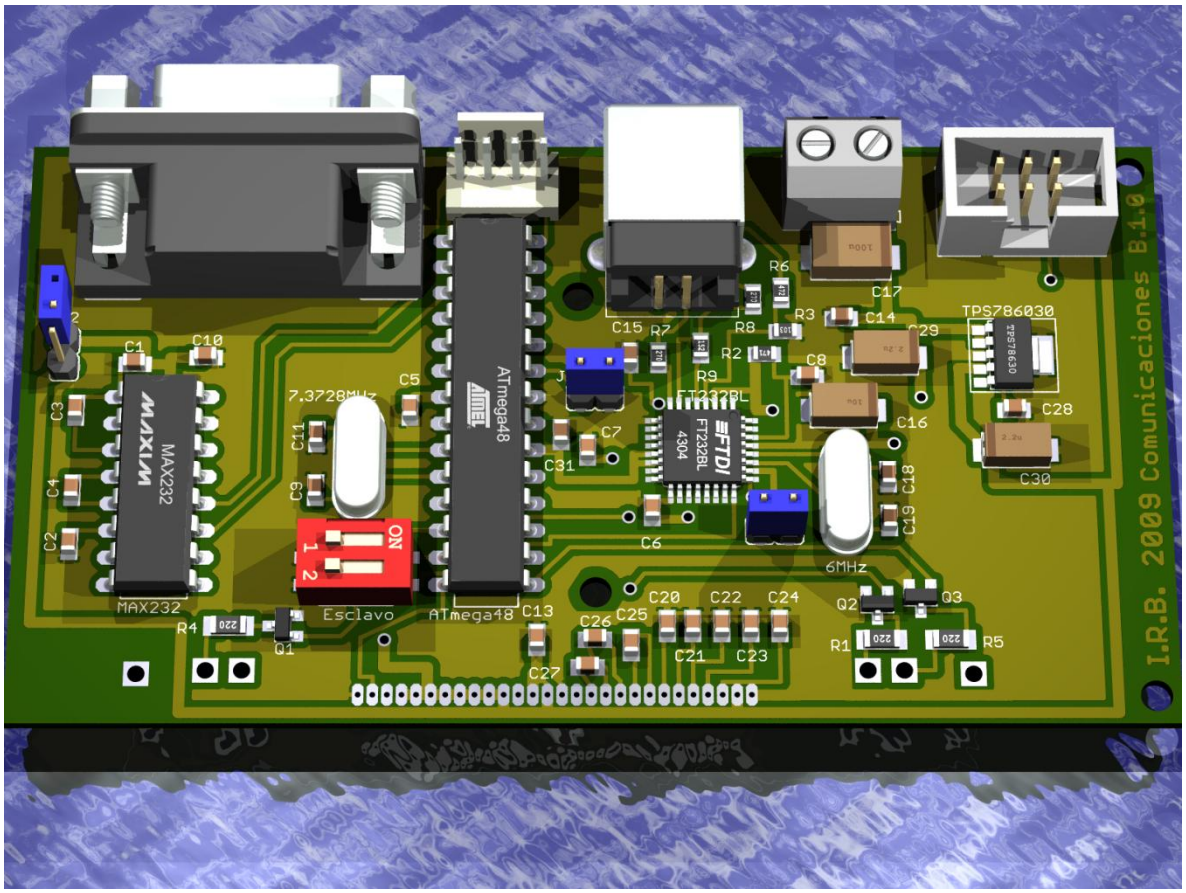


Figura 5-28: Imagen 3D top del modelo B del módulo de comunicaciones

Otra de las particularidades de este módulo con respecto a los anteriores es que tiene elementos a ambas caras del PCB. Concretamente, en la otra cara del PCB se halla además de un condensador, la pantalla LCD comentada en el apartado 5.2.3. Es por esta razón que se presenta en la Figura 5-29 una imagen virtual tridimensional de la otra cara del PCB.

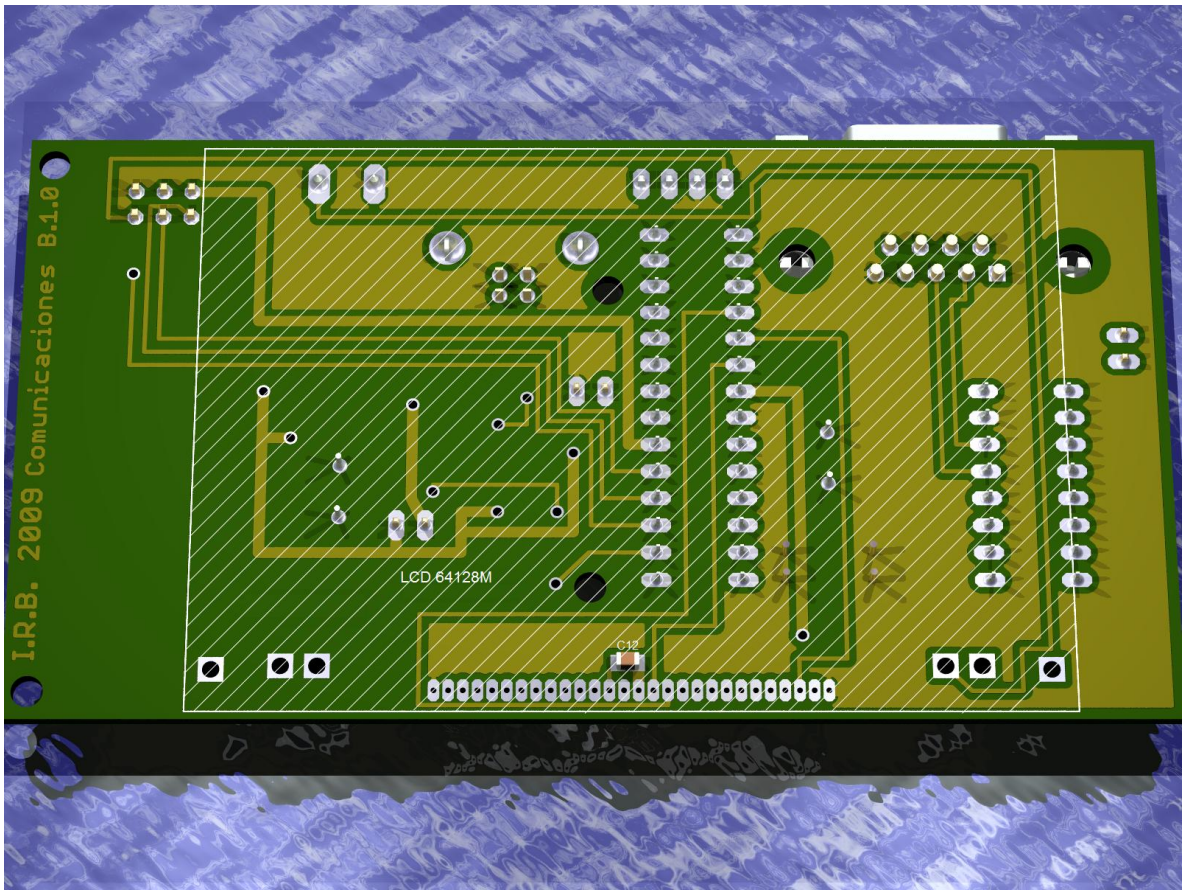


Figura 5-29: Imagen 3D bottom del modelo B del módulo de comunicaciones

Tras mostrar el PCB del módulo B de comunicaciones por ambas caras, se procede a presentar, en primer lugar, en la Figura 5-30, una ampliación de la parte superior en la que se puede observar la circuitería dedicada a la programación y comunicación.

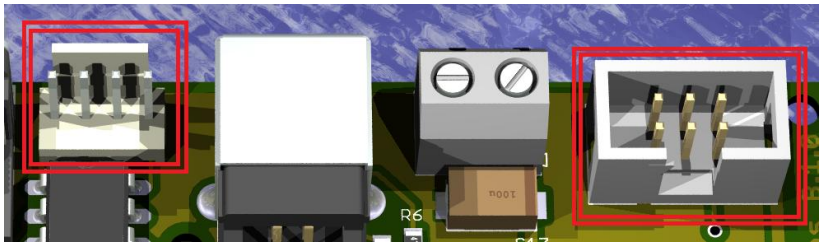


Figura 5-30: SPI e I^2C del modelo B del módulo de comunicaciones

- Recuadrado en rojo a la derecha aparece el conector encargado de las comunicaciones SPI, protocolo del que se habló en el apartado 2.3 y cuya función es la de establecer comunicación entre el microcontrolador y un ordenador para modificar el firmware del modelo B del módulo de comunicaciones. Los pines que lo componen son de arriba abajo y de derecha a izquierda GND, MOSI, +5V., $\overline{RESET}$, SCK y MISO.

- Recuadrado en rojo a la izquierda aparece el conector encargado de las comunicaciones I^2C, protocolo del que se hablo en el apartado 2.2 y cuya función es la de establecer comunicación entre el modelo B del módulo de comunicaciones y el módulo maestro. Los pines que lo componen son de derecha a izquierda SCL, SDA, GND y $\overline{RESET}$.

  Se ha añadido la señal de $\overline{RESET}$ para posibilitar un reseteo general de todos los módulos de manera simultánea en vez de resetear de manera independiente cada uno.

En segundo lugar se presenta en la Figura 5-31 otra ampliación de la parte superior en la que se observan dos de los puertos de comunicaciones del módulo.

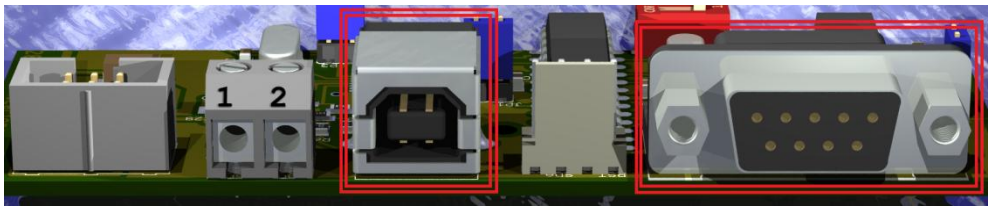


Figura 5-31: Vista frontal del modelo B del módulo de comunicaciones

- Aparecen aquí dos conectores hembra que permiten al módulo comunicarse con el exterior. En la parte izquierda aparece el conector USB tipo B y en la parte derecha el conector DB9 para comunicaciones serie RS-232. Los contactos del conector USB son de arriba abajo y de derecha a izquierda VCC, Data-, GND y Data+. Los contactos del conector DB9 son de arriba abajo y de derecha a izquierda 1.DCD, 2.RxD, 3.TxD, 4.DTR, 5.GND, 6.DSR, 7.RTS, 8.CTS y 9.RI; aunque en este módulo sólo se tienen en cuenta 3.TXD, 2.RXD y 5.GND.

En tercer lugar se presenta en la Figura 5-32 de nuevo otra ampliación de la parte superior en la que se ve la entrada al módulo.

- Lo que se aprecia aquí es una clema de dos entradas. Se trata de la alimentación del circuito y es por ello que aparecen ahí los condensadores de desacoplo de 100nF. y 100µF. Las entradas a la clema son de derecha a izquierda: entrada de 5V. y GND.

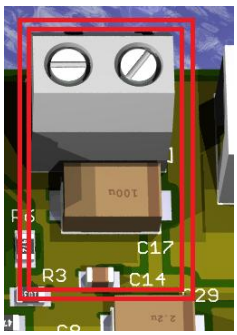


Figura 5-32: Entrada al módulo

En cuarto lugar se presenta en la Figura 5-33 una ampliación de la parte lateral derecha en la que se muestra un regulador de tensión.

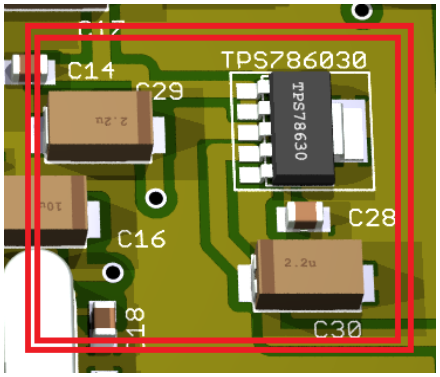


Figura 5-33: Regulador lineal 3V.

- Lo que se muestra aquí es el integrado TPS78630 de Texas Instruments. Le acompañan tres condensadores, dos de ellos de 2,2µF. (C29 y C30) y un tercero de 10nF. (C28) y la utilidad de este integrado es convertir la tensión de 5V. presente en el PCB a 3V. La razón de esta conversión es que la pantalla LCD usada trabaja a una tensión de 3.0V. y por tanto es necesario que las señales conectadas a la pantalla tengan como máximo esa tensión; no obstante, también es necesaria en algunas partes del circuito la tensión de 5V. y por ello en este PCB se trabaja al igual que en el módulo de locomoción con dos tensiones distintas.

En quinto lugar se presenta en la Figura 5-34 una ampliación de la parte lateral izquierda del módulo, dedicada a las comunicaciones serie.

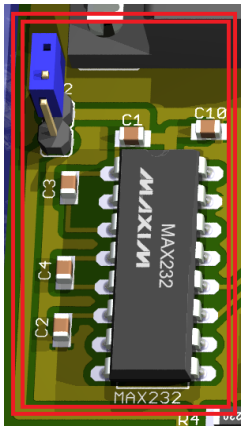


Figura 5-34: IC MAX232

- Lo que se aprecia aquí es el integrado MAX232 de MAXIM. Como ya se explicó en el apartado 5.3.1 es necesario un integrado que convierta los niveles TTL[57] con los que se trabaja, a los niveles de tensión del protocolo serie. Los condensadores que aparecen de 100nF. son los recomendados por el fabricante.

  En la esquina superior izquierda se observa el jumper JP3; su función es dar o quitar la alimentación al MAX232. Esto muy útil, ya que no sólo se ahorra energía al no alimentarlo mientras no es usado, sino que se consigue que no interfiera en el caso de que se esté estableciendo comunicación por el puerto USB[58].

[57] No se trata de niveles TTL como tales, ya que estamos trabajando en este caso particular a 3V forzados por los requerimientos de la pantalla LCD.

[58] La interferencia ocurre en nuestro caso particular ya que no trabajamos con un puerto USB real sino un puerto virtual, que requiere el uso de las señales RxD y TxD del puerto serie. Al estar el puerto serie y el puerto USB multiplexados en nuestro módulo de comunicaciones se producen interferencias si se usan al mismo tiempo ambos puertos.

Así pues, JP3 debe estar colocado si se desea establecer comunicación por el puerto RS-232 y no colocado en cualquier otro caso.

En sexto lugar se presenta en la Figura 5-35 una ampliación de la parte central del módulo, dedicada a las comunicaciones USB.

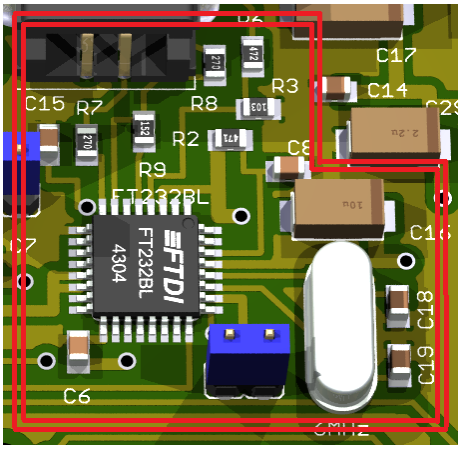


Figura 5-35: FT232BL de FTDI

- Lo que se muestra aquí es el driver encargado de la gestión de las comunicaciones USB, el FT232BL que ya se comentó en el apartado 5.3.2. Este driver se está usando en modo USB Self Powered con compatibilidad para 3.3V.; no obstante, incluso en este modo, necesita que algunos de sus pines estén alimentados a 5V. Es precisamente este integrado el que hace que en el PCB no sólo se manejen los 3V. que requiere la pantalla LCD sino también 5V. Las resistencias y condensadores que lo rodean son necesarios para un correcto funcionamiento del integrado. Además también necesita un cristal oscilador de 6MHz. para gestionar correctamente las comunicaciones y es el que se muestra en la parte derecha.
- También puede apreciarse un jumper, cuya función es, al igual que ocurría con el MAX232, dar o quitar la alimentación al integrado de manera que se consiga por una parte no interferir con las comunicaciones serie y por otra ahorrar energía. Así pues, en el caso de que se usen las comunicaciones por el puerto USB el jumper JP4 que se muestra debe estar colocado. Dicho jumper no debe estar colocado en ningún otro caso.

En séptimo lugar se presenta en la Figura 5-36 una ampliación de la parte inferior central del módulo de comunicaciones, en la Figura 5-37 una ampliación de la parte inferior derecha e izquierda y finalmente en la Figura 5-38 una ampliación de la parte trasera del módulo de comunicaciones.

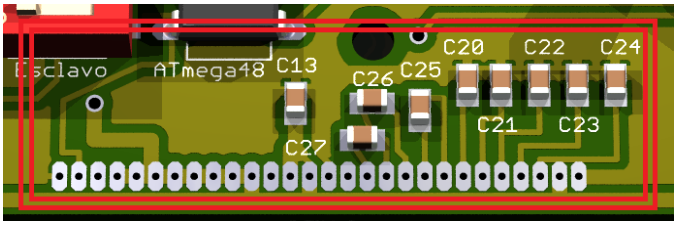


Figura 5-36: Algunos condensadores y pines de la pantalla LCD

Figura 5-37: Transistor y resistencia (Red) a la izquierda y transistor y resistencia (Green y Blue) a la derecha

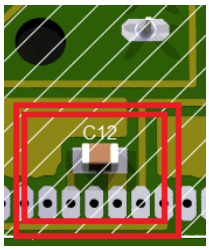


Figura 5-38: Condensador parte trasera

- En la primera y última figuras mostradas anteriormente se observan una serie de condensadores necesarios para que la pantalla funcione correctamente y recomendados por el fabricante de la misma. Los detalles de conexión de los condensadores pueden verse en el esquemático del anexo G.4; además en la última de las figuras se observa el lugar en el que irá colocada la pantalla (mostrado mediante rayas blancas diagonales). En las figuras centrales se muestran tres[59] transistores BSS138N[60] de Infineon Technologies así como tres resistencias de 22Ω, necesarias para el correcto funcionamiento del color de la pantalla.

[59] La razón de que se usen tres es porque el color se genera siguiendo el estándar RGB. Cada tupla transistor-resistencia está conectada a un LED rojo, verde o azul.

[60] Se trata de un transistor de pequeña señal MOSFET de tipo N cuyas características técnicas se especifican en http://www.infineon.com/cms/en/product/findProductTypeByName.html?q=bss138N

En octavo y último lugar se presenta, en la Figura 5-39, una ampliación de la parte central que alberga la "inteligencia" del módulo.

- Recuadrado en rojo aparece, en el centro, el microcontrolador, en este caso el ATmega48 de ATMEL, tal como se justificó en el apartado 5.5.1.1.

  El microcontrolador es el encargado de recibir las órdenes del maestro por I^2C, o bien de otro periférico u ordenador por USB o serie e interpretarlas y actuar en consecuencia en este caso mostrando algo por pantalla o transmitiendo / recibiendo información. Junto a él aparecen dos condensadores de desacoplo C5 y C31 de valor 100nF. ambos. C31 desacopla la tensión analógica y C5 desacopla la tensión digital.

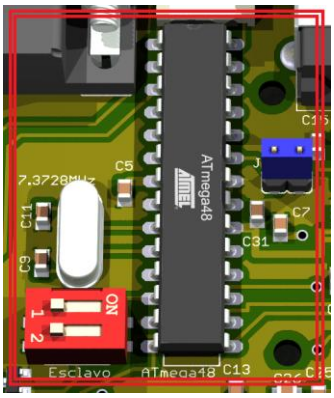


Figura 5-39: Núcleo del módulo

- Lo que aparece a la izquierda del microcontrolador es lo que se ha denominado "selector de esclavo" y su función es la de completar la dirección I^2C del módulo.

  Tal como se comentó en el apartado 5.1, los dos últimos bits de la dirección del módulo de comunicaciones se modifican de acuerdo a la posición de dicho selector, pudiendo obtener de esta manera 4 direcciones diferentes correspondientes a las diferentes posiciones. Si el selector no existiera y quisiéramos manejar más de una pantalla o usar más de un puerto USB, pensaríamos en replicar el módulo de comunicaciones, no obstante, cuando el maestro quisiera mostrar algo por una sola pantalla, se mostraría en todas las pantallas ya que todas responderían a la misma dirección I^2C; sin embargo, gracias al selector de esclavo podríamos dotar a cada PCB de una dirección distinta sin necesidad de cambiar el firmware pudiendo así mostrar contenidos independientes en cada pantalla o usar varios puertos USB/serie.

  El maestro debe, a la hora de emitir cada orden, especificar el esclavo al que se refiere dentro de un mismo módulo. De modo que una posible orden sería *"mostrar por la pantalla del esclavo número dos del modelo B del módulo de comunicaciones la frase 'Esto es una prueba' comenzando en la quinta columna de la segunda fila"*.

En la Tabla 5-8 se muestran la posiciones que deberían tener los switches del selector de esclavo dependiendo del esclavo al que el maestro desee dirigirse.

Número de esclavo	Switch 1	Switch 2	Dirección I^2C
Cero	ON	ON	0x0C
Uno	OFF	ON	0x0D
Dos	ON	OFF	0x0E
Tres	OFF	OFF	0x0F

Tabla 5-8: Posiciones de los switches del selector en función del esclavo de comunicaciones deseado

- Cerca del microcontrolador se puede observar el cristal oscilador de 7.3728MHz. que se usa para que las comunicaciones serie y USB se transmitan a ciertas velocidades con un 0,0% de error tal como se deduce de las tablas de anexo C. Junto a él se hayan dos condensadores de 22pF. (C9 y C11) necesarios para su correcto funcionamiento.
- A la derecha del microcontrolador aparece el jumper JP1, cuya función es determinar si el modelo B del módulo de comunicaciones debe funcionar en modo autogobernado, es decir recibiendo órdenes del PC por el puerto serie o USB sin necesidad de estar el maestro conectado; o si por el contrario, el módulo funcionará en modo gobernado, es decir recibiendo órdenes del maestro por el bus I^2C. En el caso de querer esta última solución, el jumper deberá estar colocado, y en el caso de que deseemos establecer conexión con un PC, el jumper no debe estar colocado, y no deberíamos olvidar colocar el jumper JP3 o JP4 anteriormente explicados. A continuación se muestra una aclaración de los jumper en la Tabla 5-9.

Colocados (SI / NO / X)	**JP1**	**JP3**	**JP4**
Modo I^2C(normal)	SI	X	X
Modo I^2C(solo pantalla)	SI	NO	NO
Modo USB	NO	SI	NO
Modo RS-232	NO	NO	SI

Tabla 5-9: Colocación jumpers del modelo A del módulo de comunicaciones

5.5.3. Diseño software

En este apartado pretenden mostrarse al lector las funciones que componen el modelo B del módulo de comunicaciones. Además del prototipo y de una detallada explicación de cada una de ellas, se explica la manera de acceder a ellas desde el exterior.

Recordemos que el modelo B del módulo de comunicaciones también debe ser capaz de recibir las órdenes mediante el protocolo de comunicaciones I^2C, de modo que, es necesario conocer el orden y la forma de enviar los parámetros de las funciones.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
inicializa_USART	uint16_t uint8_t uint8_t	void	Condición de START Escribir en el esclavo 000011xx Transmitir el comando 0xA1 Transmite MSB de bps Transmite LSB de bps Transmitir el dato paridad Transmitir el dato stop Transmite fin de cadena '\0' Condición de STOP

La tarea de esta función es configurar la USART de acuerdo a ciertos parámetros que puede decidir el usuario, tales como la velocidad en baudios, el tipo de paridad o el número de bits de parada.

El parámetro bps puede tomar un valor entre 0 y 65535. Dicho valor indica la velocidad en baudios a la que se desea configurar las comunicaciones de la USART. Los valores típicos son: 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, 115200 y 230400.

El parámetro paridad puede tomar el valor 0x00, 0x20 ó 0x30. En el caso de tomar el valor 0x00 no se usará paridad en la transmisión, en el caso de tomar el valor 0x20 se usará paridad par en la transmisión y en el caso de tomar el valor 0x30 se usará paridad impar en la transmisión.

El parámetro stop puede tomar el valor 0x00 ó 0x08. Este parámetro indica el número de bits de parada que se usarán para configurar la transmisión. En el caso de que el parámetro tome el valor 0x00 se usará un bit de parada, en el caso de que el parámetro tome el valor 0x08 se usarán dos bits de parada.

Por ejemplo, para configurar una comunicación a 57600 baudios con paridad par y dos bits de parada habría que ejecutar *inicializa_USART (57600, 0x20, 0x08)*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
transmite_USART	uint8_t	void	Condición de START Escribir en el esclavo 000011xx Transmitir el comando 0xA2 Transmitir el dato carácter Transmite fin de cadena '\0' Condición de STOP

La tarea de esta función es transmitir un byte por la USART, se trata de un método bloqueante, de modo que no se iniciara una transferencia hasta que la anterior haya acabado, quedándose a la espera la propia función en caso necesario.

El parámetro carácter puede tomar un valor entre 0 y 255. Dicho valor representa el código ASCII del carácter que se desea transmitir por la USART.

Por ejemplo, para transmitir la letra Ñ habría que ejecutar *transmite_USART ('Ñ')*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
recibe_USART	void	uint8_t	Condición de START Escribir en el esclavo 000011xx Transmitir el comando 0xA3 Transmite fin de cadena '\0' Condición de REPEATED START Leer del esclavo 000011xx Recibe carácter Condición de STOP

La tarea de esta función es recibir un byte por la USART, se trata de un método bloqueante, de modo que no se saldrá de la función hasta que un byte haya sido recibido.

La función retornará un byte cuyo valor corresponde al carácter recibido por la USART.

Por ejemplo, para escribir un mensaje al detectar la primera 'a' de una transmisión habría que ejecutar *while (recibe_USART () != 'a'); printf ("Se ha detectado la 'a'");*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Inicializa_LCD	void	void	Condición de START Escribir en el esclavo 000011xx Transmitir el comando 0xA4 Condición de STOP

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior. En concreto, lo que hace la función es:

1. Establece como salidas la señal PD3, que está conectada al pin A0 de la pantalla; la señal PD2, que está conectada al pin SCL de la pantalla y la señal PD4, que está conectada al pin SI.
2. Establece como salidas la señal PB1, que se encarga de controlar el LED verde; la señal PB2 que se encarga de controlar el LED azul; y la señal PD6, que se encarga de controlar el LED rojo de la pantalla. Configura además las tres señales para que generen PWM y se controle de este modo la saturación del color.
3. Establece como salida la señal PB0, que se encarga de generar un reset de la pantalla al ejecutarse esta función. La señal de reset se pone a nivel bajo durante un milisegundo y posteriormente a nivel alto lógico.
4. Envía algunos comandos al controlador de la pantalla para establecer tanto la tensión de referencia como el control de potencia.
5. Establece la dirección en que se mostrarán los valores por la pantalla. Define la coordenada 0,0 y finalmente habilita la pantalla.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Establecer_control_potencia	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior.

La función se encarga de controlar la circuitería interna del regulador de tensión de la pantalla. El proceso requiere de tres llamadas a esta función con las correspondientes máscaras, pasadas como parámetro. Para más detalles véase el código del anexo **¡Error! o se encuentra el origen de la referencia.**.

El proceso es gestionado por la función Inicializa_LCD().

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Establecer_tension_referencia	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior.

La función se encarga de de establecer el valor pasado como parámetro como referencia de tensión para el regulador interno. De esta manera puede controlarse el contraste de la pantalla. El proceso es gestionado por la función Inicializa_LCD().

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
LCD_SetTopPage	uint8_t uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior.

La función se encarga de realizar una serie de cálculos con los dos parámetros pasados para establecer cuál será la pagina que sirva como referencia de coordenada.

El proceso es gestionado por la función Inicializa_LCD() que establece como coordenada 0,0 la esquina superior izquierda.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
LCD_SetScreen	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior.

La función se encarga de dar valor a todos los pixeles de la pantalla siguiendo el patrón definido como parámetro, que determina como son los 8 pixeles de una columna de una página. La función puede usarse para borrar la pantalla pasando la máscara 0x00 o encender todos los píxeles, con la máscara 0xFF.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Seleccion_fila	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior.

La función se encarga de seleccionar una página de las 8 que hay (0...7). Cada página está formada por 8 pixeles → 64 pixeles de alto. La fila se define mediante el parámetro que puede tomar valor entre 0 y 7.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Seleccion_columna	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior.

La función se encarga de seleccionar una página de las 16 que hay (0...15). Cada página está formada por 8 pixeles → 128 pixeles de ancho. La columna se define mediante el parámetro que puede tomar valor entre 0 y 15.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Escribir_comando	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior.

La función se encarga de mandar un comando a la pantalla. Para ello establece la señal de control a nivel bajo y manda el comando mientras se generan pulsos de duración mínima de 50ns. de período. Como parámetro se pasa el comando que desea enviarse. Los comandos están definidos tanto en el datasheet como en el fichero lcd.h que puede verse en el anexo **¡Error! No se encuentra el origen de la referencia.**.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Escribir_dato	uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior.

La función se encarga de mostrar un dato por pantalla. Para ello establece la señal de control a nivel alto y manda el dato mientras se generan pulsos de duración mínima de 50ns. de período. El dato tiene codificados los valores de los píxeles, de modo que lo que se hace en realidad es iluminar o no una serie de ocho píxeles (verticales) en base al dato enviado como parámetro.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Color_pantalla	uint8_t uint8_t uint8_t	void	Condición de START Escribir en el esclavo 000011xx Transmitir el comando 0xA5 Transmitir el dato rojo Transmitir el dato verde Transmitir el dato azul Transmite fin de cadena '\0' Condición de STOP

La tarea de esta función es colorear el fondo de pantalla según la descripción RGB.

El parámetro rojo puede tomar el valor entre 0 y 255. Dicho valor indica la saturación de rojo. Mayor saturación implicará mayor cantidad de este color en la mezcla final. En caso de tomar valor cero, el rojo no intervendrá en el color definitivo.

El parámetro verde puede tomar el valor entre 0 y 255. Dicho valor indica la saturación de verde. Mayor saturación implicará mayor cantidad de este color en la mezcla final. En caso de tomar valor cero, el verde no intervendrá en el color definitivo.

El parámetro azul puede tomar el valor entre 0 y 255. Dicho valor indica la saturación de azul. Mayor saturación implicará mayor cantidad de este color en la mezcla final. En caso de tomar valor cero, el azul no intervendrá en el color definitivo.

Por ejemplo, para pintar el fondo de color naranja habría que ejecutar *Color_pantalla (255, 128, 0)*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Escribe_pantalla	uint8_t [] uint8_t uint8_t uint8_t	void	Condición de START Escribir en el esclavo 000011xx Transmitir el comando 0xA6 Transmitir el dato fila Transmitir el dato columna Transmitir el dato tipo Transmitir 1 a 1 los uint8_t de info acabando con un fin de cadena '\0' Condición de STOP

La tarea de esta función es mostrar en un lugar concreto de la pantalla una información determinada que puede ser un texto o un bloque de 8x8 de una imagen.

El parámetro info es un vector de variables donde cada una de las variables es de ocho bits sin signo. Cada una de ellas representa bien un carácter en código ASCII o bien una secuencia de ocho píxeles susceptible de ser representada en la pantalla como una columna de una página. En ese último caso, cada variable puede tomar el valor entre 0 y 255 y el vector debe contener necesariamente ocho variables de modo que se haga referencia a un bloque de 8x8 píxeles. El hecho de que el valor almacenado tome uno u otro de los sentidos anteriormente citados dependerá de la variable *tipo*.

El parámetro fila puede tomar el valor entre 0 y 7. Dicho valor indica la fila de la pantalla en el que desea comenzarse a escribir el texto almacenado en la variable *info* o bien el bloque de 8x8 píxeles codificados en la misma dependiendo del valor de la variable *tipo*. La fila cero indica la fila de la parte superior de la pantalla mientras que la fila 7 indica la fila de la parte inferior de la misma.

El parámetro columna puede tomar el valor entre 0 y 20 en el caso de que desee representarse texto o entre 0 y 7 en el caso de que deseen representarse bloques de 8x8 píxeles. El valor del parámetro indica la columna de la pantalla en el que desea comenzar a representarse el valor de *info*. La columna cero indica la columna de la parte izquierda de la pantalla y el valor crece a medida que se tiende a la derecha.

El parámetro tipo indica el sentido que tienen los datos almacenados en la variable *info* y su valor puede ser ó 4 ó 7. Si el contenido fueran caracteres ASCII el valor de la variable tipo debería ser **4**. Mientras que si la información almacenada en la variable *info* fuera un bloque de 8x8 píxeles entonces la variable tipo debería tener el valor **7**.

Por ejemplo, para escribir en la segunda columna (columna 1) de la primera fila (fila 0) la palabra "UAM" habría que ejecutar *Escribe_pantalla ("UAM", 0, 1, 4)*

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Encender_byte	uint8_t uint8_t uint8_t	void	Condición de START Escribir en el esclavo 000011xx Transmitir el comando 0xA8 Transmitir el dato pagina Transmitir el dato columna Transmitir el dato byte Transmite fin de cadena '\0' Condición de STOP

La tarea de esta función es la de mostrar en una columna de una página determinada un byte de información. Es decir, equivale a la función *Escribir_pantalla()*, pero es capaz de modificar una columna solamente, lo cual la hace más flexible y posibilita la modificación de una parte pequeña de una imagen y/o texto.

El parámetro página indica la página cuya columna se desea modificar, el parámetro columna indica la columna cuyo contenido se va a modificar y el parámetro byte indica el valor al que se establecerá la columna de la página escogida. Así, el valor 0xA3 encenderán los píxeles 1º, 2º, 6º y 8º empezando por la parte superior de la página.

Nombre de la función	Parámetros	Retorno	Protocolo por parte del maestro
Pinta_logo	uint8_t uint8_t uint8_t	void	El maestro no puede hacer una llamada a esta función

Se trata de una función interna del modelo B del módulo de comunicaciones, esto hace que sea inaccesible desde el exterior. La función se encarga de mostrar en un lugar determinado de la pantalla un logotipo de los dos predefinidos almacenados en memoria correspondientes a USB e I^2C, ambos detallados en el apartado 5.2.3.

El parámetro logo puede tomar el valor 1 ó el valor 10. Si se desea mostrar el logotipo de USB la variable deberá tomar valor 1, mientras que para mostrar el logotipo de I^2C el valor de logo deberá ser 10.

Los parámetros offset_fila y offset_columna indican respectivamente la fila y columna en la que desea colocarse el logotipo con respecto a la esquina superior izquierda. Ha de tenerse en cuenta que el logotipo ocupa más de una página, de modo que no debe colocarse en la última fila ni columna posible.

Una vez detalladas cada una de las funciones, se procede a mostrar en la Figura 5-25 el diagrama de flujo de la función principal del modelo B del módulo esclavo de comunicaciones. En la Figura 5-40 puede verse el modo en que se reciben datos por las comunicaciones I^2C y se realizan las diversas llamadas a las funciones correspondientes. En la Figura 5-41, sin embargo, se muestra el modo en el que se realizan las llamadas a las funciones correspondientes pero esta vez recibiendo las órdenes por el puerto serie o USB.

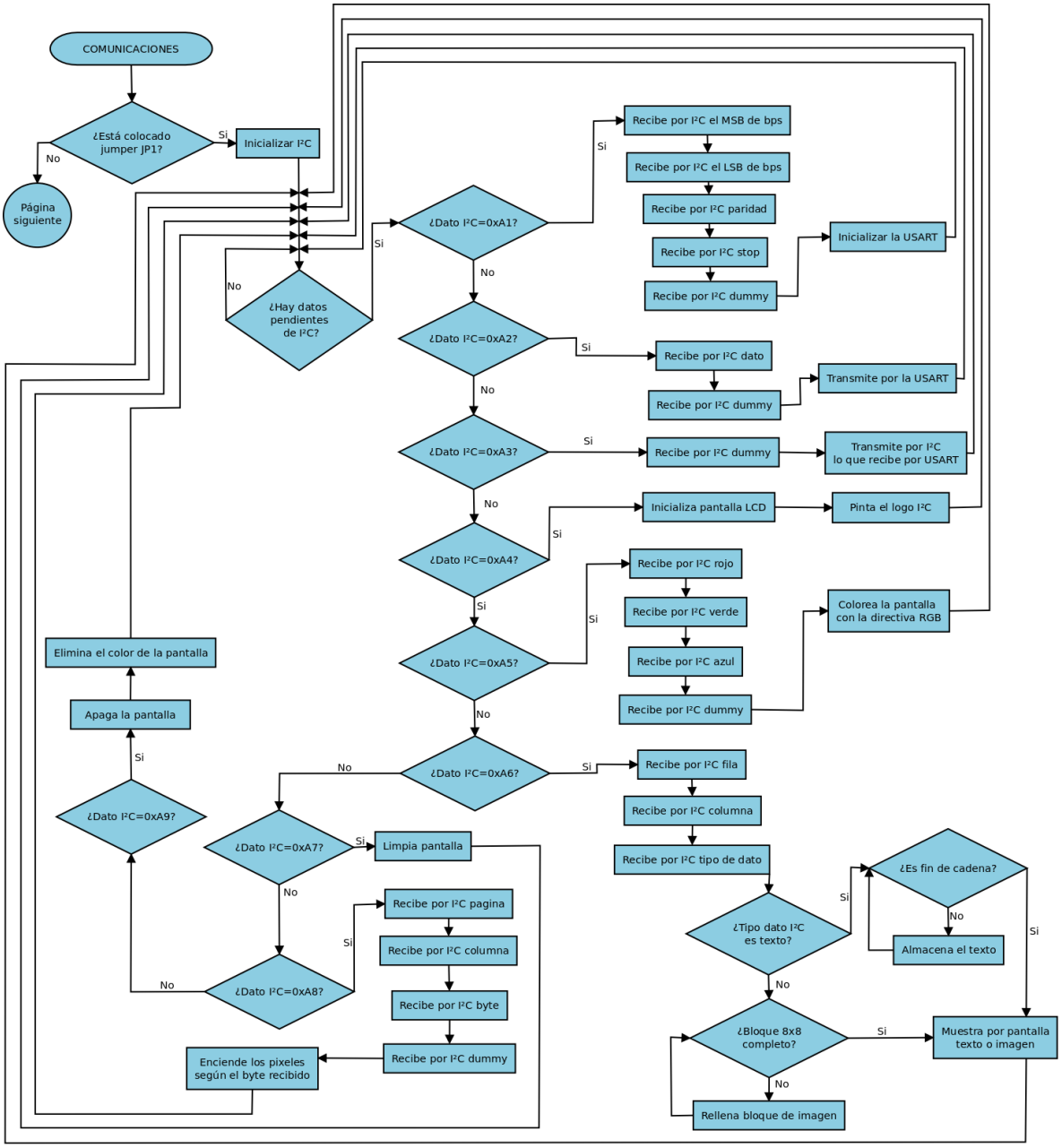


Figura 5-40: Diagrama de flujo del modelo B, módulo de comunicaciones (1/2)

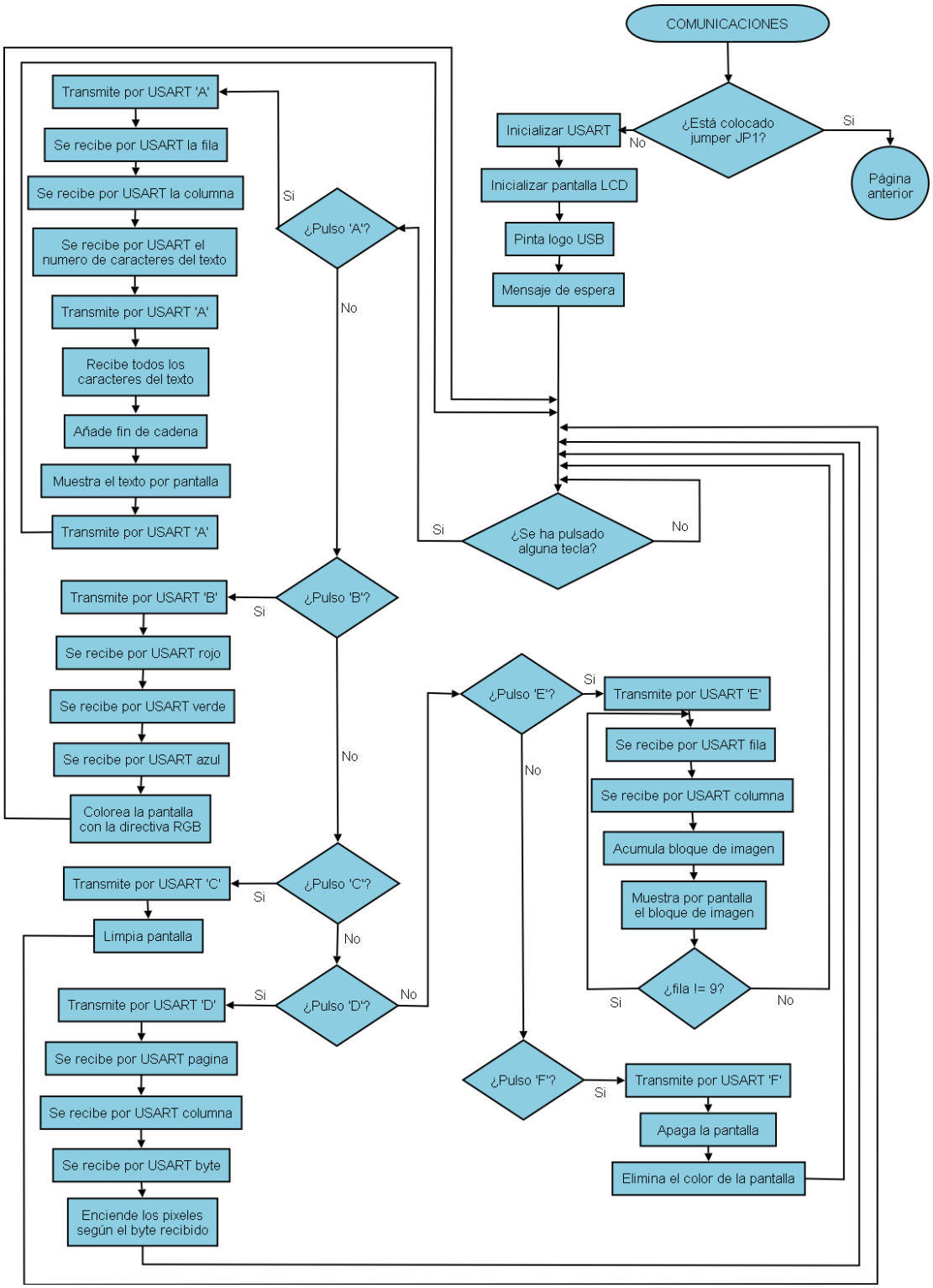


Figura 5-41: Diagrama de flujo del modelo B, módulo de comunicaciones (2/2)

5.5.4. Construcción e información técnica

5.5.4.1. Fase de prototipado del módulo

Antes de construir de manera definitiva el modelo B del módulo de comunicaciones, se hizo un prototipo de éste (Figura 5-42 izquierda), con el objetivo de encontrar posibles errores y corregir problemas de diseño. El prototipo es en realidad una ampliación del modelo A.

Una vez construido el modelo A de manera definitiva, se adaptó su prototipo para diseñar el modelo B; es por ello que físicamente ambos se parecen tanto. Sin embargo en este caso se colocó un microcontrolador diferente además de añadir el hardware necesario para el control de la pantalla. Con el objetivo de usar la pantalla tanto en el prototipo como en el módulo definitivo, se colocó una tira de pines y se soldaron algunos cables para el control del color de fondo. Una vez más se recuerda que para construir los prototipos se usó la técnica de wrapping (Figura 5-42 derecha).

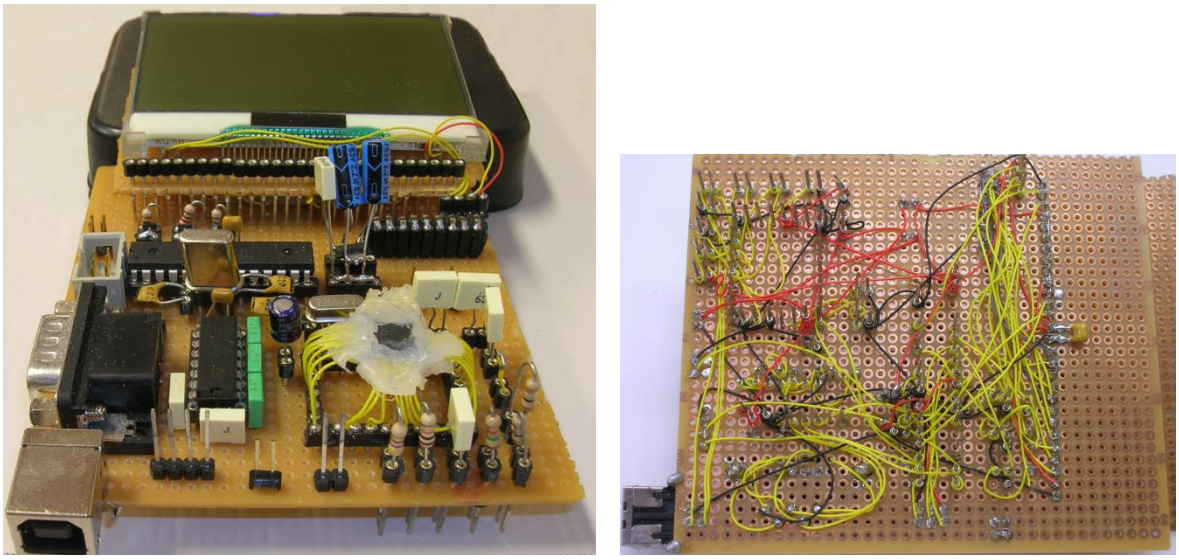


Figura 5-42: Prototipo del modelo B del módulo de comunicaciones (izda.) y wrapping asociado (dcha.)

5.5.4.2. Especificaciones, aspecto y presupuesto del módulo

En este apartado se muestra en primer lugar información acerca de las especificaciones técnicas del módulo, necesarias para saber por ejemplo la resolución de la pantalla o las dimensiones del módulo; dicha información se aprecia en la Tabla 5-10.

Especificaciones técnicas del modelo B del módulo de comunciaciones	
Tensión necesaria	5V.
Corriente máxima	85mA.
Puertos de comunicaciones	RS-232 hemnbra, USB tipo B hembra e I^2C
Tipo y resolución de la pantalla	LCD de texto y gráfica de 128x64 puntos
Color de la pantalla	Color de fondo seleccionable con la directiva RGB
Dimensiones	55,2450mm. de alto y 102,5525mm. de ancho (56,66cm^2.)

Tabla 5-10: Especificaciones técnicas del módulo de comunicaciones B

En segundo lugar se muestra, en la Figura 5-43, la parte superior del modelo B del módulo de comunicaciones una vez construido en su versión definitiva. La parte inferior, en la que se haya la pantalla, se mostró anteriormente para ilustrar algunas situaciones.

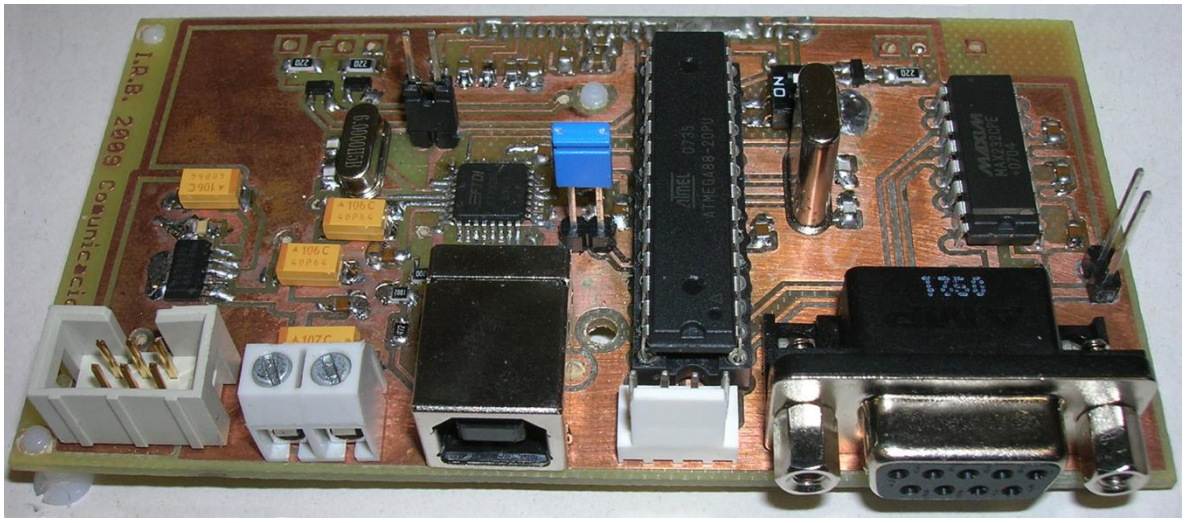


Figura 5-43: Imagen real del modelo B del módulo de comunicaciones

En tercer lugar se hace constar el precio que costaría construir el módulo de nuevo. En la Tabla 5-11, se muestra el precio de cada componente necesario para realizar el montaje del modelo B del módulo de comunicaciones. Los precios han sido extraídos de Merchan electrónica y componentes(61), exceptuando algunos de ellos, que por no encontrarse en el distribuidor anterior se han consultado en Farnell(62), estos últimos están en color azul.

[61] Los precios están sujetos a modificaciones y deben comprobarse en http://www.e-merchan.com/

[62] Los precios están sujetos a modificaciones y deben comprobarse en http://es.farnell.com/

Los que no se han encontrado en ninguno de los dos anteriores han sido consultados en RS[(63)], en este caso, han sido coloreados en naranja.

Referencia	Encapsulado	Componente	Valor	Precio
C1…8, C10, C13, C14, C31	0805	Condensador	100nF	12x0.0116€
C9, C11	0805	Condensador	22pF	2x0.022€
C12, C20…27	0805	Condensador	1µF	9x0.013€
C18, C19	0805	Condensador	27pF	2x0.016€
C29, C30	6032	Condensador	2,2µF	2x0.2€
C15	0805	Condensador	33nF	1x0.024€
C16	6032	Condensador	10µF	1x0.33€
C17	7343	Condensador	100µF	1x1.3€
C28	0805	Condensador	10nF	1x0.016€
R2	0805	Resistencia	470Ω	1x0.0038€
R3	0805	Resistencia	10kΩ	1x0.008€
R6	0805	Resistencia	4k7Ω	1x0.006€
R10	0805	Resistencia	1k5Ω	1x0.0028€
R1, R4, R5	1206	Resistencia	22Ω	3x0.0028€
R8, R9	0805	Resistencia	27Ω	2x0.006€
Q1…3	SOT23	Transistor	BSS138N	5x0.09€
ATmega48	DIL28	Microcontrolador	ATmega48	1x2.84€
MAX232	DIL16	Driver	MAX232	1x1.1€
FT232BL	QFP32	Driver	FT232BL	1x7.2€
JP1…3		Jumper	Tira min. 6 pines	1x0.51€
Q4	HC49/S	Cristal oscilador	7,3728MHz	1x1.06€
Q5	HC49/S	Cristal oscilador	6MHz	1x1.10€
LCD_64128M		Pantalla LCD	LCD_64128M	1x8.54€
TPS78630DCQR	SOT223-6	Regulador de 3V	TPS78630DCQR	1x1.94€
DB9	DB9	Conector	Serie Hembra	1x1.05€
USB		Conector	USB tipo B Hembra	1x0.48€
CONECTOR SPI		Conector	SPI	1x1.02€
J1	Molex 2,54mm	Conector	I^2C	1x0.479€
Esclavo	DIP (DIL)	Switch SMD	Selector de esclavo	1x1.28€
X2	Pasante 2 vías	Clema 5mm	Bornes para PCB	1x0.27€
Total				**31,7622€**

Tabla 5-11: Precio de los componentes del módulo de comunicaciones B

Teniendo en cuenta que construir cada dm^2 de PCB cuesta 60€. Se concluye que al coste de los componentes hay que sumar 60€ por construcción.

[63] Los precios están sujetos a modificaciones y deben comprobarse en http://es.rs-online.com/

6 Módulo maestro

6.1 Motivación del módulo

Para comprender la motivación del módulo maestro es necesario recordar las primeras páginas de este PFC, en el que se mostraba que el objetivo del mismo no era otro que diseñar una serie de módulos para una plataforma robótica genérica. Tras una breve reflexión acerca de las necesidades que puede tener cualquier robot se concluyó la necesidad de tres módulos básicos, los tres anteriormente detallados, locomoción, US&IR y comunicaciones. Finalmente se concluía la necesidad de un cuarto módulo, el módulo maestro, que fuera capaz de resolver un problema haciendo uso de los módulos esclavo.

Así pues, aunque el objetivo del PFC es la creación de los módulos esclavo, es necesario diseñar, construir e implementar un módulo maestro que se comunique con los módulos esclavo para así poder validar el funcionamiento de éstos.

El módulo maestro no es más que un "gestor" cuya función principal es la de hacer llamadas a los módulos esclavo para que realicen las tareas pertinentes. En este sentido, se puede hacer una analogía con el ser humano.

Las manos, los pies, los oídos, los ojos, etc., serían los módulos esclavo; cada uno realiza únicamente las tareas en las que está especializado mientras que el cerebro sería el módulo maestro que envía una serie de órdenes para resolver un problema concreto, por ejemplo, cruzar la calle.

Mientras que en el ser humano el cerebro se comunica con el resto del cuerpo mediante el sistema nervioso, el módulo maestro se comunica con el resto de módulos mediante el bus de I^2C.

Una vez justificada la necesidad de un módulo maestro que integre y gestione el sistema sería lógico comenzar un análisis acerca de las condiciones que debe reunir el núcleo central del módulo que, al igual que en los módulos esclavo, es un microcontrolador. No obstante, en el caso del módulo maestro no existen una serie de características mínimas bien definidas. Éstas vendrán determinadas por la aplicación a la que se desee destinar el sistema completo, es decir;

- Si se deseara hacer un mapeo tridimensional de una habitación (para lo cual se necesitaría el módulo de locomoción y el módulo US&IR), sería deseable que el microcontrolador del módulo maestro tuviera una memoria flash grande (para almacenar y procesar los datos) y una frecuencia de trabajo elevada (para realizar la tarea en tiempo real).
- Si se quisiera realizar un sistema para medir la altura de un objeto y mostrarla por una pantalla textualmente y según un código de colores (rojo = bajo, verde = alto y naranja = mediano) sería necesario el uso del módulo US&IR y el modelo B del módulo de comunicaciones y en ese caso no sería imprescindible que el microcontrolador del módulo maestro tuviera una memoria flash grande ni una frecuencia elevada.

Se concluye este apartado dejando patente que las características que debe cumplir el microcontrolador en el caso del módulo maestro dependerán de la aplicación que se desee realizar con los módulos esclavo. No obstante, existe un requisito fundamental que debe cumplir el microcontrolador y es que sea capaz de enviar órdenes según el protocolo I^2C ya que todos los módulos esclavo han sido diseñados para comprender dicho protocolo.

En el apartado siguiente se muestra la API implementada en el maestro para que cualquier usuario sea capaz de manejar las funcionalidades de los módulos esclavo abstrayéndose de los detalles y enfocando todo su esfuerzo en la resolución de una tarea a alto nivel.

6.2 API para el control de todos los módulos por I^2C

En este apartado se hace un recorrido por las funciones implementadas en el maestro para obtener las funcionalidades propias de cada uno de los módulos esclavo. En este sentido, las funciones implementadas no son más que llamadas a las funciones de los esclavos, siguiendo el protocolo I^2C y el orden definido para cada una de ellas.

Así pues, se ha creído conveniente el hecho de no dar más que una muy breve información acerca de la utilidad de cada una de las funciones; para ampliar dicha información y usar de manera adecuada cada una de las funciones se recomienda acudir al apartado de diseño software del capítulo correspondiente al módulo esclavo al que pretende hacerse referencia. También puede acudirse al cd que se acompaña en el que aparece, además del código fuente del maestro, una cabecera con información para el uso de los parámetros de cada una de las funciones.

A continuación aparece un recuadro de color verde para cada una de las funciones que se han implementado en el maestro (las cuales aparecen en color rojo). La función a la que hacen referencia aparece en color negro en el interior del recuadro.

Para el uso del ***módulo US&IR*** se han implementado las siguientes funciones gracias a las cuales se pueden obtener todas las características de tal módulo.

void leer_ultrasonidos (uint8_t activar, uint16_t sensores[], uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo de US&IR concreto, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void lee_US (uint8_t activar, uint16_t sensores[]);

que está implementada en todos los módulos US&IR con el objetivo conocer la distancia que hay entre uno o varios sensores de ultrasonidos y el obstáculo/os más cercanos a él/ellos.

void leer_infrarrojos (uint8_t activar, uint16_t sensores[], uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo de US&IR concreto, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void lee_IR (uint8_t activar, uint16_t sensores[]);

que está implementada en todos los módulos US&IR con el objetivo conocer la distancia que hay entre uno o varios sensores de infrarrojos y el obstáculo/os más cercanos a él/ellos.

Para el uso del ***módulo de locomoción*** se han implementado las siguientes funciones gracias a las cuales se pueden obtener todas las características de tal módulo.

void avanza_distancia (uint8_t movimiento, uint8_t porcentaje, uint8_t radio, uint8_t motor, uint16_t distancia, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo de locomoción concreto, definido mediante el parámetro *num_esclavo*, que ejecute la orden **void avanza_distancia (uint8_t movimiento, uint8_t porcentaje, uint8_t radio, uint8_t motor, uint16_t distancia);** que está implementada en todos los módulos de locomoción con el objetivo de avanzar una distancia concreta con uno o ambos motores, hacia delante o hacia detrás y con una velocidad determinada.

void girar_angulo (uint8_t porcentaje, uint8_t angulo, uint8_t sentido, uint8_t radio, uint8_t ancho, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo de locomoción concreto, definido mediante el parámetro *num_esclavo*, que ejecute la orden **void girar_angulo (uint8_t porcentaje, uint8_t angulo, uint8_t sentido, uint8_t radio, uint8_t ancho);** que está implementada en todos los módulos de locomoción con el objetivo de girar un ángulo concreto definido en grados hacia la derecha o la izquierda a una velocidad determinada.

uint16_t calcula_distancia (uint8_t motor, uint8_t radio, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo de locomoción concreto, definido mediante el parámetro *num_esclavo*, que ejecute la orden que está implementada

uint16_t calcula_distancia (uint8_t motor, uint8_t radio);

en todos los módulos de locomoción con el objetivo de calcular la distancia recorrida por uno o ambos motores desde la última vez que se ejecutó una llamada al módulo.

void mover_motor (uint8_t movimiento, uint8_t porcentaje, uint8_t motor, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo de locomoción concreto, definido mediante el parámetro *num_esclavo*, que ejecute la orden que está implementada

void mover_motor (uint8_t movimiento, uint8_t porcentaje, uint8_t motor);

en todos los módulos de locomoción con el objetivo de mover de manera indefinida uno o ambos motores hacia delante o hacia detrás a una velocidad concreta.

uint8_t en_movimiento (uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo de locomoción concreto, definido mediante el parámetro *num_esclavo*, que ejecute la orden que está implementada

uint8_t en_movimiento ();

en todos los módulos de locomoción con el objetivo de mover de manera indefinida uno o ambos motores hacia delante o hacia detrás a una velocidad concreta.

Para el uso del ***modelo A del módulo de comunicaciones*** se han implementado las siguientes funciones gracias a las cuales se obtienen las características de tal módulo.

void inicializa_USART_A (uint16_t bps, uint8_t paridad, uint8_t stop, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo A de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void inicializa_USART(uint16_t bps, uint8_t paridad, uint8_t stop);

que esta implementada en todos los modelos A de los módulos de comunicaciones con el objetivo de inicializar la USART a una velocidad en baudios concreta empleando un método de detección de errores en base a paridad par, impar o sin paridad y con uno o dos bits de parada.

void transmite_USART_A (uint8_t datos, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo A de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void transmite_USART(uint8_t datos);

que esta implementada en todos los modelos A de los módulos de comunicaciones con el objetivo de transmitir un byte por la USART o en su defecto mantenerse a la espera del envío del mismo ya que se trata de una función bloqueante.

uint8_t recibe_USART_A (uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo A de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

uint8_t recibe_USART();

que esta implementada en todos los modelos A de los módulos de comunicaciones con el objetivo de recibir un byte de la USART o en su defecto mantenerse a la espera de la recpción ya que se trata de una función bloqueante.

Y por último, para el uso del ***modelo B del módulo de comunicaciones*** se han implementado las siguientes funciones gracias a las cuales se pueden obtener todas las características de tal módulo.

void inicializa_USART_B (uint16_t bps, uint8_t paridad, uint8_t stop, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void inicializa_USART(uint16_t bps, uint8_t paridad, uint8_t stop);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de inicializar la USART a una velocidad en baudios concreta empleando un método de detección de errores en base a paridad par, impar o sin paridad y con uno o dos bits de parada.

void transmite_USART_B (uint8_t datos, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void transmite_USART(uint8_t datos);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de transmitir un byte por la USART o en su defecto mantenerse a la espera del envío del mismo ya que se trata de una función bloqueante.

uint8_t recibe_USART_B (uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

uint8_t recibe_USART();

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de recibir un byte de la USART o en su defecto mantenerse a la espera de la recpción ya que se trata de una función bloqueante.

void Inicializa_pantalla (uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void Inicializa_LCD (); Pinta_logo(10, 6, 14);

que estan implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de preparar la pantalla LCD para la futura recepción de comandos y datos y mostrar el logotipo de I^2C en el extremo inferior derecho de la pantalla.

void Color_pantalla (uint8_t rojo, uint8_t verde, uint8_t azul, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void color_pantalla(uint8_t rojo, uint8_t verde, uint8_t azul);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de mostrar el fondo de pantalla de un color determinado según la descripción RGB.

void Encender_byte (uint8_t pagina, uint8_t columna, uint8_t byte, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void Encender_byte(uint8_t pagina, uint8_t columna, uint8_t byte);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de dar valor a los 8 píxeles de una columna y página determinadas de acuerdo a la información del parámetro byte.

void Limpia_pantalla (uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

LCD_SetScreen (0x00);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de poner a cero (borrar) todos los píxeles de la pantalla.

void Apaga_pantalla (uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

Escribir_comando(0xAE);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de apagar la pantalla, de manera que no responda a ningún comando ni dato.

void Escribe_texto (char texto[], uint8_t fila, uint8_t columna, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void Escribe_pantalla(uint8_t info[], uint8_t fila, uint8_t columna, 0x04);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de escribir en un lugar determinado de la pantalla un texto ó carácter.

void Pinta_pagina (uint8_t imagen[], uint8_t fila, uint8_t columna, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void Escribe_pantalla(uint8_t info[], uint8_t fila, uint8_t columna, 0x07);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de mostrar en un lugar determinado de la pantalla un bloque de 8x8 píxeles pasado como parámetro (8 posiciones del vector info[] cada uno de tipo uint8_t).

void Pinta_imagen (uint8_t offset_fila, uint8_t offset_columna, uint8_t num_esclavo);

Con una llamada a esta función el maestro solicita a un esclavo concreto del modelo B de comunicaciones, definido mediante el parámetro *num_esclavo*, que ejecute la orden

void Escribe_pantalla(uint8_t info[], uint8_t fila, uint8_t columna, 0x07);

que esta implementada en todos los modelos B de los módulos de comunicaciones con el objetivo de mostrar en un lugar determinado de la pantalla una imagen de cómo máximo 128x64 píxeles monócroma almacenada en un fichero al que se debe llamar Dibujo.bmp (o modificar el nombre en el Makefile) y que posteriormente se codificada en el fichero *imagen.h* mediante el fichero *decodificador.c* implementado especialmente para tal fin de este PFC.

6.3 Aplicación de ejemplo implementada

6.2.1. Descripción

Con el objetivo de validar el funcionamiento de todos los módulos esclavos se ha decidido implementar una aplicación sencilla que hiciera uso de los módulos de locomoción, US&IR y comunicaciones (modelo B).

En líneas generales el objetivo de la aplicación es que una plataforma robótica recorra una hilera de obstáculos colocados a su derecha y una vez que encuentre un hueco del tamaño adecuado realice las maniobras pertinentes para aparcar en él (Figura 6-1).

Figura 6-1: Hilera de dos obstáculos y robot en la posición inicial

Con el objetivo de dotar a la plataforma de un efecto visual más atrayente e incrementar su valor añadido se han implementado una serie de actuaciones que completan la tarea. A continuación se describe con mayor detalle la aplicación final.

En primer lugar la plataforma robótica realiza un breve juego de luces con cuatro LEDs bicolor para indicar el encendido del robot. Posteriormente la pantalla LCD del módulo de comunicaciones se enciende en color rojo y muestra el logotipo de I^2C para informar de que las comunicaciones se han inicializado con éxito. Comienza entonces el avance de la plataforma robótica y en tiempo real se muestra por pantalla de manera gráfica los huecos que se van detectando en la hilera de obstáculos (Figura **6-2**).

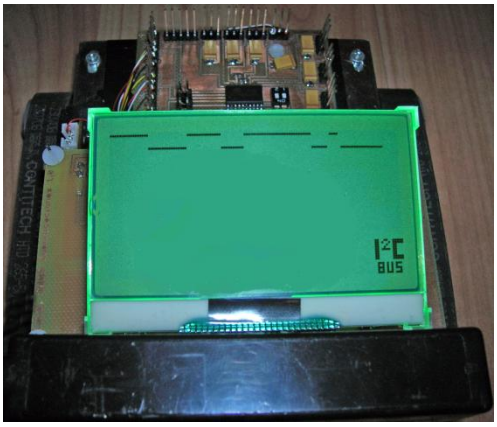


Figura 6-2: Detección en tiempo real

Cuando el robot detecta que comienza un hueco la pantalla se vuelve de color verde y una vez que el hueco acaba vuelve a colorearse en rojo mostrando además información acerca de la anchura y profundidad del hueco. Si el hueco no satisface las medidas necesarias para aparcar, el robot sigue avanzando en busca de otros lugares para aparcar sin dejar de mostrar información por la pantalla sobre la forma de la hilera de obstáculos. En cuanto el robot detecte un hueco adecuado, tras mostrar de nuevo la anchura y profundidad del mismo se coloreará la pantalla en color azul y se aparecerá en la misma el mensaje "Se ha detectado hueco". A continuación aparecerá el mensaje "Iniciando aproximación a 1cm." y la plataforma robótica se aproximará a la hilera de obstáculos situándose a un centímetro de la misma tras lo cual se mostrará por la pantalla el mensaje de "Aproximación correcta". Una vez realizada la aproximación aparece por pantalla el mensaje "Iniciando maniobras" y es entonces cuando el robot inicia las maniobras necesarias para aparcar en el hueco detectado. Para aparcar se ha tratado de simular los movimientos realistas que realizaría un vehículo real. Una vez aparcado el vehículo se realizan una serie de movimientos en el caso de ser necesarios para situarse en el punto medio del hueco. Para finalizar se muestra el mensaje "…Se ha aparcado…" y posteriormente aparece una imagen de un vehículo deportivo mientras se colorea la pantalla con seis colores diferentes de manera consecutiva. Finalmente la pantalla LCD se apaga.

En el caso de que la plataforma robótica no consiguiera hallar ningún hueco avanzaría hasta situarse a menos de 4cm. de un objeto que tuviera frente a sí, típicamente una pared y mostraría por pantalla uno de los siguientes mensajes informativos:

- "No hay huecos suficientemente anchos"; en el caso de que el último hueco detectado fuera demasiado estrecho como para aparcar en él.
- "No hay huecos suficientemente profundos"; en el caso de que el último hueco detectado no fuera lo suficientemente profundo como para garantizar maniobras de aparcamiento correctas.
- "No hay huecos suficientemente anchos ni suficientemente profundos"; en el caso de que el último hueco detectado fuera demasiado pequeño tanto en anchura como en profundidad como para aparcar en él.

6.2.2. Diseño hardware

El módulo maestro diseñado en este caso es muy simple y está directamente orientado a cumplir los requisitos del problema particular que se pretende resolver; es por ello que no se ha construido en versión definitiva sino únicamente en su fase de prototipado. No existe por tanto un layout del mismo aunque puede consultarse el esquemático en el anexo G.5.

El módulo maestro está compuesto básicamente por cuatro LEDs bicolor, un microcontrolador, dos resistencias de pull-up de 4k7Ω. para el bus de comunicaciones I^2C, un botón de reset, el conector de I^2C, el conector de alimentación y el conector de SPI. Puede cambiarse el firmware del maestro para destinar el sistema a cualquier otra tarea siempre que el código generado sea inferior a 8kbytes.

En cuanto al hardware del sistema en su conjunto, para poder probar la aplicación descrita anteriormente es necesario en primer lugar disponer de una plataforma física sobre la que situar todos los elementos y posteriormente interconectar los módulos entre sí y con algunos sensores. El objetivo de este apartado es dar a conocer esos aspectos concretos de la aplicación de prueba diseñada.

La estructura física empleada ha sido diseñada y cedida para su uso en este PFC por el profesor D. Guillermo González de Rivera Peces y a continuación se adjuntan algunas fotos de la misma en la Figura 6-3.

Figura 6-3: Vista superior (izquierda) y posterior (derecha) de la estructura física empleada

Sobre la estructura anteriormente mostrada se han situado los módulos de locomoción, US&IR, el modelo B de comunicaciones y el maestro.

El módulo de locomoción se ha colocado de manera estratégica en la parte inferior de la estructura junto con dos motores de corriente continua Faulhaber de los que se habló en el apartado 3.2.1; de esta manera queda oculto mientras el robot avanza y se puede aprovechar en mayor medida el espacio que ofrece la estructura física.

El módulo US&IR se ha colocado en la parte trasera superior de la plataforma y conectado a él se han situado tres sensores de ultrasonidos (apartado 4.2.1) y un sensor de infrarrojos (apartado 4.2.2). Uno de los sensores de ultrasonidos es el modelo SRF04 y los otros dos son SRF05 (de esta manera queda demostrada la compatibilidad de ambos). El sensor de infrarrojos GP2D12 se ha situado en el lateral derecho de la estructura. En cuanto a los sensores de ultrasonidos, uno de ellos se ha situado en el frontal de la estructura, por la parte inferior de ésta, de manera que queda oculto y los otros dos, también ocultos por la parte inferior se han colocado de manera que apuntan a la parte trasera derecha y a la parte trasera izquierda respectivamente. Se ha considerado que esta colocación era la óptima para tener control en cualquier momento acerca de la distancia a los obstáculos.

En la parte delantera superior se ha colocado el módulo maestro junto con la batería que proporciona dos tensiones diferentes, una tensión de 5V y otra de 9V.

Junto a la batería y sobre el módulo maestro se ha colocado el modelo B del módulo de comunicaciones con su parte posterior orientada hacia arriba de manera que el usuario pueda ver el contenido que ofrece la pantalla LCD.

Finalmente se han realizado todas las conexiones necesarias:

- *Cableado de alimentación*; se conectaron todos los módulos a la tensión de alimentación de 5V. y el módulo de locomoción de conecto además a otra tensión de 9V. ambas referidas a la masa común del sistema.
- *Cableado de comunicaciones*; se interconectaron todas las señales de SDA, SCL y GND de los módulos dando lugar a un bus I^2C. Además se conectaron las señales de $\overline{RESET}$ de todos los módulos dando lugar a una línea de $\overline{RESET}$ general.
- *Cableado de sensores*; se conectaron los tres sensores de ultrasonidos SRF04/05 a los conectores de ultrasonidos del módulo US&IR. El sensor de ultrasonidos delantero se conectó a la posición del sensor 4, el sensor de

ultrasonidos que apuntaba a la posición trasera derecha se conectó a la posición del sensor 3 y el sensor de ultrasonidos que apuntaba a la posición trasera izquierda se conectó a la posición del sensor 2. En cuanto al sensor de infrarrojos, éste se conecto a la posición correspondiente al sensor 4. También se conectaron los motores a los conectores del módulo de locomoción.

Con el objetivo de tener una idea más clara del conjunto final se adjuntan una serie de imágenes del sistema definitivo desde diferentes perspectivas. (El sistema final construido tiene unas dimensiones de 14cm. de longitud, 9,8cm. de altura y 16cm. de anchura).

Figura 6-4: De arriba abajo y de izquierda a derecha se muestran perspectivas del lateral izquierdo, del lateral derecho, de la parte trasera y de la parte posterior de la plataforma construída.

6.2.3. Notas de la aplicación

1. No debe olvidarse que la aplicación únicamente trata de mostrar el correcto funcionamiento de todos los módulos siendo gestionados por un maestro.
2. La plataforma robótica solo es capaz de aparcar hacia la derecha puesto que se le ha conectado un único sensor de infrarrojos en la parte derecha. El módulo está preparado para admitir otro sensor de infrarrojos en la parte izquierda pero no se ha considerado necesario aparcar hacia ambos lados. No obstante si se desea aparcar tanto hacia la derecha como hacia la izquierda bastaría con modificar muy pocas líneas del código de la aplicación principal y añadir un sensor de infrarrojos más.
3. La plataforma necesita huecos de un tamaño mínimo de 30cm. de anchura y 20cm. de profundidad. En cualquier hueco que cumpla dichos requisitos el robot sería capaz de aparcar de manera adecuada.
4. Si durante el avance de la plataforma robótica algún objeto se cruzará por delante del mismo a menos de 4cm. la aplicación terminaría. Esto se ha hecho simulando por ejemplo el paso de un peatón demasiado cerca del frontal del vehículo. Podrían diseñarse algoritmos para esquivar al objeto en tales situaciones, pero no se han implementado.
5. Se han simulado las maniobras de un vehículo real aparcando para dotar de mayor realismo la aplicación, no obstante podría haberse usado el giro de 90º sobre sí mismo para simplificar el aparcamiento del robot.
6. La plataforma robótica debe situarse a un mínimo de 4cm. de distancia lateral con respecto a la hilera de vehículos que se analizará durante el avance.

Finalmente, la aplicación de prueba diseñada ***ha funcionado correctamente*** en diferentes suelos y con obstáculos de diferentes colores y texturas. Además, ***el desarrollo del sistema ha sido muy rápido y sencillo*** debido a la abstracción que permitía la API desarrollada para el módulo maestro.

6.2.4. Diseño software

El software de la aplicación de ejemplo planteada hace uso de la API presentada anteriormente para el uso de los módulos esclavo, pero además se han implementado una serie de funciones para hacer más legible el código. A continuación, en la página siguiente,

en la Figura 6-5, se muestra el diagrama de flujo de la aplicación principal. El diagrama está hecho a muy alto nivel para simplificar la comprensión del mismo; sin embargo, para mayor detalle se recomienda acudir al cd que se acompaña en el que se muestra el código fuente.

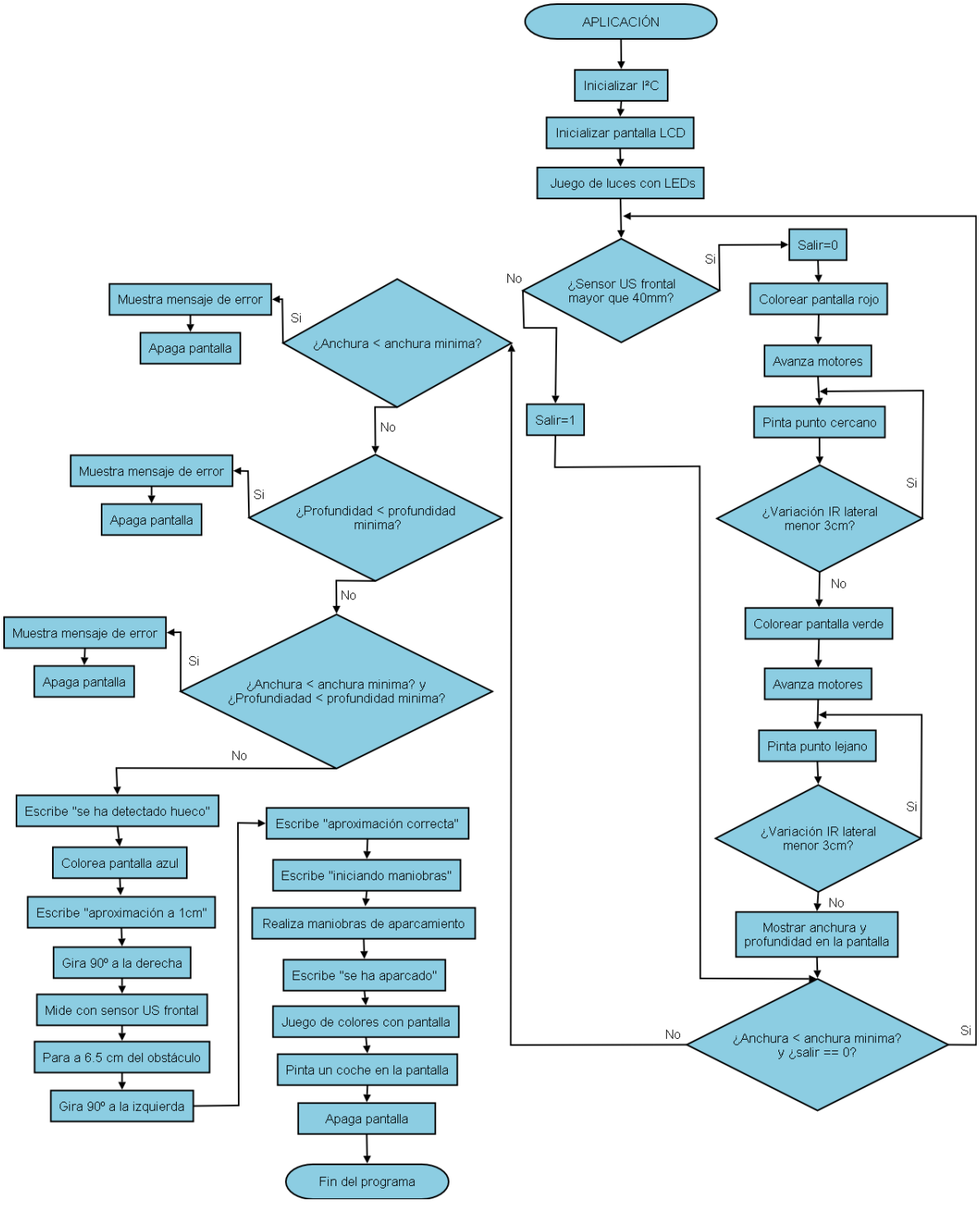


Figura 6-5: Diagrama de flujo de la aplicación implementada

7 Integración, pruebas y resultados

7.1 Prueba de sensores de ultrasonidos

- **Objetivo**: Cálculo de la distancia existente hasta un obstáculo mediante el sensor número cuatro de ultrasonidos.
- **Módulos involucrados:** Es necesario el uso de tres módulos.
  - ✓ Módulo US&IR: encargado de gobernar el sensor de ultrasonidos.
  - ✓ Módulo de comunicaciones A: encargado de enviar al ordenador la medida realizada.
  - ✓ Módulo maestro: encargado de dar las órdenes a los dos módulos citados anteriormente.
- **Conexiones necesarias**: Es necesario realizar una serie de conexiones previas.
  1. Se conectan al bus I^2C de comunicaciones los módulos US&IR, comunicaciones A y maestro. Para ello se deben interconectar las señales SDA, SCL, y GND de los tres módulos.
  2. Se conecta un sensor de ultrasonidos al conector cuatro del módulo US&IR.
  3. Se conecta un cable USB entre el módulo de comunicaciones A y el ordenador.
  4. Se conectan las señales de alimentación (+5V.) de los tres módulos a una fuente de alimentación que entregue la tensión necesaria.

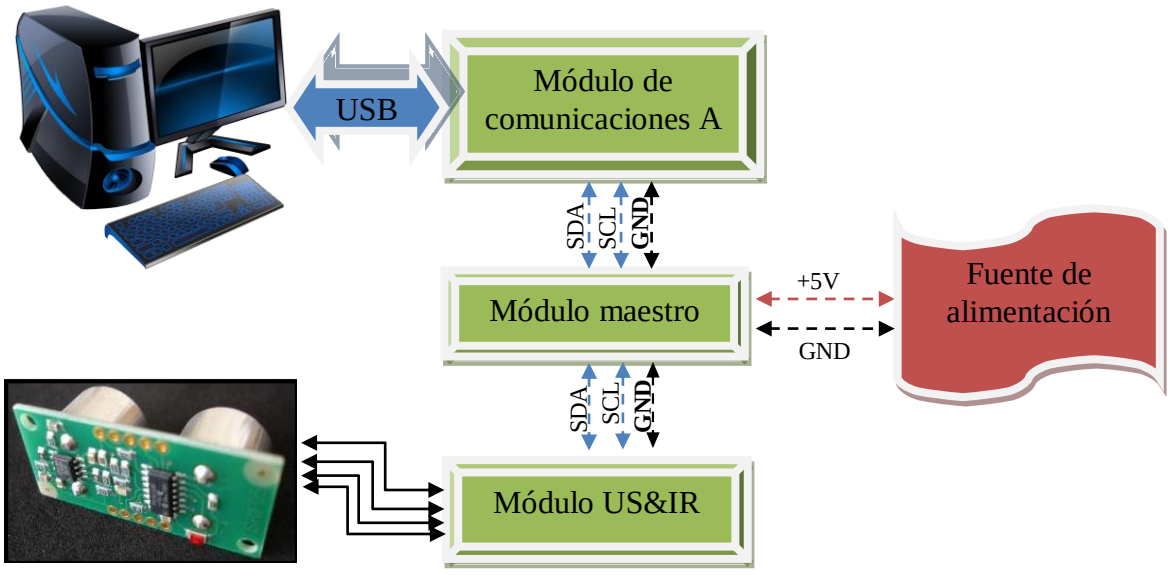


Figura 7-1: Conexiones realizadas para la prueba 1

- **Situación:** Se coloca el sensor de ultrasonidos a 8,2 cm del obstáculo. Se abre un hiperterminal en el ordenador para establecer comunicación con el sistema.
    - Cuando el usuario presiona una tecla cualquiera del ordenador, el módulo de comunicaciones A lo detecta y se lo comunica al módulo maestro.
    - El módulo maestro ordena al módulo US&IR que realice la medición de la distancia al obstáculo usando el sensor de ultrasonidos número 4.
    - El módulo US&IR realiza la medición y transmite el resultado obtenido, en milímetros, al módulo maestro.
    - El módulo maestro ordena al módulo de comunicaciones A que muestre el resultado por el hiperterminal con un cierto formato.
- **Resultados:** Los resultados obtenidos se muestran de dos maneras, por un lado se observa el hiperterminal y por otro lado se observa un analizador lógico conectado al bus I^2C y al sensor de ultrasonidos. Concretamente,
    - Se ha conectado el cable D29 al pin SCL del módulo de US&IR.
    - Se ha conectado el cable D30 al pin SDA del módulo de US&IR.
    - Se ha conectado el cable D0 al pin "Eco" del sensor de ultrasonidos SRF05.
    - Se ha conectado el cable D1 al pin "Disparo" del sensor de ultrasonidos SRF05.

Se ha tomado en esa situación la captura de pantalla que se muestra en la Figura 7-2 y que se analiza en la siguiente página.

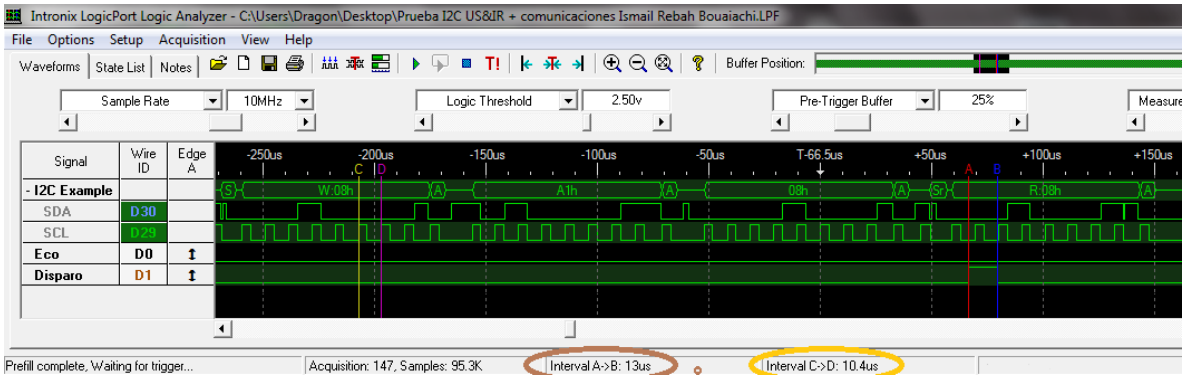


Figura 7-2: Captura de pantalla tomada número 1

Se observa en primer lugar la condición de START[64], posteriormente el maestro indica su deseo de escribir en el esclavo número 0x08 que corresponde al esclavo número cero del módulo US&IR; dicho esclavo asiente a la petición.

El maestro envía el comando 0xA1, que indica la llamada a la función leer_US de dicho esclavo. El módulo US&IR asiente de nuevo, a continuación el maestro envía de manera codificada su deseo de que la medición la realice el sensor número cuatro (0b00001000).

El esclavo asiente y se observa cómo envía un disparo de 13µs. al sensor correspondiente. La duración del disparo se ha redondeado en color marrón. Mientras tanto prosigue la comunicación I^2C y se observa cómo tras un REPEATED START el maestro indica su deseo de leer la distancia calculada.

El esclavo usa la técnica del clock stretching para que el maestro se mantenga a la espera mientras se está realizando la medida.

En la Figura 7-2 se podía observar también la frecuencia de I^2C empleada, en este caso de 100kHz. tal como se deduce del período de la señal SCL redondeada en amarillo.

Posteriormente se redujo el zoom del analizador lógico, y se tomó otra captura de pantalla, que incluía el resultado mostrado por el hiperterminal, esta situación se ilustra en la Figura 7-3 mostrada en la página siguiente.

[64] Previamente, hay más comunicación I^2C, la relativa a la inicialización de la USART y la espera de que el usuario pulse una tecla, pero se muestra, intencionadamente, sólo una parte de las comunicaciones relativa al sensor de US.

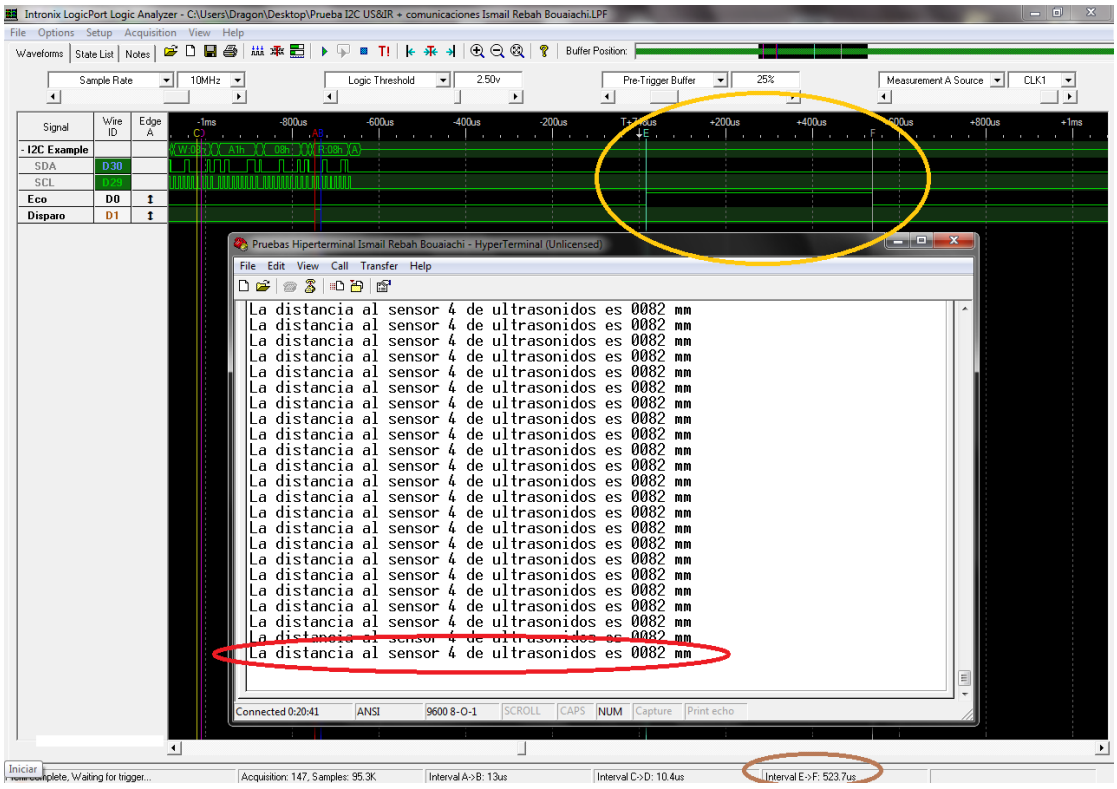


Figura 7-3: Captura de pantalla tomada número 2

En esta imagen pueden verse además de todos los detalles anteriores el eco recogido por el sensor de ultrasonidos. El eco en sí ha sido redondeado en amarillo, mientras que con un círculo de color marrón se puede observar la duración de este. Según el analizador lógico, la duración del pulso de eco es de 523,7µs. Si usamos la ecuación [4.4] vemos que dicha duración corresponde a una distancia de 90,12mm.

Sin embargo, la distancia real y la distancia que muestra el hiperterminal durante todas las ejecuciones de la prueba es de 82mm. Esto se debe a que existe un offset, entre la distancia real y la distancia estimada en base al eco, de +8mm. Dicho offset es corregido por el módulo US&IR como puede comprobarse en el código mostrado en el cd que se acompaña.

En último lugar se ha tomado otra captura de pantalla, esta vez con el zoom aún más alejado. La imagen se muestra en la Figura 7-4 y en ella se aprecia mejor la duración del eco. Se ha redondeado en color gris parte del resto de la comunicación I^2C, ya que una vez calculada la distancia es necesario que se muestre por pantalla. Y en color marrón se han redondeado dos zonas en las que se produce el anteriormente citado clock stretching, ya

que la velocidad a la que el maestro está enviando los datos al módulo de comunicaciones es más rápida que la velocidad a la que el módulo de comunicaciones los transmite al P.C. y es por ello que el módulo de comunicaciones "ralentiza" la velocidad del bus I^2C.

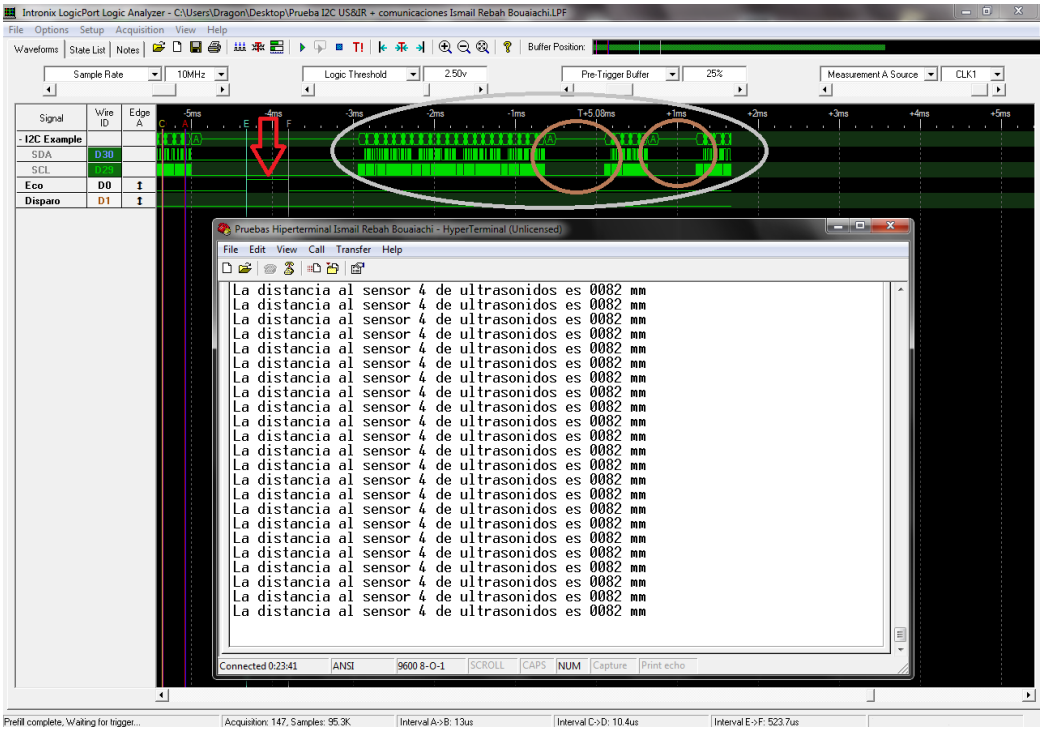


Figura 7-4: Captura de pantalla tomada número 3

- **Conclusión:** Del éxito de la prueba realizada se concluye lo siguiente:
  - ✓ El módulo de comunicaciones es capaz de comunicarse con el P.C. para mostrar información de otros módulos con el formato que decida el módulo maestro.
  - ✓ El módulo US&IR funciona con las especificaciones requeridas, la anchura del disparo es la correcta, y el registro de la duración del pulso de eco también es correcto ya que se ha estimado adecuadamente la distancia al obstáculo. También se respeta la espera entre la finalización del eco y la emisión del siguiente disparo ya que de no ser así, las medidas realizadas no serían iguales puesto que se irían acumulando ecos anteriores que falsearían el resultado. Para comprobar dicha situación se ha reducido el tiempo de espera de 50ms. a 5µs. y se ha comprobado que el sistema funcionaba erróneamente.
  - ✓ El bus de comunicaciones I^2C funciona perfectamente y no solo comprenden las comunicaciones los módulos diseñados sino que el propio analizador lógico es capaz de interpretarlo de manera correcta ya que sigue los estándares.

7.2 Prueba de comunicaciones I^2C

- **Objetivo**: Mostrar mediante el osciloscopio una captura de la comunicación I^2C establecida entre el módulo maestro y un esclavo, p.ej., el módulo de locomoción.
- **Módulos involucrados:** Es necesario el uso de dos módulos.
  - ✓ Módulo de locomoción: encargado de ejecutar la orden que recibe por parte del módulo maestro.
  - ✓ Módulo maestro: encargado de enviar alguna orden concreta al módulo de locomoción mediante el protocolo I^2C.
- **Conexiones necesarias**: Es necesario realizar una serie de conexiones previas.
  1. Se conectan al bus I^2C de comunicaciones el módulo de locomoción y el maestro. Para ello se deben interconectar las señales SDA, SCL, y GND de los dos módulos.
  2. Se conectan las señales de alimentación (+5V.) de los dos módulos a una fuente de alimentación que entregue la tensión necesaria. Se conecta la señal de alimentación (>+5V.) para entregar tensión suficiente a los motores. En este caso se han alimentado los motores con 7,5V.

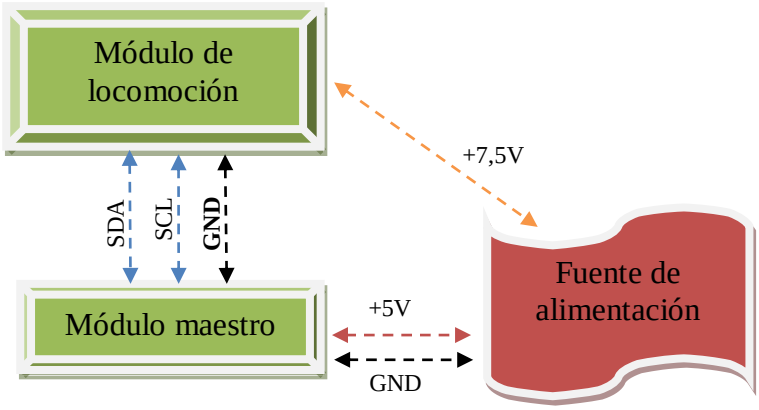


Figura 7-5: Conexiones realizadas para la prueba 1

- **Situación:** Al conectarse la alimentación, el módulo maestro establece comunicación con el esclavo número cero del módulo de locomoción mediante el protocolo I^2C con el objetivo de que el motor izquierdo avance de manera indefinida al 50% de su velocidad máxima. Para que tal evento ocurra es necesario ejecutar la instrucción ***mover_motor (AVANZAR, 50, MOTOR_IZQUIERDO, 0)*** por parte del maestro.

  La instrucción anterior se muestra decodificada en la página siguiente:

1. Envío de la condición de START por parte del maestro.
2. Envío de la dirección (sólo escritura = 0x04) del esclavo por parte del maestro.
3. Envío del asentimiento por parte del esclavo.
4. Envío del código correspondiente a la instrucción mover_motor() (0xA4) por parte del maestro.
5. Envío del asentimiento por parte del esclavo.
6. Envío del código correspondiente al parámetro AVANZAR (0x00) por parte del maestro.
7. Envío del asentimiento por parte del esclavo.
8. Envío del código correspondiente al parámetro porcentaje (50%) (0x32) por parte del maestro.
9. Envío del asentimiento por parte del esclavo.
10. Envío del código correspondiente al parámetro MOTOR_IZQUIERDO (0x01) por parte del maestro
11. Envío del asentimiento por parte del esclavo.
12. Envío de la condición de STOP por parte del maestro.

Así, el código hexadecimal resultante es:

START | (0x04<<1) | 0 | 0xA4 | 0 | 0x00 | 0 | 0x32 | 0 | 0x01 | 0 | STOP

Visto en binario resulta:

START | 00001000 | 0 | 10100100 | 0 | 00000000 | 0 | 00110010 | 0 | 00000001 | 0 | STOP

- **Resultados:** Los resultados obtenidos se muestran mediante un osciloscopio. La motivación de usar un osciloscopio es poder analizar con mayor detalle la forma de las señales que intervienen en la comunicación. Las sondas del osciloscopio se han conectado al bus I^2C mediante el conector presente en el módulo maestro. Concretamente,
  - Se ha conectado la sonda de color amarillo al pin SCL del módulo maestro.
  - Se ha conectado la sonda de color verde al pin SDA del módulo maestro.
  - Se han usado resistencias de pull-up de valor 4k7Ω.
  - Se ha usado comunicación I^2C en modo Standard.

Se ha tomado en esa situación la captura de pantalla que se muestra en la Figura 7-6 y en la que se aprecian varias cosas. En primer lugar se observan las condiciones en las que se ha tomado la captura: 2,5V. por división vertical, y en principio 100µs. por división horizontal. La comunicación dura por tanto 500µs. y se realiza a 5V. de tensión (TTL). También es posible comprobar que se está usando el modo standard de I2C ya que la frecuencia es de ~100kHz.

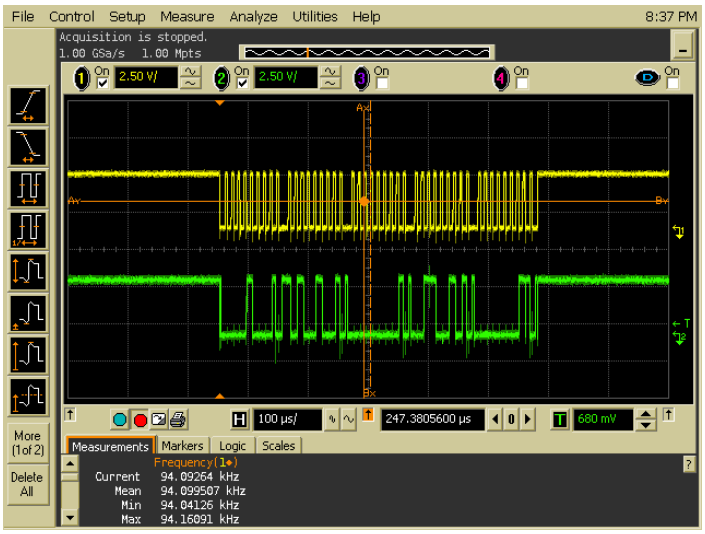


Figura 7-6: Captura de pantalla tomada número 1

A continuación, con el objetivo de comparar la secuencia obtenida con la secuencia teórica citada anteriormente se amplía el zoom hasta obtener 50µs. por división horizontal. Se muestra dicha imagen en la Figura 7-7.

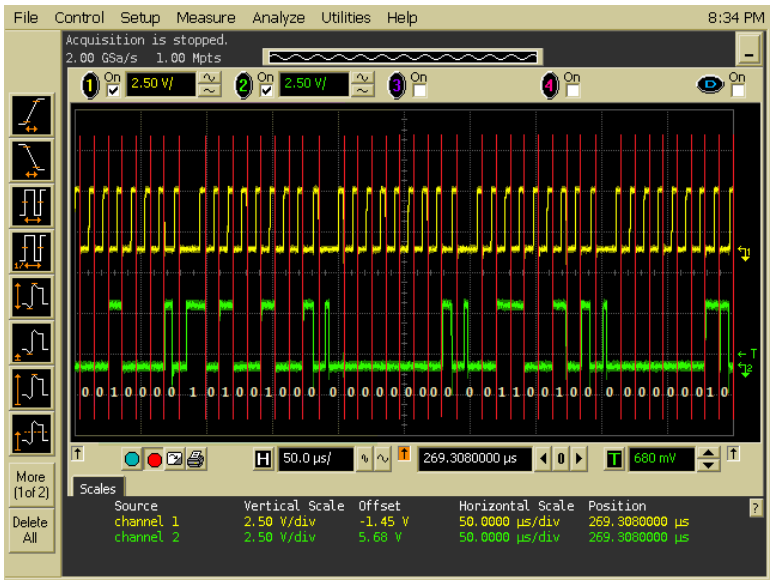


Figura 7-7: Captura de pantalla tomada número 2

Para facilitar al lector la tarea de comparación con la secuencia teórica se han pintado sobre la captura una serie de líneas rojas que dividen los pulsos de la línea SCL (recuerde que el dato de la señal SDA sólo es válido mientras la señal SCL está en estado alto lógico) y se ha añadido el valor de la señal SDA. Nótese que al ampliar la captura no ha sido posible abarcar la secuencia completa, de manera que los dos primeros bits así como las condiciones de inicio y parada no se pueden apreciar en la imagen.

En último lugar se ha tomado otra captura de pantalla, esta vez con el zoom aún más ampliado, hasta 2µs. por división horizontal. La imagen se muestra en la Figura 7-8 y en ella se aprecia la forma de diente de sierra comentada en el apartado 2.2. Se ha medido tanto el tiempo de subida como el tiempo de bajada de la señal de SCL (asumiendo que la señal SDA cumple las mismas medidas ya que ambas están sujetas a las mismas condiciones) resultando un tiempo de subida de 306,57ns. y un tiempo de bajada de 13,75ns.

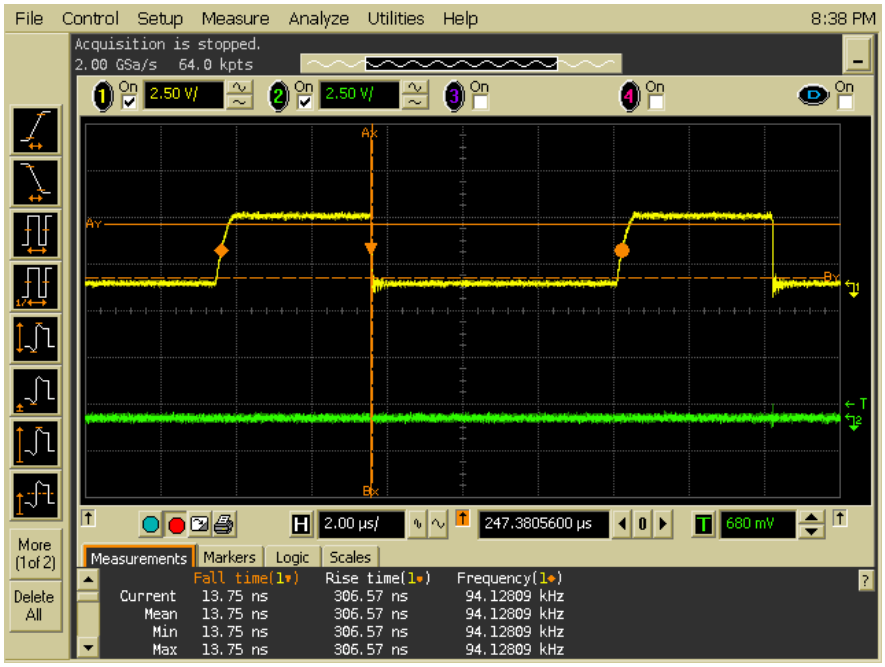


Figura 7-8: Captura de pantalla tomada número 3

De esta última imagen se deduce que los transistores MOSFET colocados a la entrada de los pines de las señales SDA y SCL del microcontrolador son muy buenos (tiempo de bajada muy rápido) y que el valor de las resistencias de pull-up escogido es bueno (tiempo de subida suficientemente rápido).

- **Conclusión:** Del éxito de la prueba realizada se concluye lo siguiente:
  - ✓ El módulo maestro es capaz de mantener comunicaciones con los módulos esclavo mediante el uso del protocolo I^2C.
  - ✓ El protocolo de comunicaciones usado cumple los requisitos de frecuencia standard (~100kHz.) y de validez de datos; coincidiendo las hipótesis teóricas con los resultados experimentales obtenidos.
  - ✓ Las resistencias de pull-up empleadas son adecuadas ya que la forma de los pulsos de la señal es parecida a un pulso ideal.

7.3 Prueba de canales PWM

- **Objetivo**: Monitorizar mediante el osciloscopio algunas capturas que demuestren como la velocidad de los motores del módulo de locomoción se controla mediante la modulación por ancho de pulso explicada en el apartado 3.4.2.
- **Módulos involucrados:** Es necesario el uso de dos módulos.
  - ✓ Módulo de locomoción: encargado de ejecutar la orden que recibe por parte del módulo maestro.
  - ✓ Módulo maestro: encargado de enviar alguna orden concreta al módulo de locomoción mediante el protocolo I^2C.
- **Conexiones necesarias**: Es necesario realizar una serie de conexiones previas.
  1. Se conectan al bus I^2C de comunicaciones el módulo de locomoción y el maestro. Para ello se deben interconectar las señales SDA, SCL, y GND de los dos módulos.
  2. Se conectan las señales de alimentación (+5V.) de los dos módulos a una fuente de alimentación que entregue la tensión necesaria. Se conecta la señal de alimentación (>+5V.) para entregar tensión suficiente a los motores. En este caso se han alimentado los motores con 7,5V.

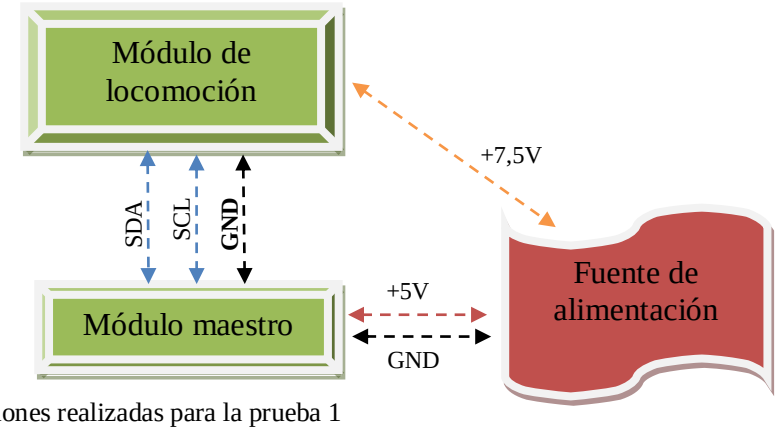


Figura 7-9: Conexiones realizadas para la prueba 1

- **Situación:** Al conectarse la alimentación, el módulo maestro establece comunicación con el esclavo número cero del módulo de locomoción mediante el protocolo I^2C con el objetivo de que el motor derecho retroceda de manera indefinida al 75% de su velocidad máxima y el motor izquierdo avance también de manera indefinida pero al 25% de su velocidad máxima.

Para que tales eventos ocurran es necesario ejecutar las siguientes instrucciones por parte del maestro: ***mover_motor (RETROCEDER, 75, MOTOR_DERECHO, 0)***;

mover_motor (AVANZAR, 25, MOTOR_IZQUIERDO, 0);

La ejecución de las instrucciones anteriores conlleva una serie de comunicaciones entre el maestro y el esclavo pero que no se analizarán en esta prueba ya que aquí únicamente pretende analizarse el resultado que genera dicha comunicación.

- **Resultados:** Los resultados obtenidos se muestran mediante un osciloscopio. La motivación de usar un osciloscopio es poder analizar con mayor detalle la forma de las señales de PWM generadas. Las sondas del osciloscopio se han conectado a los conectores de los motores del módulo de locomoción. Concretamente,
  - Se ha conectado la sonda de color amarillo a la señal M+ del conector para el motor derecho del módulo de locomoción.
  - Se ha conectado la sonda de color verde a la señal M+ del conector para el motor izquierdo del módulo de locomoción.
  - Se ha tomado como valor de GND para la sonda de color amarillo la señal M- del conector derecho y para la de color verde la señal M- del motor izquierdo.

Se ha tomado en esa situación la captura de pantalla que se muestra en la Figura 7-10 y en la que se aprecian varias cosas. En primer lugar se observan las condiciones en las que se ha tomado la captura: 2,5V. por división vertical, y 20µs. por división horizontal.

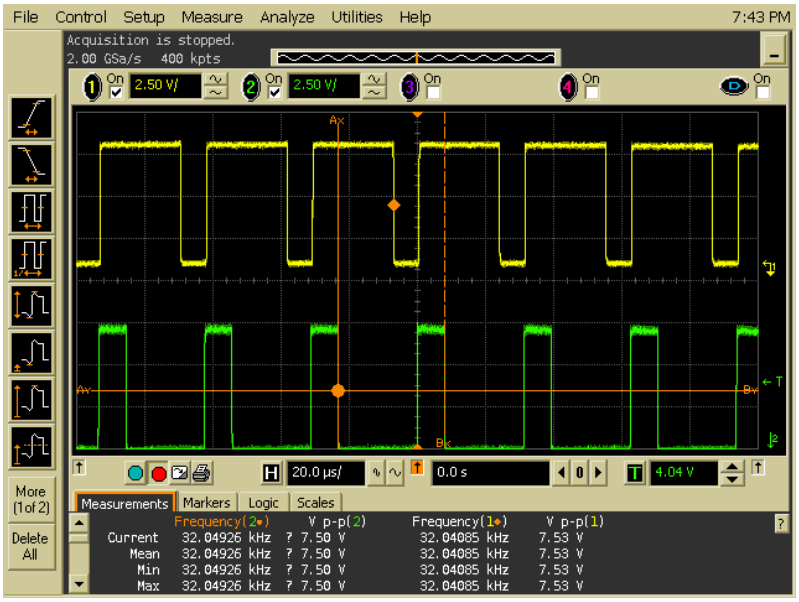


Figura 7-10: Captura de pantalla tomada número 1

Se observa también la frecuencia con la que se generan las señales de PWM (~32kHz.) tal como se especificó en el apartado 3.7. Nótese que la tensión pico a pico que recibirían los motores en el caso de estar conectados es de ~7,5V; que coincide con la entregada por la fuente de alimentación.

A continuación, se ha tomado otra captura de la misma situación pero esta vez incluyendo otras medidas de interés que pueden observarse en la parte inferior de la captura de pantalla tomada. Se muestra dicha imagen en la Figura 7-11. Las medidas incluídas en dicha figura son, en primer lugar los ciclos útiles de las señal PWM (duty cycle) que coincide de manera aproximada con las órdenes enviádas por el maestro, 25,7% (~25%) en el motor izquierdo (sonda verde) y 75,7% (~75%) en el motor derecho (sonda amarilla). El ciclo útil empleado en cada caso hace que la tensión media entregada sea 1,88V. (aproximadamente el 25% de 7,5V.) en un caso y 5,16V. (aproximadamente el 75% de 7,5V.) en el otro.

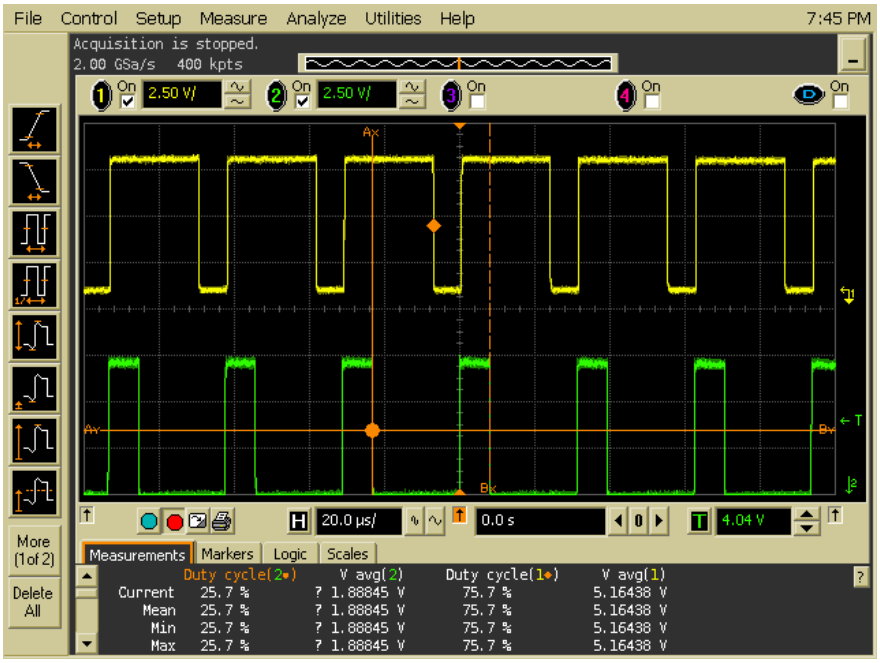


Figura 7-11: Captura de pantalla tomada número 2

En último lugar se ha tomado otra captura de pantalla, esta vez con el objetivo de calcular el desfase existente entre las dos señales de PWM generado. Dicho desfase es constante, de valor 16µs tal como puede observarse en la parte inferior de la Figura 7-12.

La diferencia entre ambas señales se debe al hecho de que el maestro ordena primero al motor derecho que se mueva y posteriormente al izquierdo.

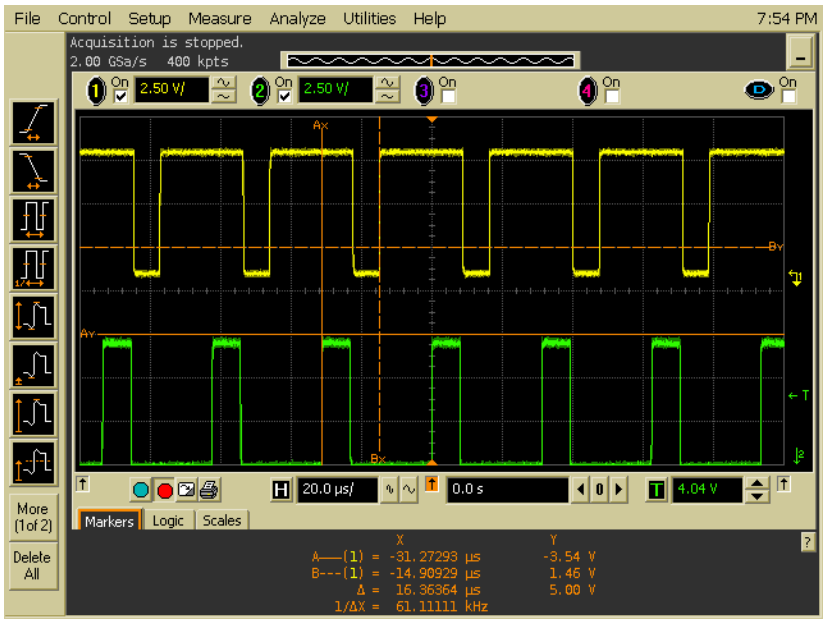


Figura 7-12: Captura de pantalla tomada número 3

- **Conclusión:** Del éxito de la prueba realizada se concluye lo siguiente:
  - ✓ La frecuencia a la que se generan las señales de PWM es de 32kHz tal como se esperaba. Dicha frecuencia es suficientemente alta como para no ser audible por el ser humano y suficientemente baja como para no generar ruido eléctrico.
  - ✓ Es posible hacer que ambos motores reciban tensiones diferentes gracias a la modulación por ancho de pulso. El ciclo de trabajo de las señales PWM, que llegan a los motores, ordenada por el maestro, es correcto.
  - ✓ Existe un desfase entre las señales que llegan al motor derecho y al motor izquierdo si se ejecutan dos órdenes de manera secuencial (una para cada motor).

8 Conclusiones y trabajo futuro

8.1 Conclusiones

Si bien lo que se pretendía en este PFC era construir los elementos necesarios para montar una plataforma genérica modular para desarrollo con robots móviles; lo que se ha conseguido es mucho más que eso.

- ✓ Se han creado una serie de módulos capaces de cooperar para resolver un problema.
- ✓ Se ha establecido una jerarquía en la que un módulo es capaz de dividir el problema en subproblemas más simples y dar las órdenes necesarias para que cada uno de los módulos esclavos resuelva una parte.
- ✓ Se han demostrado las ventajas de una plataforma modular, en cuanto a la optimización de: consumo (la resolución de tareas que impliquen un menor número de módulos requerirán un menor consumo energético), espacio (los sistemas que requieran el uso de menos módulos tendrán un tamaño menor), etc.
- ✓ Se ha diseñado una plataforma muy robusta, capaz de controlar 8 motores, 4 pantallas LCD, 8 puertos USB, 8 puertos serie RS-232, 16 sensores de ultrasonidos y 28 sensores de infrarrojos (teniendo en cuenta ciertas combinaciones incompatibles entre ultrasonidos e infrarrojos).
- ✓ Se han implementado dos versiones de un mismo tipo de módulo para que se ajuste mejor a la tarea que se pretende resolver; por ejemplo, si la tarea solo requiere puertos USB ó serie, se debe usar el módulo A de comunicaciones en vez de usar el B, consiguiendo un ahorro económico, energético, de espacio físico, etc.
- ✓ Los módulos han sido diseñados de manera que es posible "apilar" unos sobre otros consiguiendo una estructura en forma de torre, para ello se han colocado en cuatro lugares estratégicos unos agujeros de métrica 2,8mm. Además, en dicha configuración no sería necesario interconectar mediante cables los módulos sino que gracias a la colocación de los conectores de I^2C y de alimentación, quedarían directamente interconectados unos con otros por "contacto" reduciendo el ruido eléctrico y la probabilidad de diafonía.
- ✓ Se ha querido mostrar como un mismo módulo es capaz de realizar tareas atendiendo a otro módulo maestro por I^2C o mediante USB; en ese último caso, el

maestro podría ser un ordenador, un usuario o incluso otra plataforma robótica transmitiendo información para una colaboración a más alto nivel. Esta dualidad de puertos para recibir órdenes es muy interesante ya que hace que el número de elementos capaces de controlar este módulo crezca exponencialmente. Dicha funcionalidad ha sido implementada en el modelo B del módulo de comunicaciones.

- ✓ Se ha demostrado como la convivencia entre diferentes protocolos (SPI, I^2C, USB, RS-232) dentro de un mismo módulo es no solo viable, sino enriquecedora en cuanto al uso óptimo de transmisión y recepción de información en cada caso.
- ✓ Y para finalizar, se ha querido mostrar de una forma un tanto distendida el uso de todos los módulos cooperando entre ellos de manera eficaz resolviendo la tarea de encontrar un hueco entre una hilera de obstáculos y aparcar en él. No obstante, los módulos diseñados están dotados de tantas funcionalidades que existen cientos de aplicaciones más complejas que podrían resolverse haciendo uso de ellos.

8.2 Trabajo futuro

A continuación se listan algunos de los muchos módulos que me habría gustado diseñar pero que por falta de tiempo no pudo ser.

- Módulo capaz de controlar una cámara de video. Así el robot podría por ejemplo diferenciar colores o reconocer formas.
- Una tercera versión del módulo de comunicaciones que incluyera un puerto PS/2 para controlar un teclado (esto sí que se implementó y construyó pero no hubo tiempo para probarlo por completo de manera que finalmente no se incluyó en los diseños definitivos). Así por ejemplo podría dotarse al robot de una contraseña.
- Un módulo capaz de grabar sonido o voz y posteriormente reproducirla. Así el robot podría por ejemplo responder vocalmente ante determinados eventos, sin ir más lejos, podría reproducir la frase grabada “he aparcado correctamente”.
- Un módulo capaz de transmitir información por radiofrecuencia, bluetooth o WiFi. Así, el robot podría, por ejemplo, mostrar por la pantalla LCD los mensajes del Skype o del Messenger.
- Un módulo capaz de controlar motores paso a paso y servomotores. Así podrían usarse cualquiera de los tres tipos de motores más usados en robótica.

9 Bibliografía

1. **Kernighan, Brian W.** The C programming language . s.l. : Prentice Hall, 1988. 0131103628.

2. [En línea] 25 de 11 de 2009. http://www.i2c-bus.org/i2c-bus/.

3. [En línea] 25 de 11 de 2009. http://www.esacademy.com/faq/i2c/general/i2chisto.htm.

4. [En línea] 25 de 11 de 2009. http://www.esacademy.com/faq/i2c/general/i2cspecver.htm.

5. [En línea] 25 de 11 de 2009. www.nxp.com.

6. [En línea] 25 de 11 de 2009. http://www.i2c-bus.org/twi-bus/.

7. [En línea] 25 de 11 de 2009. http://www.i2c-bus.org/how-i2c-hardware-works/.

8. **García, Antonio F. Díaz.** [En línea] 25 de 11 de 2009. http://atc.ugr.es/~afdiaz/fich/bus_i2c.pdf.

9. [En línea] 25 de 11 de 2009. http://robots-argentina.com.ar/Comunicacion_busI2C.htm .

10. [En línea] 25 de 11 de 2009. http://www.i2c-bus.org/how-i2c-hardware-works/.

11. [En línea] 25 de 11 de 2009. http://www.alegsa.com.ar/Dic/spi.php.

12. [En línea] 25 de 11 de 2009. http://es.wikipedia.org/wiki/Serial_Peripheral_Interface.

13. [En línea] 25 de 11 de 2009. http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus#cite_note-0.

14. SPI Block Guide- Freescale Semiconductor. [En línea] V3.06, 26 de 10 de 2007. http://www.freescale.com/files/microcontrollers/doc/ref_manual/S12SPIV3.pdf.

15. [En línea] 25 de 11 de 2009. http://es.wikipedia.org/wiki/Serial_Peripheral_Interface.

16. [En línea] 25 de 11 de 2009. http://www.infowarehouse.com.ve/pugoz/ingelect/ingelec_motorcc.pdf.

17. [En línea] 25 de 11 de 2009. http://es.wikipedia.org/wiki/Fuerza_de_Lorentz.

18. **Horn, Delton T.** Basic Electronics Theory. Fourth Edition. s.l. : Mcgraw-Hill Publishing Co, 1994. 0-8306-4200-5.

19. [En línea] 25 de 11 de 2009. www.robotroom.com/DistanceSensor3.html.

20. [En línea] 25 de 11 de 2009. www.terra.es/personal/fremiro/Archivos/GP2d12%20.pdf.

21. Fundamentals of RS-232 Serial Communications. [En línea] APPLICATION NOTE 83, 21 de 3 de 2001. http://www.maxim-ic.com/appnotes.cfm/an_pk/83/.

22. [En línea] 25 de 11 de 2009. http://www.lammertbies.nl/comm/cable/RS-232.html.

23. [En línea] 25 de 11 de 2009. http://www.monografias.com/trabajos11/usbmem/usbmem.shtml.

24. **Axelson, Janet L.** USB Complete: Everything you need to Develop Custom USB Peripherals, Third Edition. s.l. : Lakeview Research., 2005. 1-931448-02-7.

25. [En línea] 25 de 11 de 2009. http://www.xataka.com/accesorios/usb-30-a-fondo.

26. [En línea] 25 de 11 de 2009. http://es.wikipedia.org/wiki/Universal_Serial_Bus.

27. **Philips, NXP founded by.** I2C-bus specification and user manual. [En línea] Rev. 3.0, 19 de 07 de 2007. http://www.esacademy.com/faq/i2c/general/i2cspecver.htm. UM10204.

28. Revista uControl "Electronica en general Pics en particular". Número 1, 2008. Disponible en http://www.ucontrol.com.ar/revista/0001/ucontrol_revista_0001.pdf.

29. **Mitzner, Kraig.** Complete PCB Design Using OrCad Capture and Layout. s.l. : Elsevier/Newnes, 2007. 978-0-7506-8214-5.

30. [En línea] 25 de 11 de 2009. http://www.cadsoftusa.com/Tour/tour00.htm.

31. [En línea] 25 de 11 de 2009. http://www.euskalnet.net/shizuka/rs232.htm.

32. **Richard H. Barnett, Larry O'Cull, Sarah Cox.** Embedded C programming and the Atmel AVR. s.l. : Thomson Delmar Learning, 2006. 9781418039592.

33. [En línea] 25 de 11 de 2009. http://es.wikipedia.org/wiki/Plug-and-play.

34. [En línea] 25 de 11 de 2009. http://es.wikipedia.org/wiki/RS-232.

Apéndice

Junto con este PFC se adjunta un CD cuyo contenido es el que se lista a continuación:

- ✓ Todos los archivos que componen el código fuente del módulo de locomoción (***firmware.c, locomoción.c, locomoción.h, USI_Slave.c, USI_Slave.h y Makefile***)
- ✓ Todos los archivos que componen el código fuente del módulo US&IR (***firmware.c, infrarrojos.c, infrarrojos.h, ultrasonidos.c, ultrasonidos.h, tablas_magicas.h, generador_tabla.c, USI_Slave.c, USI_Slave.h y Makefile***)
- ✓ Todos los archivos que componen el código fuente del módulo de comunicaciones A (***firmware.c, comunicaciones.c, comunicaciones.h, USI_Slave.c, USI_Slave.h y Makefile***)
- ✓ Todos los archivos que componen el código fuente del módulo de comunicaciones B (***firmware.c, lcd.c, lcd.h, comunicaciones.c, comunicaciones.h, letras.h, imagen_logos.h, delay_x.h, TWI_Slave.c, TWI_Slave.h y Makefile***)
- ✓ La API implementada para que el maestro pueda controlar todos los módulos (***locomoción.c, locomoción.h, US_IR.c, US_IR.h, comunicaciones_A.c, comunicaciones_A.h, comunicaciones_B.c, comunicaciones_B.h, decodificador.c, USI_Master.c, USI_Master.h***)
- ✓ Todo el código citado anteriormente también en versión imprimible.
- ✓ El código implementado para la aplicación de ejemplo (***firmware.c y Makefile***)
- ✓ Las librerías para el control de la pantalla LCD del módulo de comunicaciones B tanto para Windows como para Linux (***LCD_IRB.h, LCD_linux_IRB.a, LCD_win_IRB.lib y Makefile***)
- ✓ Los esquemáticos y layouts de los cuatro módulos esclavos, y el esquemático del maestro, todo ello realizados en Eagle
- ✓ Versión electrónica del presente PFC

Glosario

SPI	*Serial Peripheral Interface*
I^2C	*Inter-Integrated Circuit*
TWI	*Two Wire Interface*
PWM	*Pulse Width Modulation*
USI	*Universal Serial Interface*
PCB	*Printed Circuit Board*
DSP	*Digital Signal Processor*
MSBF	*Most Significant Bit First*
RoHS	*Restriction of Hazardous Substances*
RGB	*Red, Green, Blue*
USART	*Universal Synchronous/Asynchronous Receiver Transmitter*
MSB	*Most Significant Byte*
LSB	*Least Significant Byte*
S.O.	*Sistema Operativo*
PC	*Personal Computer*
ADC	*Analog to Digital Converter*
IC	*Integrated Circuit*
PSD	*Position Sensitive Detector*
ASCII	*American Standard Code for Information Interchange*
API	*Application programming interface*
DTE	*Data Terminal Equipment*
DCE	*Data Circuit-Terminating Equipment*
SMD	*Surface Mounted Device*
USB	*Universal Serial Bus*
IEEE	*Institute of Electrical and Electronics Engineers*

Anexos

A Valores recomendados para las resistencias de I^2C

La revisión más actual acerca de las especificaciones para un bus I^2C a la hora de escribir estas líneas es la número 3 del 19 de junio del 2007 (27); en ella se recogen tres gráficas a partir de las cuales determinar el valor máximo y mínimo para las resistencias de pull-up, Rp, y las resistencias serie, Rs.

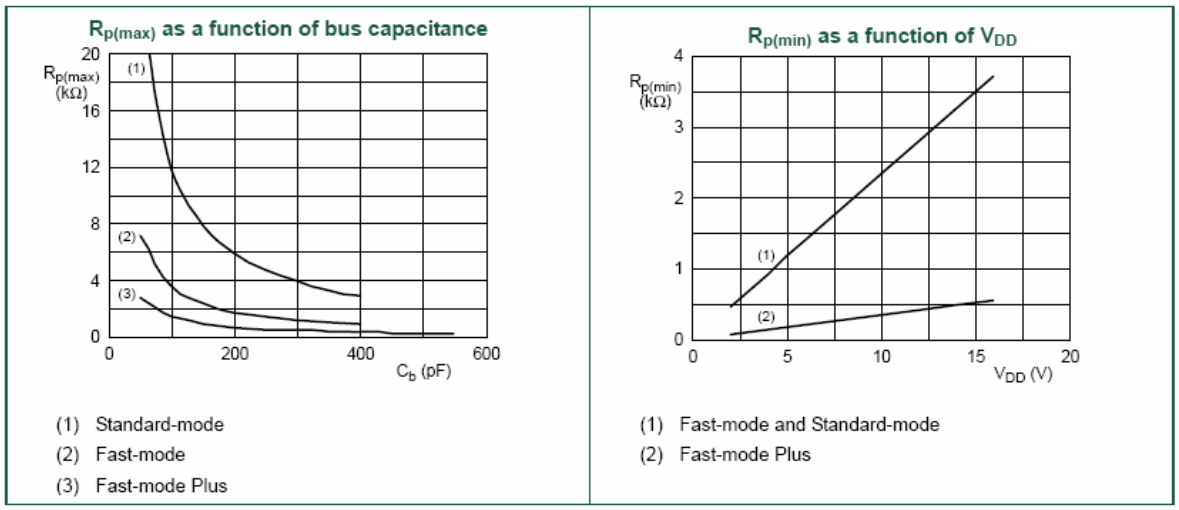


Figura A-1: Gráficas para el cálculo de las resistencias de pull-up

A partir de las dos gráficas anteriores se observa que para una tensión de alimentación como la que usamos en la placa de locomoción, 5V., se recomienda usar resistencias de pull-up con un valor mínimo de 1k2Ω. para el modo Standard y Fast y 215Ω. para el modo Fast Plus.

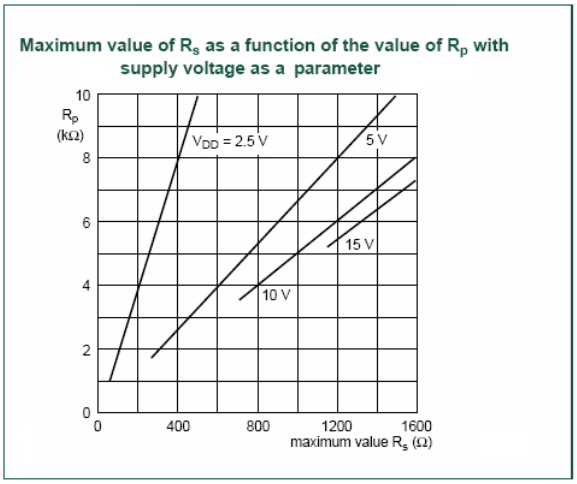


Figura A-2: Gráfica para el cálculo de las resistencias serie

A partir de la gráfica anterior se observa que para una tensión de alimentación como la que usamos en la placa de locomoción, 5V., y una resistencia Rp de 4k7Ω. se recomienda usar resistencias serie con un valor de 700Ω.

A continuación se muestran las direcciones reservadas en el protocolo de comunicación I^2C según la revisión de las especificaciones citada al inicio de este anexo.

Debe tenerse en cuenta, que aunque algunas direcciones estén reservadas, pueden usarse siempre y cuando se esté seguro de que no serán necesarias en el bus.

Dirección del esclavo	Bit de lectura/escritura	Descripción
0000 000	0	**Llamada general**
0000 000	1	**Byte de START**
0000 001	X	**Direccionamiento CBUS**
0000 010	X	**Reservado para diferentes formatos de bus**
0000 011	X	**Reservado para uso futuro**
0000 1XX	X	**Código maestro en modo high speed**
1111 1XX	X	**Reservado para uso futuro**
1111 0XX	X	**Direccionamiento de esclavos de 10 bits**

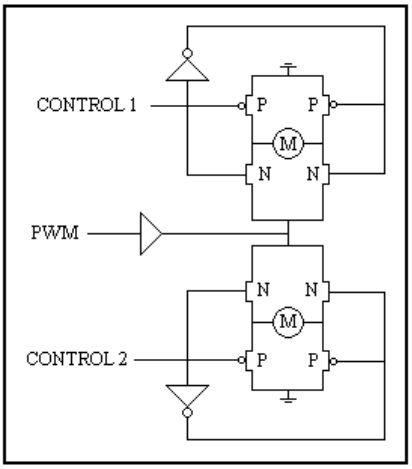
Tabla A-1: Direcciones reservadas en el bus I^2C

B Alternativas para reducir el número de canales PWM

A continuación se muestra una posible alternativa que permite controlar dos motores con tan solo un canal de PWM y dos señales de control, una para cada motor. La idea se basa en la aplicación conjunta de Puentes H y tecnología de PWM, como puede verse en los esquemas que se muestran a continuación.

La alternativa fue rechazada porque requería añadir ocho transistores, un buffer y dos inversores lógicos, lo cual además de ser más costoso tanto económicamente como a la hora de diseñar, introduciría ruido que con el uso de cuatro canales de PWM se ahorra. Además, ambos motores girarían necesariamente a la misma velocidad, lo cual resta funcionalidad.

Bajo estas líneas se muestra la alternativa descrita anteriormente. En la parte izquierda aparecen las tres señales requeridas para el control de los motores, CONTROL1, CONTROL2 y PWM.



La señal PWM será la encargada de dotar a los motores de la velocidad que se desee dependiendo de su ciclo de trabajo útil, como ya se vio en el apartado correspondiente.

La señal CONTROL 1 es la encargada de controlar el sentido de giro del motor de la parte superior mientras la señal CONTROL 2 es la encargada de controlar el giro del motor de la parte inferior de la Figura B-3.

Figura B-3: Diseño alternativo

Si la señal de control tiene valor uno lógico, entonces se activan los transistores P y N correspondientes (en color rojo) para que la señal PWM recorra el motor produciendo giro en un sentido concreto, como muestra la Figura B-4 izquierdo.

Si por el contrario, la señal de control tiene valor cero lógico, entonces se activarán los transistores P y N correspondientes (en color verde) para que la señal PWM recorra el motor produciendo giro en el otro sentido, como muestra la figura Figura B-4 derecha.

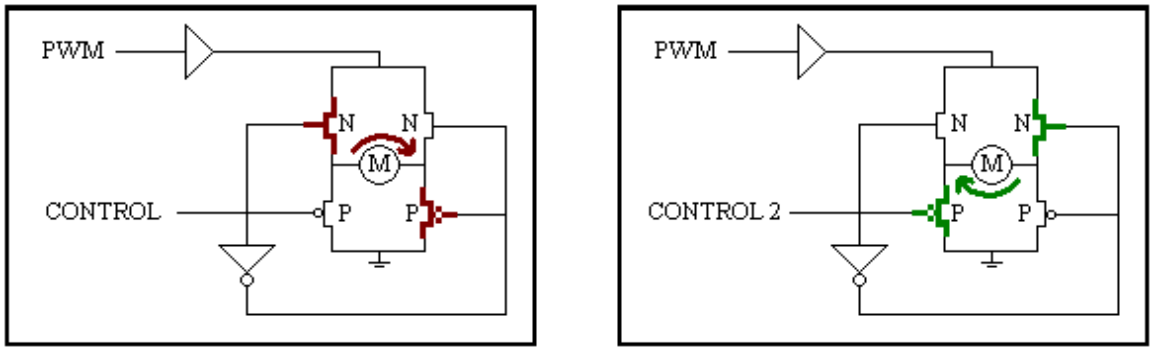


Figura B-4: Control con valor 1 lógico (izda.) y control con valor 0 lógico (dcha.)

Podría pensarse en el uso de puentes H implementados en circuitos integrados para evitar el ruido introducido por los transistores y hacer esta alternativa más viable, pero el problema es que los integrados comerciales (L293, TLE4207G, etc) no tienen la misma conexión interna que en la Figura B-3; requieren el uso de dos señales de control para un único motor y además esas señales de control deberían de ser PWM si se quiere controlar la velocidad, por lo cual se vuelve a cuatro canales de PWM para controlar los dos motores(65).

[65] Esa es precisamente la alternativa usada en el proyecto, el uso de cuatro canales de PWM mediante puentes H implementados en circuitos integrados.

C Probabilidad de error de la USART

La motivación de este anexo es mostrar una serie de tablas extraídas de las hojas de datos del microcontrolador ATtiny2313 y ATmega88 acerca del porcentaje de error cometido por la USART en función de la frecuencia del reloj y la velocidad de transmisión.

Baud Rate (bps)	f_{osc} = 1.0000 MHz				f_{osc} = 1.8432 MHz				f_{osc} = 2.0000 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max. (1)	62.5 kbps		125 kbps		115.2 kbps		230.4 kbps		125 kbps		250 kbps	

Tabla C-2: Porcentaje de error para frecuencias de 1MHz. a 2MHz.

Baud Rate (bps)	f_{osc} = 3.6864 MHz				f_{osc} = 4.0000 MHz				f_{osc} = 7.3728 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max. (1)	230.4 kbps		460.8 kbps		250 kbps		0.5 Mbps		460.8 kbps		921.6 kbps	

Tabla C-3: Porcentaje de error para frecuencias de 3,6864MHz. a 7,3728MHz.

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max. (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

Tabla C-4: Direcciones Porcentaje de error para frecuencias de 8MHz. a 14,7456MHz.

Baud Rate (bps)	f_{osc} = 16.0000 MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max. (1)	1 Mbps		2 Mbps	

Tabla C-5: Porcentaje de error para una frecuencia de 16MHz.

D Calibración del sensor de infrarrojos GP2D12

La motivación de este anexo es mostrar los resultados obtenidos fruto de la calibración del sensor de infrarrojos GP2D12 usado en este PFC.

El sensor GP2D12 entrega a su salida una tensión analógica proporcional a la distancia a la que se halla un objeto del mismo. La tensión entregada varía entre 2,4V. a los 10cm. y 400mV. a los 80cm.

Teniendo en cuenta que la salida del sensor no es lineal, es necesario hallar una función que asocie la salida a la distancia existente. Además, puesto que el fabricante acota el campo de trabajo entre 10 y 80cm., solo se tendrán en cuenta los valores en ese rango[(66)].

Para la calibración del sensor se ha colocado un objeto entre 10cm. y 80cm., alejándolo de un cm. en un cm. Así pues, se colocó el obstáculo a 10 cm., 11cm., 12 cm., etc.

La hoja de especificaciones del fabricante acerca del sensor hace referencia a la importancia del color del obstáculo. Se ha comprobado que las medidas cambian considerablemente en función del color del obstáculo.

En la calibración realizada se ha usado un obstáculo rectangular de 10 cm. de alto y 12 cm. de ancho de color blanco; es por esta razón que cuánto más difiera el color del obstáculo del color blanco, mayor será el error con respecto a los resultados que se presentan en este anexo y con los que trabaja el módulo US&IR.

Par calibrar de manera adecuada y fiable del sensor es necesario disponer de aparatos de medida los suficientemente precisos que nos permitan obtener una lectura del valor entregado a la salida sin oscilaciones, constante y con gran resolución. La calibración debería ser realizada en un laboratorio especializado y deberían tenerse en cuenta obstáculos de diferentes tamaños y colores.

En este PFC no se ha dispuesto de tales medios y tampoco se ha creído oportuno el conseguir una extrema precisión, ya que a pesar de conseguir una buena resolución, al integrar dicha medida en el sistema no estaría exenta de una cierta incertidumbre ya que no se conoce de antemano el entorno en el que se empleará el sensor (recuérdese que es para una plataforma genérica) ni se puede controlar el tipo de obstáculo al que se enfrentará el

[66] En numerosos estudios se ha demostrado que el campo de trabajo podría extenderse de 8 cm a casi 120 cm si bien el error cometido crece exponencialmente con el crecimiento del campo de trabajo.

sensor. De modo que se cree más conveniente trabajar con los posibles errores cometidos y tenerlos en cuenta para compensarlos de algún modo, por ejemplo promediando la medida o haciendo medidas conjuntas de sensores de ultrasonidos y sensores de infrarrojos.

En primer lugar se ha estimado el error obtenido por el ADC integrado en el microcontrolador empleado en el módulo US&IR. Para ello se ha anotado el escalón de cuantificación al que el microcontrolador asignaba cada valor analógico entregado por el sensor. Para la cuantificación se ha empleado un conversor analógico-digital de 10 bits. Por ello existen 1024 escalones de cuantificación posibles. El valor de referencia empleado para el conversor ha sido de 2,56V. ya que la máxima salida del sensor (se produce a 10cm.) tiene un valor de aproximadamente 2,40V. Así, la precisión del cuantificador es de 2.5mV.

En la Figura E-5 se presentan los resultados obtenidos al estimar el error del cuantificador. En línea continua se presenta la ecuación y=0.4x que representa la curva ideal del cuantificador descrito en el párrafo anterior.

Mediante mínimos cuadrados se ha hallado la ecuación que mejor se ajusta a los resultados experimentales obtenidos mostrados en línea puntuada obteniendo finalmente la ecuación y=0.4094x-6.0003. El error introducido por la cuantificación es del 2,35%.

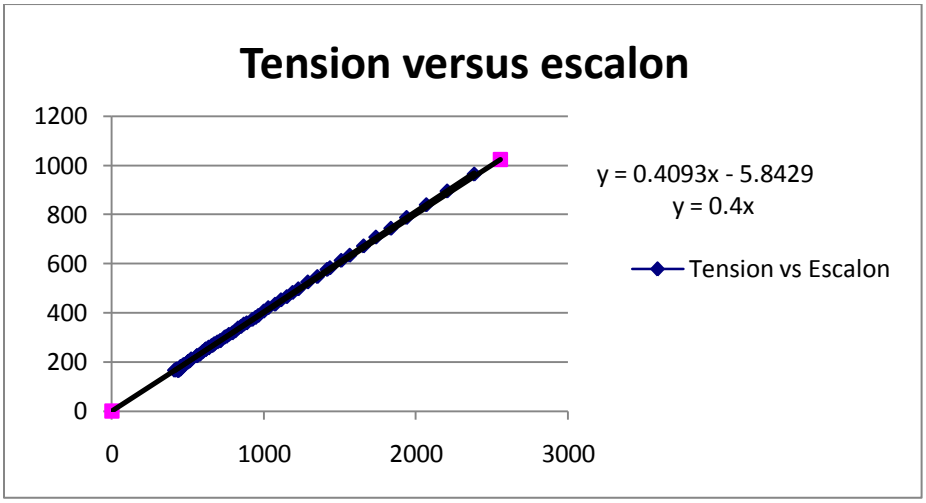


Figura E-5: Estimación del error de cuantificación del ADC

A continuación, se presentan en la Figura E-6 los resultados obtenidos que relacionan el escalón asignado por el cuantificador con la distancia entre el objeto y el sensor GP2D12.

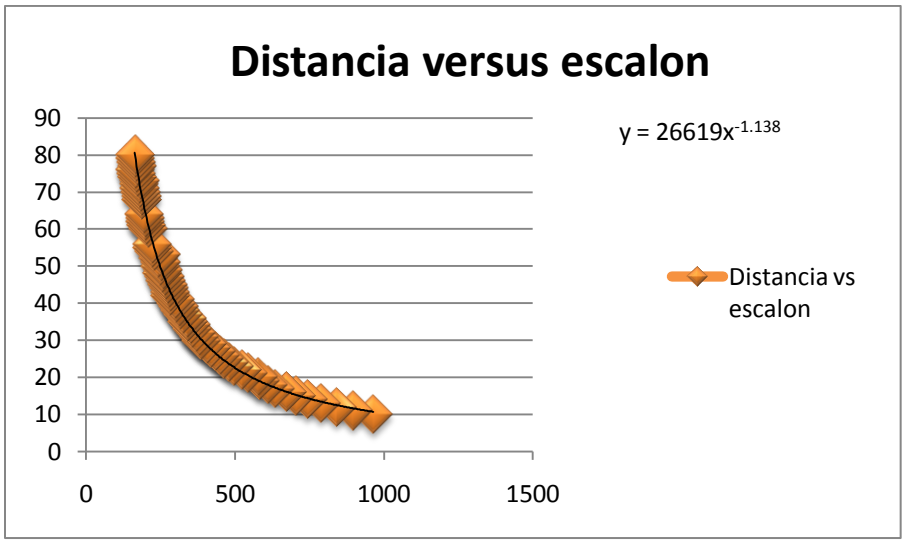


Figura E-6: Conversión de escalones de cuantificación en cm.

La línea negra representa la tendencia que mejor se ajusta a los resultados obtenidos experimentalmente. Se trata de una tendencia potencial, cuya ecuación es $26619x^{-1.138}$.

Es esa ecuación la que se usa en el módulo US&IR para convertir el escalón de cuantificación, asociado por el microcontrolador a la tensión entregada por el sensor, en la distancia (en cm.) que se halla entre el objeto y el GP2D12.

Finalmente, se presenta en la Figura E-7 la gráfica obtenida que relaciona la distancia entre el obstáculo y el sensor de infrarrojos con la tensión entregada por este último. En está gráfica no se ha introducido el error debido a la cuantificación y su utilidad es meramente informativa ya que no se puede trabajar con la tensión analógica sin cuantificar.

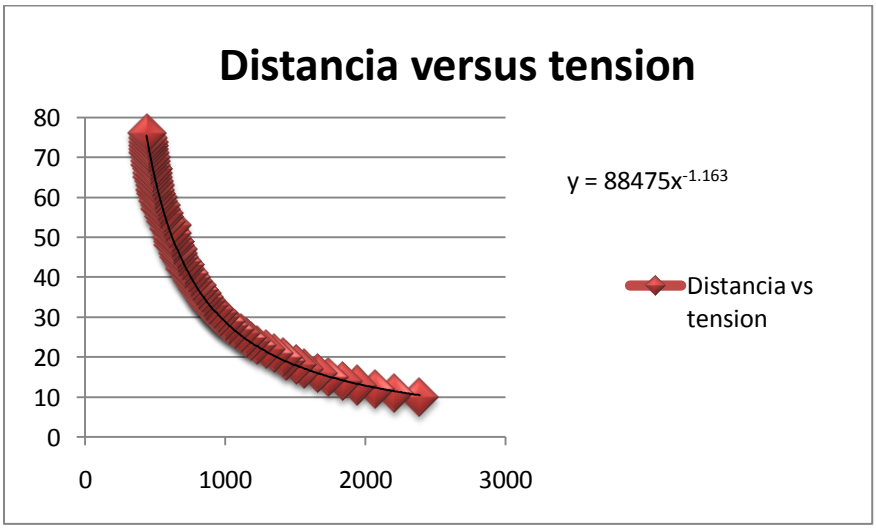


Figura E-7: Distancia en cm. en función de la tensión analógica entregada

E Modificación del firmware

En este anexo se pretende explicar una de las posibles formas en las que puede modificarse el programa del módulo maestro. Por el mismo procedimiento podría modificarse o actualizarse el firmware de los módulos esclavo, si bien esto no se recomienda ya que podrían quedar inutilizados. **Queda bajo la responsabilidad del usuario la modificación del firmware de los módulos esclavo.**

La modificación del firmware puede realizarse desde diferentes S.O. así, a continuación se detallan la forma y los programas que son necesarios instalar en el P.C dependiendo del S.O.[67]

E.1 En Windows

En primer lugar se descarga el programa gratuito **WinAVR**[68], que incluye una serie de herramientas muy útiles para trabajar con microcontroladores de la serie AVR de Atmel.

Ya que en este PFC se ha usado también un programador por puerto paralelo[69], hemos de configurar el sistema para que podamos trabajar con dicho puerto; para ello, basta con ejecutar el archivo install_giveio.bat[70] que se encuentra en *C:\WinAVR-20090313\bin*.

Para modificar el firmware, algunos módulos como US&IR, ejecutan parte del código en el PC. En tales casos, es necesario disponer del compilador *gcc*. Aunque el propio programa WinAVR dispone de gcc, se usará el compilador gcc de **MinGW**[71].

Para escribir / modificar el firmware se ha usado la herramienta **Programmer's Notepad**, que viene incluida con WinAVR.

Una vez modificado el programa pulse sobre *Tools-›[WinAVR] Make all* para ejecutar el archivo makefile correspondiente que se muestra en el apartado D.4 de este anexo.

---

[67] Probado en Windows XP, Windows 7 (Compilación 7600) y Linux Kubuntu 9.10 (Kernel 2.6.28.16)

[68] La descarga se realiza de http://sourceforge.net/projects/winavr/files/. A la hora de escribir estas líneas, la versión más actual es la del 13 de Marzo de 2009.

[69] Se ha usado tanto el programador por puerto paralelo "alf" como por USB "USBasp", ver anexo E.

[70] En Windows XP basta con hacer doble click sobre él. En Windows 7 es necesario ejecutarlo como administrador y en modo de compatibilidad con Windows XP Service Pack 3.

[71] Se trata de implementación de los compiladores GCC para la plataforma Win32, que permite migrar la capacidad de este compilador en entornos Windows. Puede descargarse la última versión de https://sourceforge.net/project/showfiles.php?group_id=2435&package_id=240780

E.2 En Linux

En Linux es necesario instalar una serie de paquetes para obtener la funcionalidad necesaria para modificar el programa del módulo maestro o cargar el firmware de los módulos esclavos. Los paquetes necesarios son gcc-avr, binutils-avr, avr-libc y avrdude.

En la Figura G-8 se muestra la manera de instalar dichos paquetes. En este caso, el sistema informa de que ya están instalados en su versión más reciente.

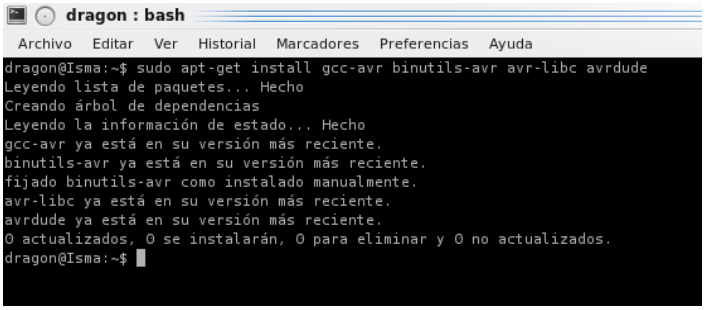


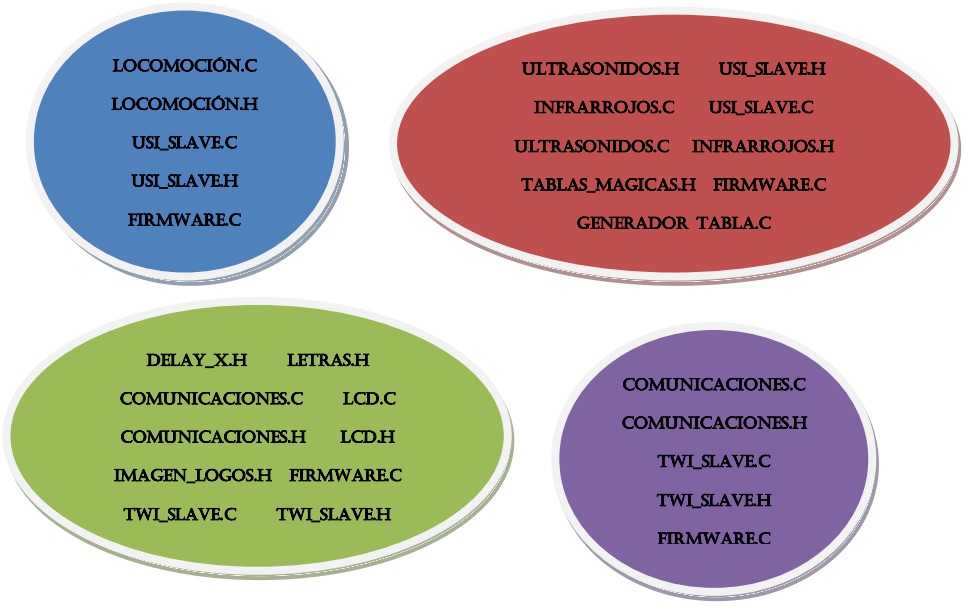
Figura G-8: Instalación de paquetes necesarios en linux

E.3 Archivos necesarios

Cada uno de los módulos se compone de una serie de archivos sin los cuales no podría funcionar ni modificarse su firmware. La motivación de este apartado es dar a conocer cuáles son esos archivos dependiendo del módulo que se desea modificar.

En la Figura G-9 se muestran una serie de elipses coloreadas. Cada una de las elipses representa un módulo, concretamente:

- En la azul aparecen los archivos relativos al módulo de locomoción.
- En la morada aparecen los archivos relativos al módulo de comunicaciones A.
- En la verde aparecen los archivos relativos al módulo de comunicaciones B.
- En la roja aparecen los archivos relativos al módulo US&IR.
- En la naranja aparecen los archivos obligatorios y opcionales del maestro.



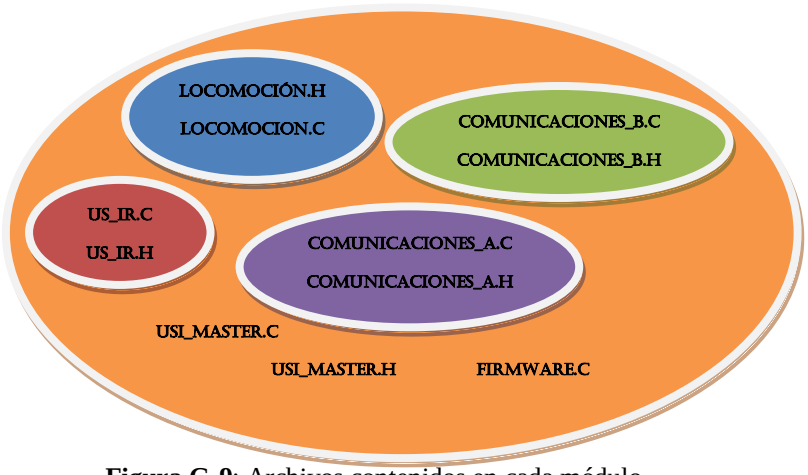


Figura G-9: Archivos contenidos en cada módulo

E.4 Makefile

Para este PFC concreto, se ha creado un archivo Makefile con campos configurables de modo que se adapte fácilmente un mismo esqueleto a los diferentes módulos. En la Figura G-10 se muestra el archivo en cuestión.

```
#############################################################
##        Makefile creado por Ismail Rebah Bouaiachi       ##
##  Plataforma generica para desarrollo con robots móviles ##
##                                                         ##
##                          2009                           ##
##                                                         ##
#############################################################
nucleo         =[redacted]
nucleo_abrev   =[redacted]
programador    =[redacted]
modulos_c      =[redacted]
modulos_h      =[redacted]
objetos        =[redacted]
tipo_I2C       =[redacted]
fuse_bajo      =[redacted]
fuse_alto      =[redacted]
fuse_extendido =[redacted]
arch_aux       =[redacted]
ARG            =[redacted]

all: $(arch_aux) $(tipo_I2C).h $(modulos_h)
    avr-gcc -g -Os -Wall -I. -mmcu=$(nucleo) -I/usr/lib/avr/include/ -c firmware.c $(tipo_I2C).c $(modulos_c)
    avr-gcc -g -mmcu=$(nucleo) -o firmware.elf firmware.o $(tipo_I2C).o $(objetos)

    avr-objcopy -j .text -j.data -O ihex firmware.elf firmware.hex
    avrdude -c $(programador) -p $(nucleo_abrev) -U flash:w:firmware.hex:i

fuses:
    avrdude -c $(programador) -p $(nucleo_abrev) -U lfuse:w:$(fuse_bajo):m
    avrdude -c $(programador) -p $(nucleo_abrev) -U hfuse:w:$(fuse_alto):m
    avrdude -c $(programador) -p $(nucleo_abrev) -U efuse:w:$(fuse_extendido):m

clean:
    rm -f *.o *.elf *.hex *~ *.exe $(arch_aux)

$(arch_aux): [redacted]
    [redacted]
```

Figura G-10: Makefile genérico para el PFC

A continuación se presentará en una serie de figuras el contenido que debe tener cada uno de los campos dependiendo del módulo en el que se desea usar en el caso del programador por puerto paralelo. En el caso de que desee usarse el programador por puerto USB debe sustituirse **"alf"** por **"USBasp"**.

En la Figura G-11 a la izquierda se muestra el contenido para el módulo de locomoción, y a la derecha para el modelo A del módulo de comunicaciones.

```
nucleo          = attiny2313
nucleo_abrev    = t2313
programador     = alf
modulos_c       = locomocion.c
modulos_h       = locomocion.h
objetos         = locomocion.o
tipo_I2C        = USI_Slave
fuse_bajo       = 0xC4
fuse_alto       = 0xDF
fuse_extendido  = 0xFF
arch_aux        =
```

```
nucleo          = attiny2313
nucleo_abrev    = t2313
programador     = alf
modulos_c       = comunicaciones.c
modulos_h       = comunicaciones.h
objetos         = comunicaciones.o
tipo_I2C        = USI_Slave
fuse_bajo       = 0xEC
fuse_alto       = 0xDF
fuse_extendido  = 0xFF
arch_aux        =
```

Figura G-11: Contenido para locomoción (izda.) y comunicaciones A (dcha.)

En Figura G-12 se muestra el contenido para el módulo US&IR.

```
nucleo          = attiny461
nucleo_abrev    = t461
programador     = alf
modulos_c       = infrarrojos.c ultrasonidos.c
modulos_h       = infrarrojos.h ultrasonidos.h
objetos         = infrarrojos.o ultrasonidos.o
tipo_I2C        = USI_Slave
fuse_bajo       = 0xDF
fuse_alto       = 0xDF
fuse_extendido  = 0xFF
arch_aux        = tablas_magicas.h

$(arch_aux): generador_tabla.c
    C:\MinGW\bin\gcc -c generador_tabla.c
    C:\MinGW\bin\gcc -g -Wall -o generador_tabla.exe generador_tabla.o
    ./generador_tabla.exe
```

Figura G-12: Contenido para US&IR

En la Figura G-13 se muestra el contenido para el modelo B del módulo de comunicaciones.

```
nucleo          = atmega88
nucleo_abrev    = m88
programador     = alf
modulos_c       = comunicaciones.c lcd.c
modulos_h       = comunicaciones.h lcd.h delay_x.h letras.h imagen_logos.h
objetos         = comunicaciones.o lcd.o
tipo_I2C        = TWI_Slave
fuse_bajo       = 0xEC
fuse_alto       = 0xDF
fuse_extendido  = 0x01
arch_aux        =
```

Figura G-13: Contenido para comunicaciones B

Finalmente, en la Figura G-14 se muestra el contenido para el maestro. Se ha recuadrado en color la parte opcional que depende de los módulos que se deseen añadir.

```
nucleo          = attiny44
nucleo_abrev    = t44
programador     = alf
modulos_c       = locomocion.c US_IR.c comunicaciones_B.c comunicaciones_A.c
modulos_h       = locomocion.h US_IR.h comunicaciones_B.h comunicaciones_A.h
objetos         = locomocion.o US_IR.o comunicaciones_B.o comunicaciones_A.o
tipo_I2C        = USI_Master
fuse_bajo       = 0xC2
fuse_alto       = 0xDF
fuse_extendido  = 0xFF
arch_aux        = imagen.h
ARG             = Dibujo.bmp
$(arch_aux): decodificador.c
    C:\MinGW\bin\gcc -c decodificador.c
    C:\MinGW\bin\gcc -o decodificador decodificador.o -lm
    ./decodificador $(ARG)
    rm -f imagen_t.h
```

Figura G-14: Contenido para el maestro

F Construcción física del PCB

En este anexo se pretenden mostrar todos los pasos necesarios para poder construir los módulos descritos en este PFC. Se explica, por tanto, el camino a seguir desde el layout hasta que el módulo existe en la realidad.

Tanto los esquemáticos como el layout y las imágenes 3D[72] han sido generados con el programa Eagle Layout Editor de la empresa CadSoft. Se trata de un programa gratuito y multiplataforma, disponible para Windows, Linux y Mac.

En el momento en el que se escriben estas líneas, la versión más actual de Eagle es ***Eagle Layout Editor 5.6.0*** para los tres S.O.

Una vez que se tiene el layout de cada uno de los módulos es necesario usar una máquina que lo entienda y lo transforme en un módulo real. En este PFC en particular se usará el *plotter fresador ProtoMat S100* de LPKF. Este plotter espera un archivo *.lmd que será combinación de los Gerber generados anteriormente. Dicho archivo será creado con el software ***CircuitCAM 6.0***.

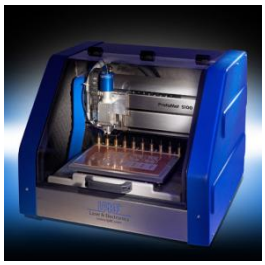


Figura E-15: ProtoMat S100

A continuación se describirá el proceso referido a la construcción del módulo de locomoción, para los demás módulos basta con cambiar donde ponga *locomoción* por el nombre del módulo que desee construirse.

El primer paso consiste en generar cuatro ficheros Gerber[73], ***locomocion.top***, ***locomocion.bot***, ***locomocion.fab*** y ***locomocion.drd***. La generación de estos archivos es muy sencilla. Locomocion.top contendrá información acerca de la capa superior del PCB, locomocion.bot contendrá información acerca de la capa inferior, locomocion.fab contendrá información acerca del contorno y finalmente locomocion.drd contendrá información acerca del taladrado.

[72] En realidad para las imágenes se ha usado el programa Eagle 3D en combinación con el programa POV-Ray, ambos gratuitos.

[73] Cuatro es el número mínimo de ficheros necesarios para generar un PCB y son .top, .bot., .fab y .drd.

1. Abrá el programa pulsando sobre EAGLE 5.6.0
2. El proyecto debe estar situado, por defecto, en la carpeta */Mis Documentos/Eagle*. Si es así, abrá el proyecto pulsando sobre Open Project como se muestra en la Figura E-16, expandiendo Projects/Eagle y pulsando con el botón secundario del ratón sobre la carpeta *Locomoción*.

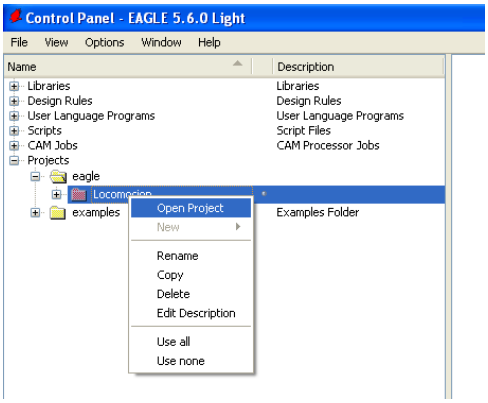


Figura E-16: Panel de control de Eagle 5.6.0

3. Se abrirán ahora dos ventanas, una con el esquemático del proyecto (*locomoción.sch*) y otra con el layout (*locomoción.brd*) del mismo.
4. De momento interesa el layout. Ejecute CAM Processor como se muestra en la Figura E-17, pulsando sobre el botón señalado en color rojo. A continuación seleccione en la casilla Device *GERBER_RS274X*, en la casilla File teclee *.../Mis documentos/eagle/ Locomocion/locomocion.top* y en el panel de la derecha seleccione únicamente las capas Top, Pads, Vias y Dimension.

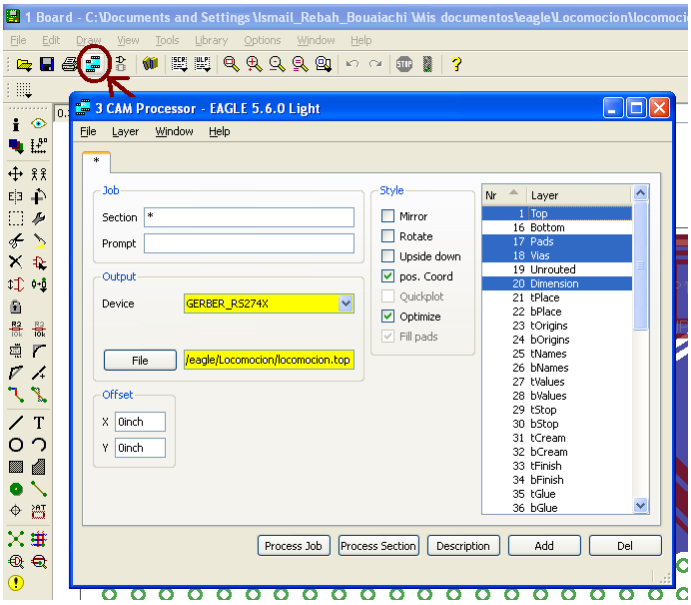


Figura E-17: Generación de locomocion.top

5. Pulse sobre el primer botón inferior Process Job. Si no sale ningún mensaje de error significará que se ha creado correctamente el primero de los ficheros Gerber, *locomoción.top*.

6. A continuación debe crear *locomocion.bot*; para ello, debe sustituir en File, donde pone locomocion.top, por locomocion.bot; además, a la derecha debe deseleccionar la capa Top, y seleccionar la capa Bottom, quedando como en la Figura E-18. Pulse Process Job.

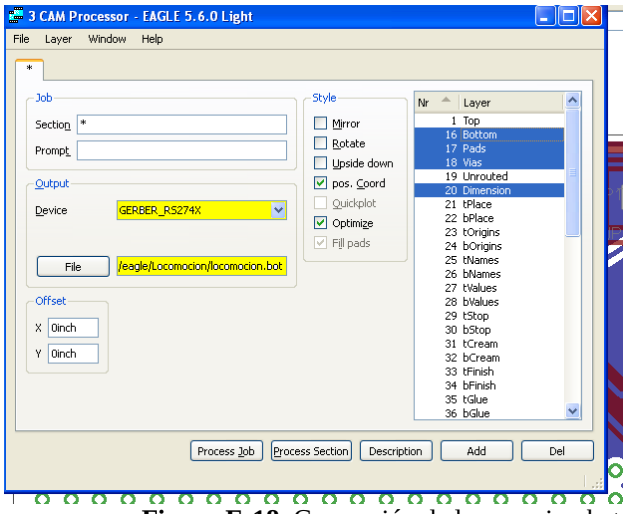


Figura E-18: Generación de locomocion.bot

7. Lo siguiente es crear *locomocion.fab*; para ello, debe sustituir en File, donde pone locomocion.bot, por locomocion.fab; además, a la derecha debe deseleccionar la capa Bottom y la capa Pads, quedando como en la Figura E-19. Pulse Process Job.

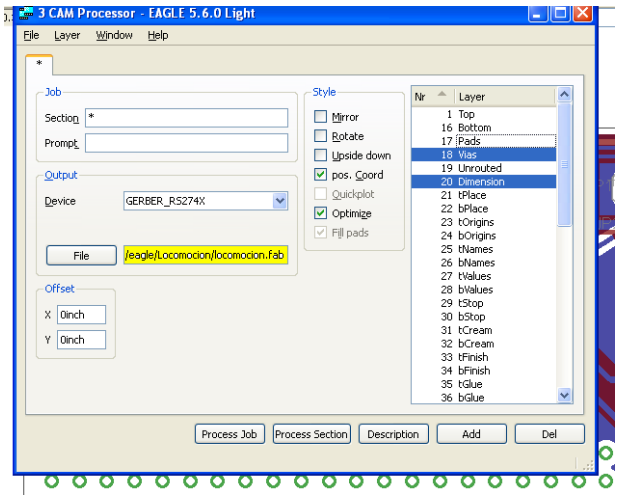


Figura E-19: Generación de locomocion.fab

8. Por último solo queda por generar el archivo locomocion.drd. La generación de este archivo es un poco más complicada que las anteriores. En primer lugar ha de pulsar el botón ULP de *locomocion.brd*, tras lo cual le aparecerá una ventana como la de la Figura E-20. En ella ha de seleccionar *drillcfg.ulp*.

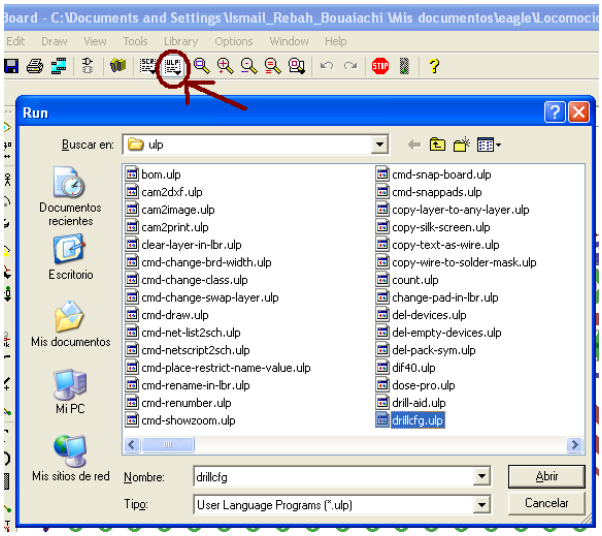


Figura E-20: Selección de drillcfg.ulp

9. Una vez que pulse el botón de abrir, le preguntará acerca de si quiere usar mm o pulgadas, ha de seleccionar pulgadas. Tras ello, le mostrará los taladros que se van a asociar al módulo. Podría añadir o suprimir taladros en función de si tiene físicamente o no dichas herramientas. Pulse sobre el botón Ok y guarde el archivo que se genera con el nombre *locomocion.drl* cuando se lo pida.

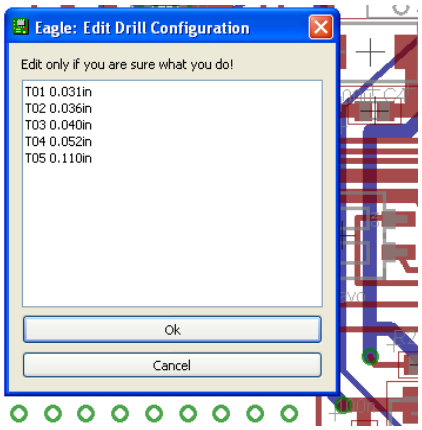


Figura E-21: Selección de los taladros

10. Ejecute de nuevo el CAM Processor; esta vez en Device seleccione EXCELLON_RACK, tras ello le aparecerá una nueva casilla, Rack, al pulsar sobre la casilla tendrá la opción de escoger el archivo anteriormente generado, *locomocion.drl*. Una vez seleccionado, pulse abrir.

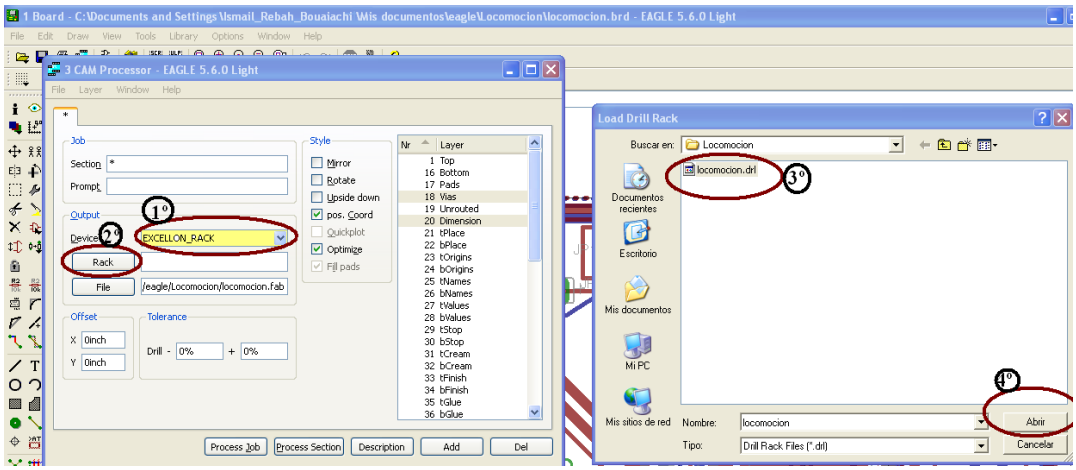


Figura E-22: Generación del archivo locomocion.drl

11. Ahora debe sustituir en File, donde pone locomocion.fab, por locomocion.drd; además, a la derecha debe deseleccionar la capa Vias y la capa Dimension, y seleccionar las capas Drill y Holes quedando como en la Figura E-23.

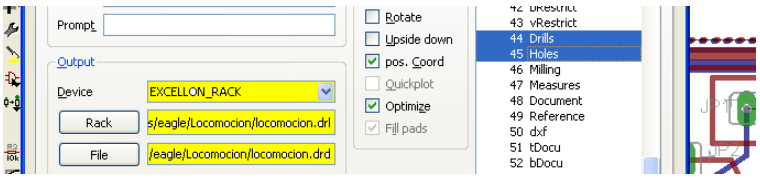


Figura E-23: Uso del archivo locomocion.drl

Una vez hecho lo anterior es muy importante que cambie la casilla Device a EXCELLON y aumente la tolerancia a por ejemplo un 0.1%.

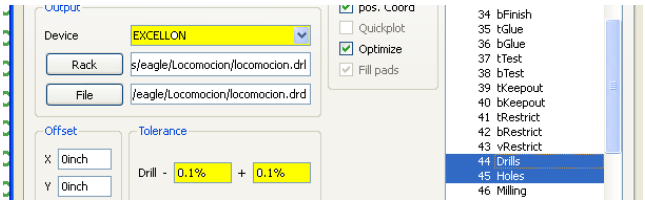


Figura E-24: Generación del archivo locomocion.drd

Al pulsar sobre Process Job, se generará el archivo *locomocion.drd*.

Una vez hecho todo lo anterior ya ha conseguido generar los cuatro archivos Gerber necesarios. Para construir el PCB ha de combinar esos cuatro archivos en un único archivo, locomocion.lmd. Ese es precisamente el objetivo del programa CircuitCAM. Ha de seguir pues los siguientes pasos.

1. Abra el programa pulsando sobre CircuitCAM 6.1.
2. Pulse sobre el icono Import, seleccione los cuatro archivos Gerber y pulse abrir, como se muestra en la Figura E-25.

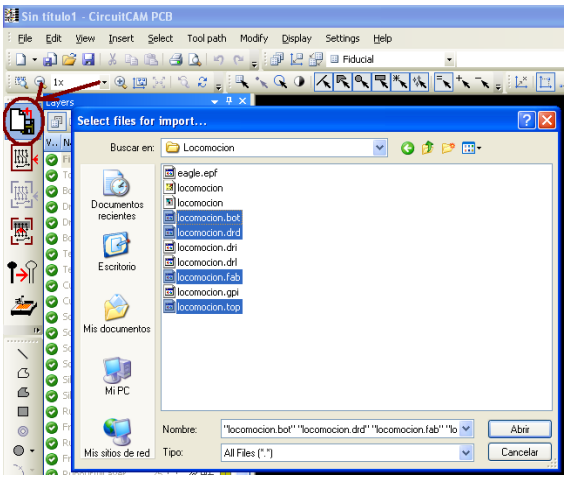


Figura E-25: Importación de Gerbers

3. El siguiente paso es asociar los archivos Gerber a las capas que representan. De este modo, locomocion.top a TopLayer, locomocion.bot a BottomLayer, locomocion.fab a BoardOutline y locomocion.drd a DrillPlated o DrillUnplated, como se puede ver en la Figura E-26. Finalmente pulse Ok.

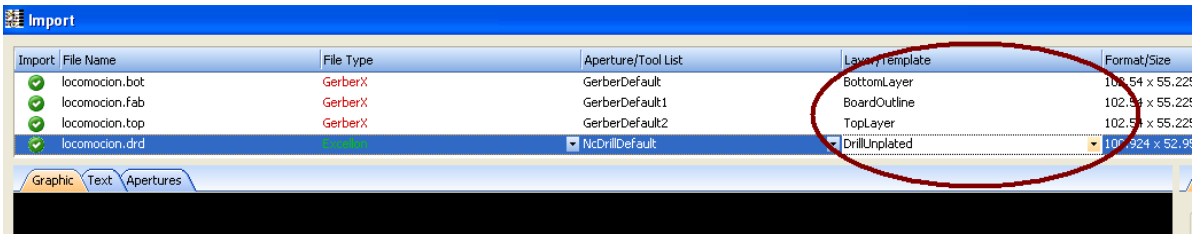


Figura E-26: Asignación entre los archivos Gerber y las capas físicas

4. El siguiente paso es generar el contorno del PCB, para ello ha de pulsar sobre el icono Contour Routing y rellenar los recuadros como en la imagen siguiente. Finalmente pulse Run y verá como el contorno del PCB que había en pantalla se vuelve notablemente más grueso.

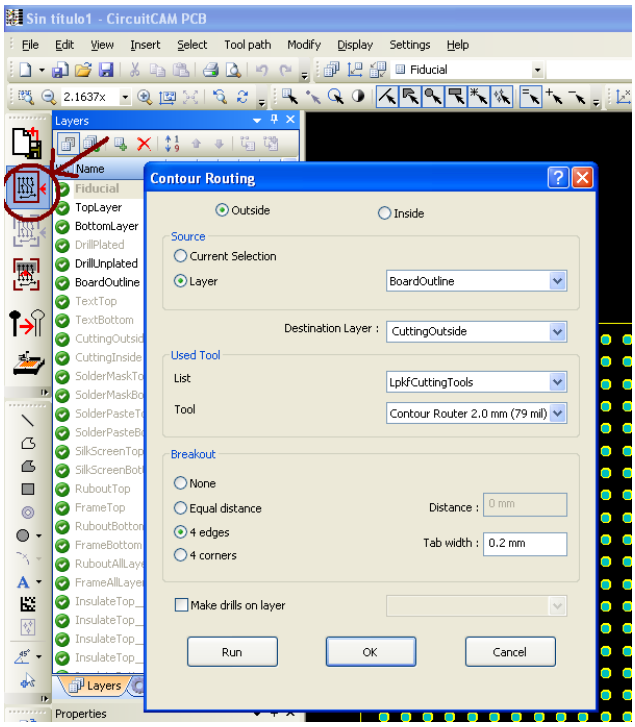


Figura E-27: Generando el contorno del PCB

5. El siguiente paso es elegir las zonas del PCB en las que quiere que el cobre sea eliminado; por defecto sólo se elimina el cobre necesario para dejar las pistas y los pads aislados; pero puede ser muy interesante eliminar más, sobre todo para soldar con mayor facilidad los componentes. Para ello ha de pulsar sobre el icono Rubout All Layers y hacer un recuadro que abarque todo el PCB o las zonas que desea que sean aisladas. Este proceso se aplicará a ambas caras, si quiere que solo se aisle una de ellas debe seleccionar Rubout Bottom o Rubout Top.

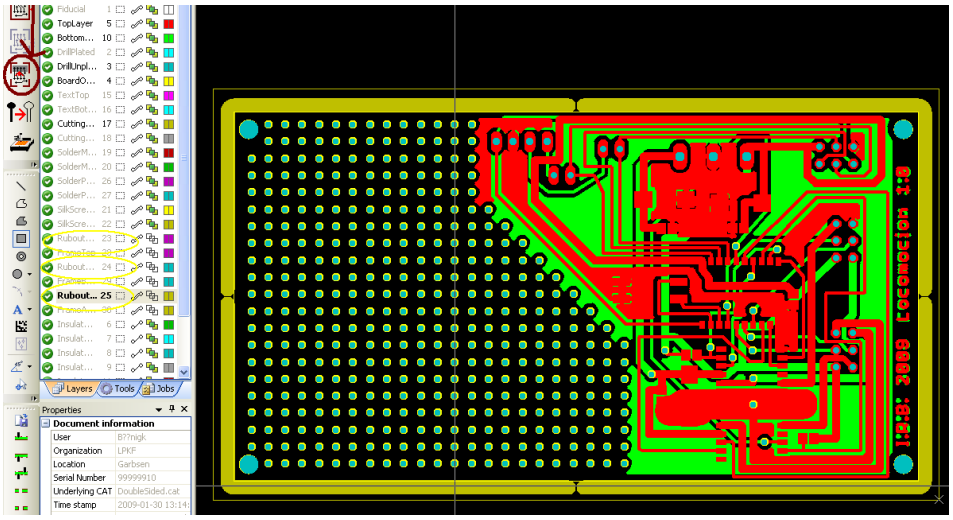


Figura E-28: Selección del cobre que se desea eliminar

6. Una vez seleccionadas las zonas, o hecho el rubout de todo el PCB se valida pulsando sobre el icono Insulate All Layers, tras pulsar el PCB se llenará de pequeños rectángulos azules que indican los lugares que hay que eliminar de cobre.

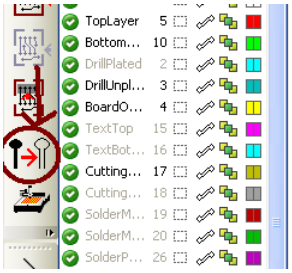


Figura E-29: Eliminación del cobre

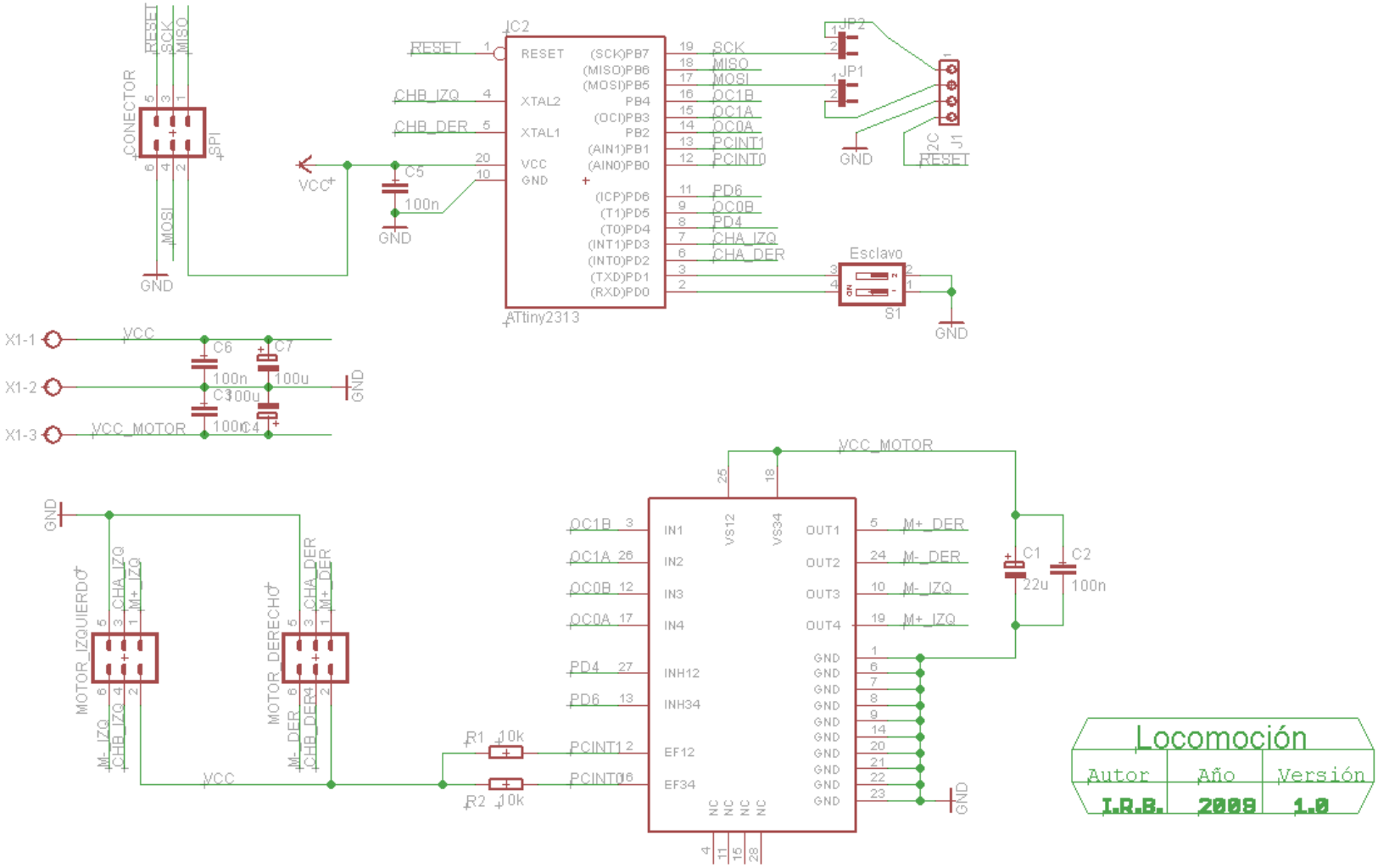
7. El último paso consiste en la generación del fichero locomocion.lmd que contiene toda la información acerca de la construcción del PCB. Así pues, debe pulsar sobre el icono Export LkpfCircuitBoardPlotter. Tras pulsar guarde con el nombre *locomocion.cam*. El programa le informará de que también se ha guardado en la misma ubicación *locomocion.lmd*.

El fichero *locomocion.lmd* es el que tiene que abrir desde el programa BoardMaster para finalmente conseguir construir el módulo de locomoción.(28)

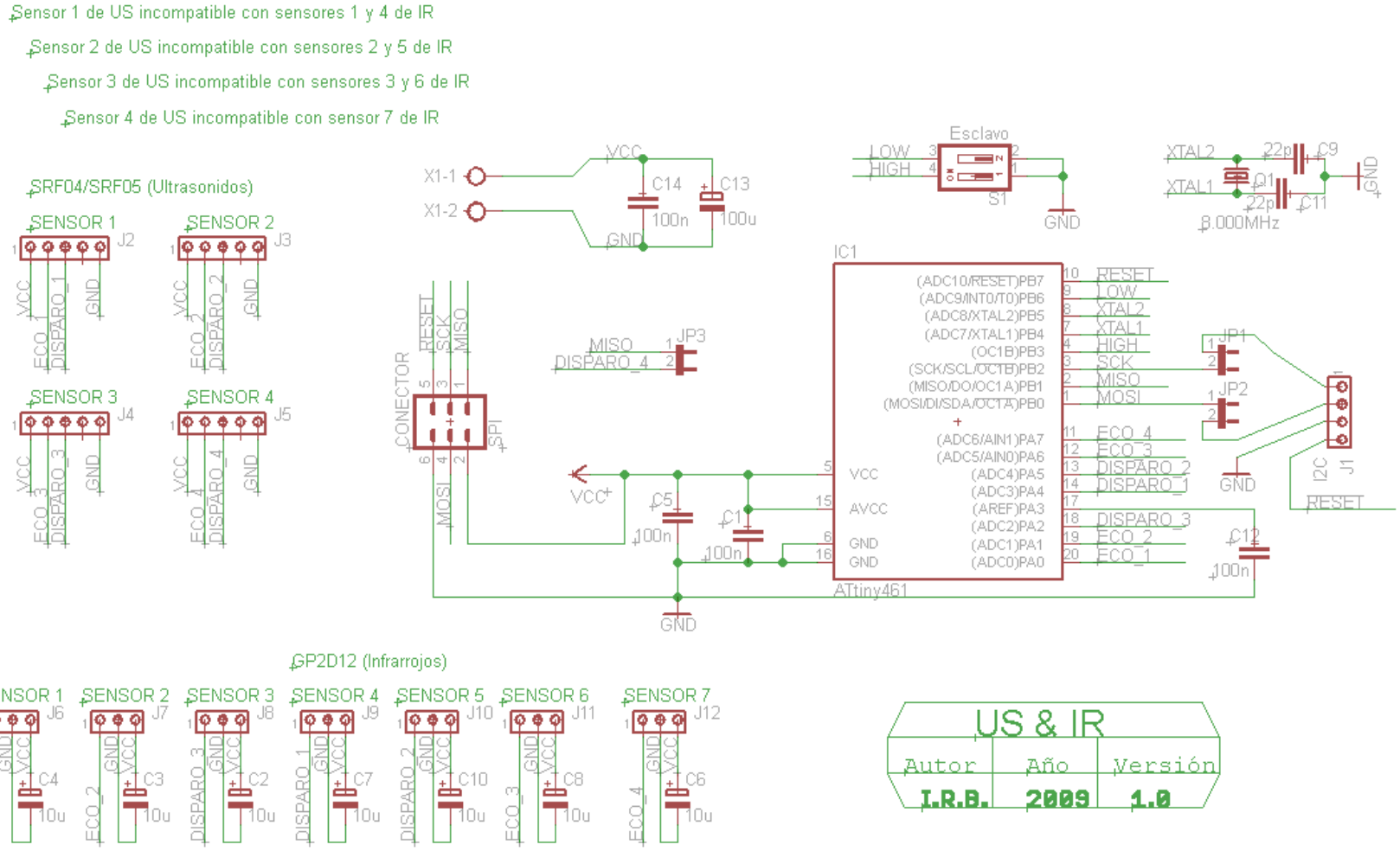
Nota: ***El proceso detallado para la construcción de los módulos es el seguido durante la realización de este PFC, no obstante es posible su construcción con el uso de otras máquinas fresadoras o incluso por métodos caseros***

G Esquemáticos y layouts de los módulos

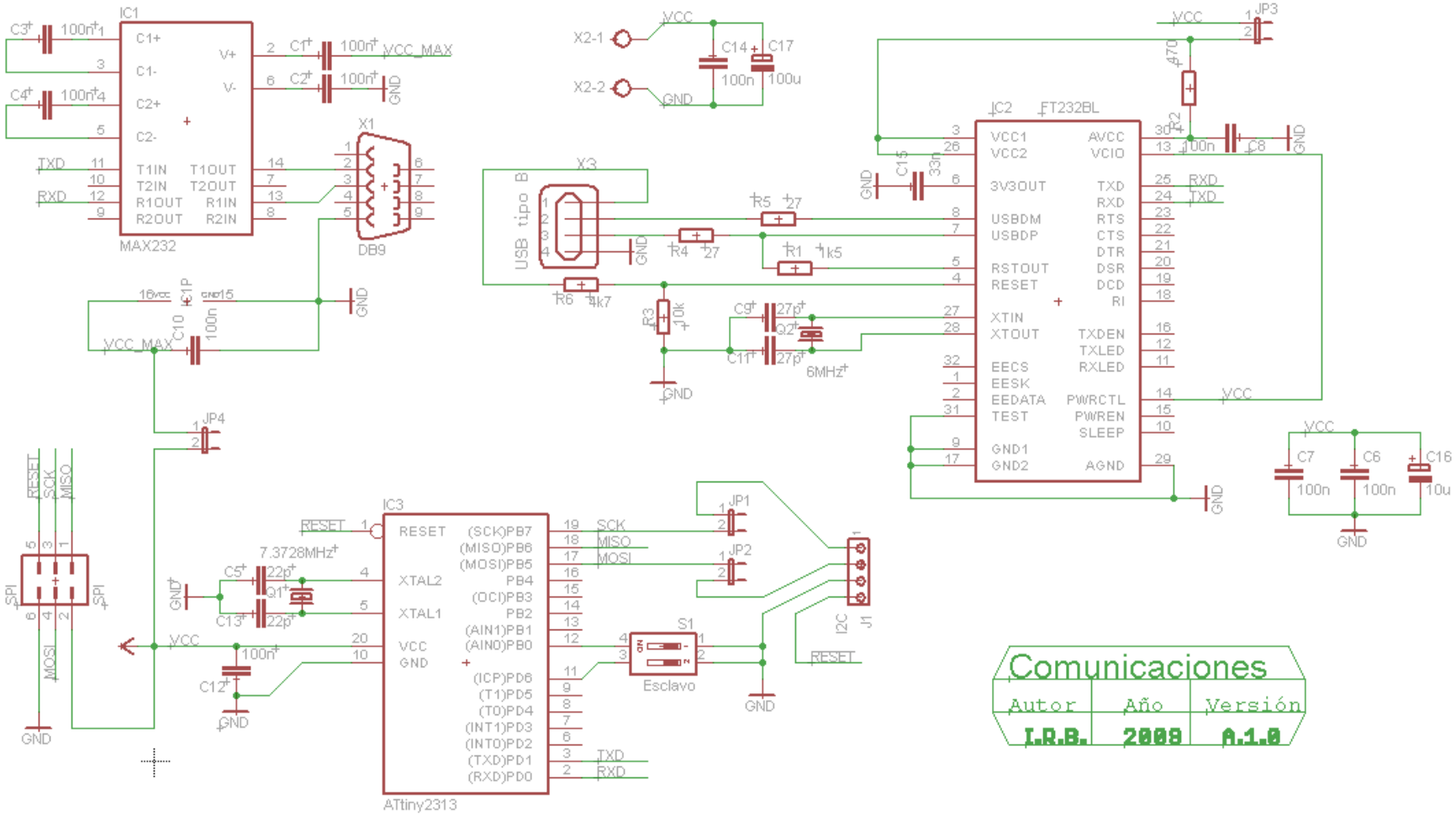
G.1 Esquemático del módulo de locomoción



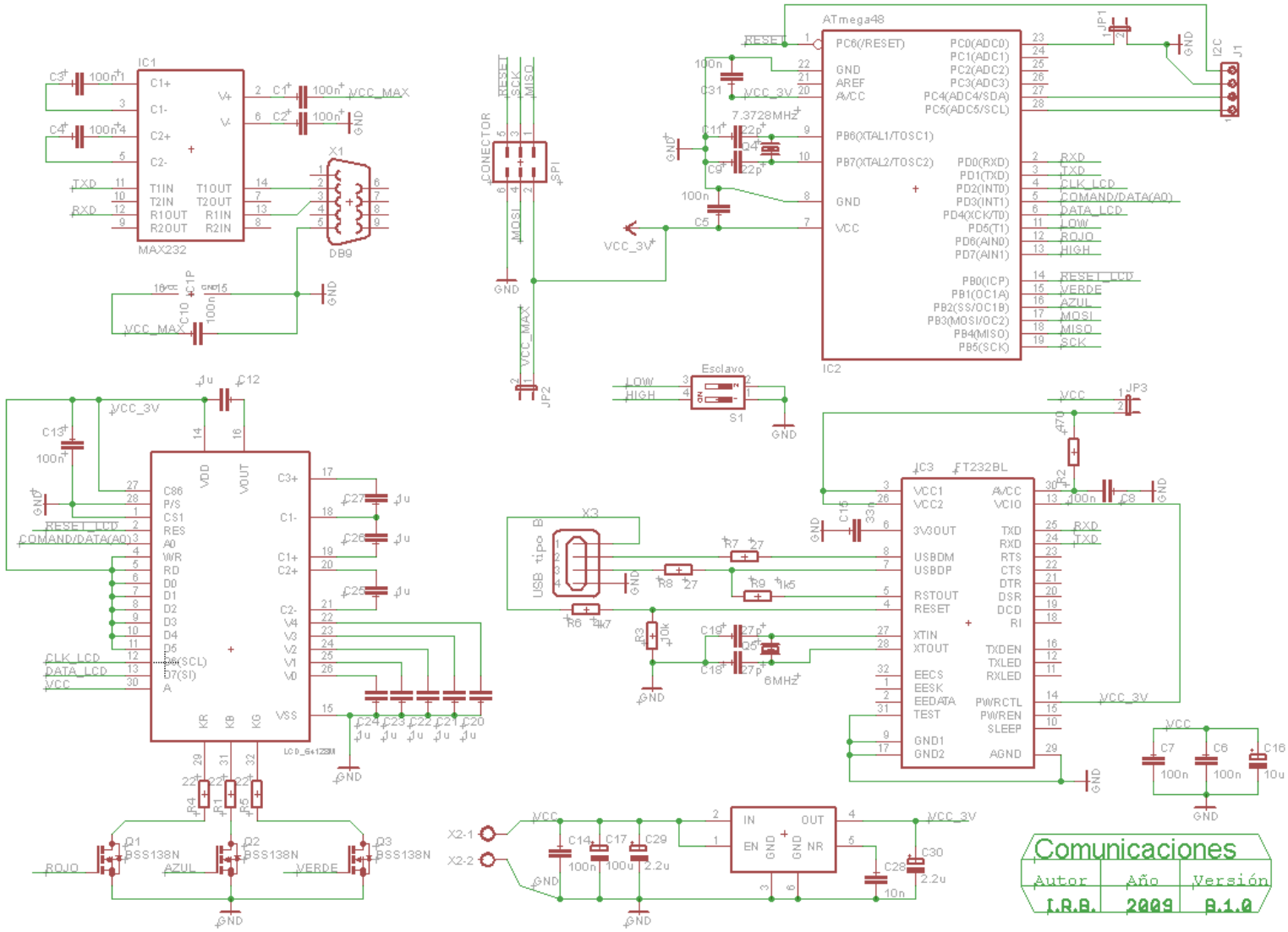
G.2 Esquemático del módulo US & IR



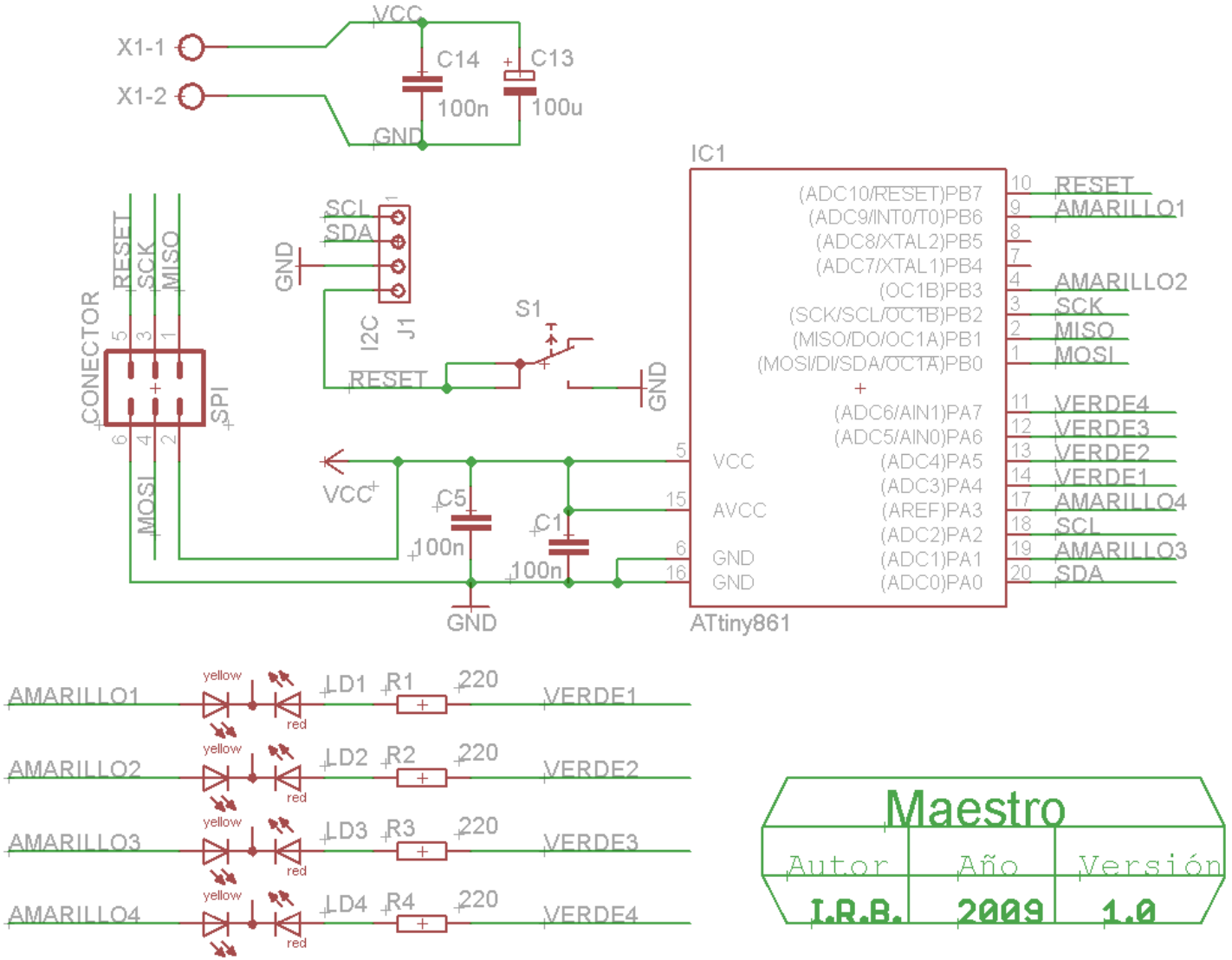
G.3 Esquemático del módulo de comunicaciones modelo A



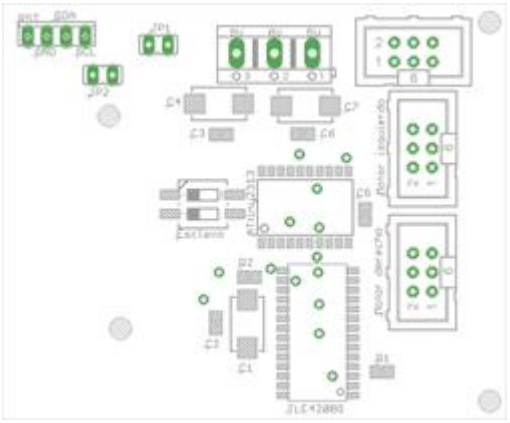
G.4 Esquemático del módulo de comunicaciones modelo B



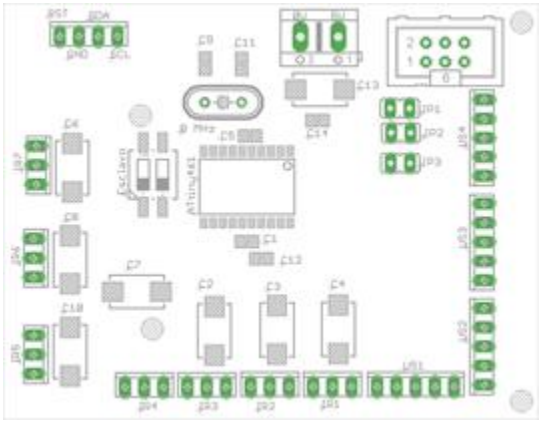
G.5 Esquemático del módulo maestro



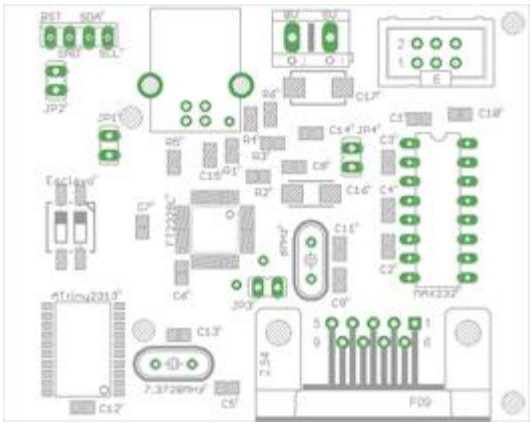
G.6 Layouts “components” de los módulos a tamaño real



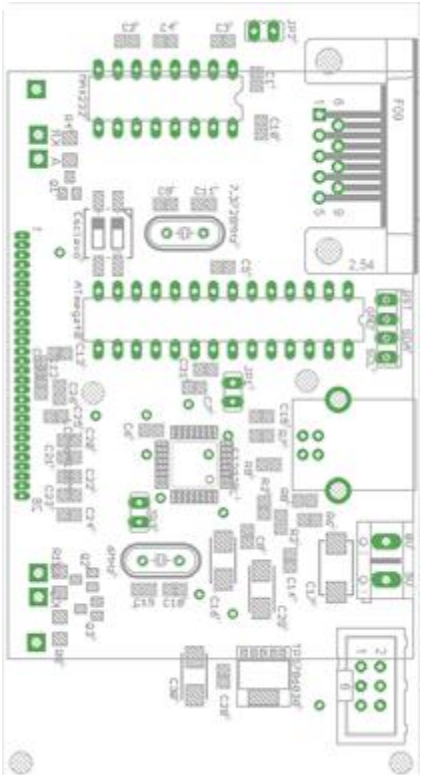
Módulo de Locomoción



Módulo US&IR

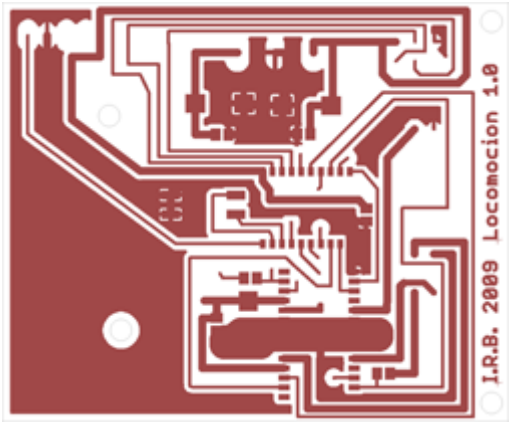


Módulo de comunicaciones A

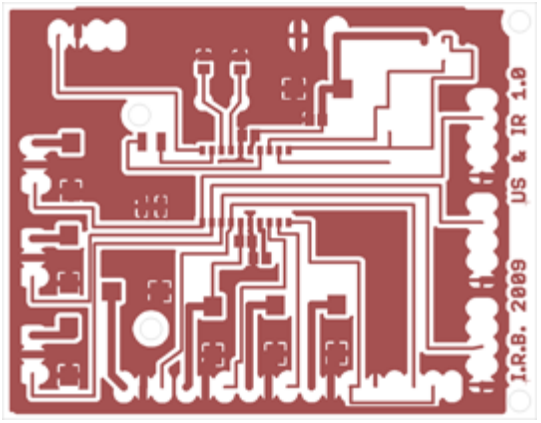


Módulo de comunicacionesB

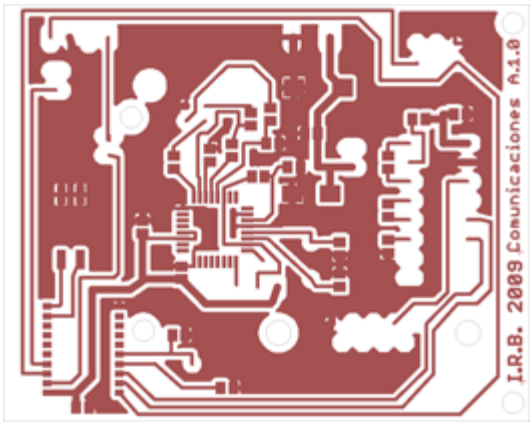
G.7 Layouts “top” de los módulos a tamaño real



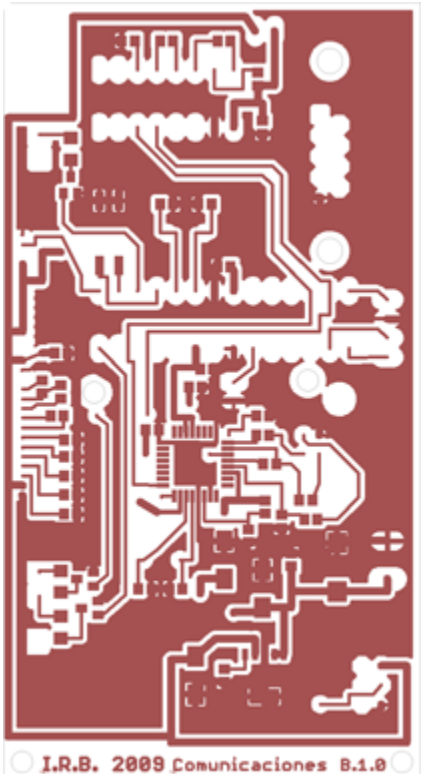
Módulo de Locomoción



Módulo US&IR

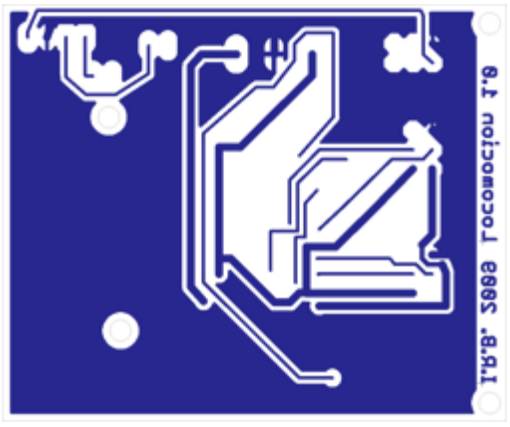


Módulo de comunicaciones A

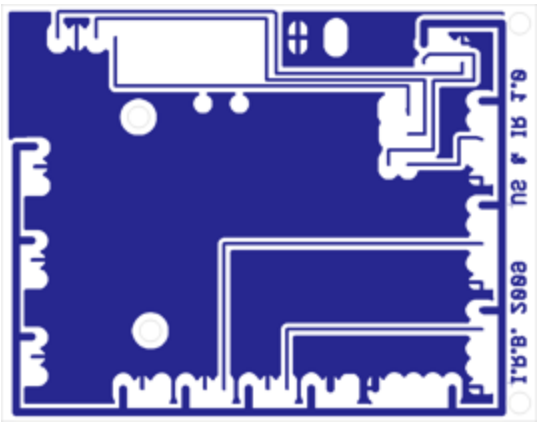


Módulo de comunicacionesB

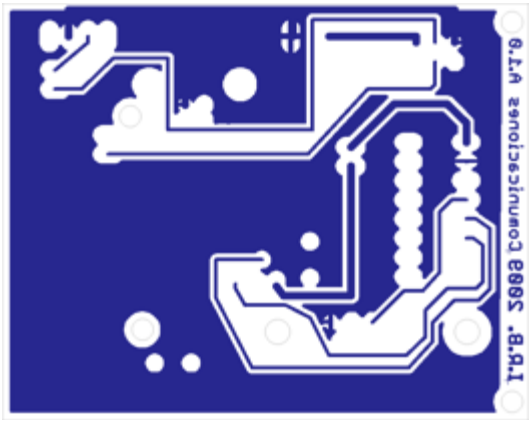
G.8 Layouts “bottom” de los módulos a tamaño real



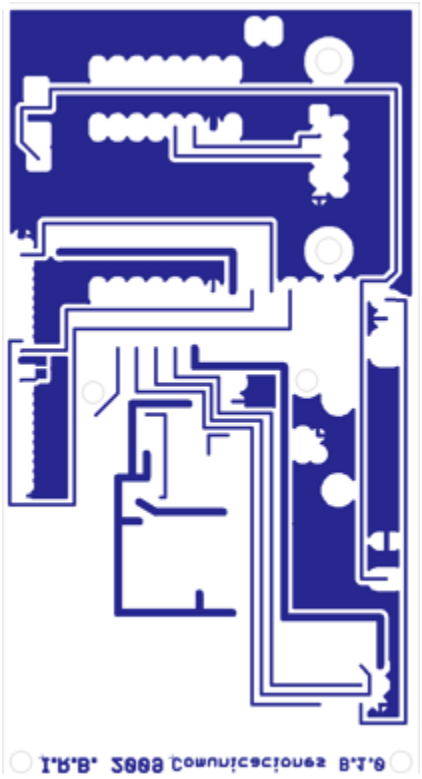
Módulo de Locomoción



Módulo US&IR



Módulo de comunicaciones A



Módulo de comunicacionesB

PRESUPUESTO

1) **Ejecución Material**
   - Compra de ordenador personal (Software incluido) ..... 800,00 €
   - Compra de sensores y motores ..... 200,00 €
   - Coste de los tres módulos esclavo (componentes + construcción) ..... 324,18 €
   - Coste del módulo maestro (componentes) ..... 15,00 €
   - Material de oficina ..... 70,00 €
   - Material de laboratorio (analizador lógico, equipo de soldadura, etc.) ... 869,00 €
   - Total de ejecución material ..... 2318,48 €
2) **Gastos generales**
   - 16 % sobre Ejecución Material ..... 370,96 €
3) **Beneficio Industrial**
   - 6 % sobre Ejecución Material ..... 139,11 €
4) **Honorarios Proyecto**
   - 1280 horas a 15 € / hora ..... 19200 €
5) **Material fungible**
   - Gastos de impresión ..... 240 €
   - Encuadernación ..... 20 €
6) **Subtotal del presupuesto**
   - Subtotal Presupuesto ..... 22288,55 €
7) **I.V.A. aplicable**
   - 16% Subtotal Presupuesto..... 3566,17 €
8) **Total presupuesto**
   - Total Presupuesto ..... 25854,72€

Madrid, Diciembre de 2009

El Ingeniero Jefe de Proyecto

Fdo.: Ismail Rebah Bouaiachi

Ingeniero Superior de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de una plataforma genérica para desarrollo con robots móviles. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos

aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.