

**UNIVERSIDAD DE ALCALÁ**

**Escuela Politécnica Superior**

**Ingeniería Electrónica**



**Trabajo Fin de Carrera**

**Manual de referencia para el desarrollo de robots de  
Eurobot**

Javier Baliñas Santos

2016



**UNIVERSIDAD DE ALCALÁ**  
**ESCUELA POLITÉCNICA SUPERIOR**

**Ingeniería Electrónica**

**Trabajo Fin de Carrera**

**Manual de referencia para el desarrollo de robots de Eurobot**

Autor: Javier Baliñas Santos

Director: Julio Pastor Mendoza

**Tribunal:**

**Presidente:** Luis Miguel Bergasa Pascual

**Vocal 1º:** Luciano Boquete Vázquez

**Vocal 2º:** Julio Pastor Mendoza

Calificación: .....

Fecha: .....



**A mis padres. Maritere y Julio**



# Agradecimientos

*Más vale un minuto de ilusión que mil horas de razonamiento.*

Roberto Barra-Chicote

Este trabajo y sobre todo lo que expone es fruto de muchas horas de trabajo y esfuerzo de muchas personas. En lo que me atañe, me gustaría agradecer a todas las personas que me han acompañado, apoyado y que han sido parte de cada uno de los robots en los que he estado implicado.

Agradecer a Julio la iniciativa de crear allá por el año 2003 un equipo de Eurobot. Igualmente agradecerle el apoyo, ayuda e implicación en los equipos de Eurobot que se formaron desde entonces. También me gustaría aprovechar para felicitarle por el esfuerzo, la constancia y la tenacidad en el promover las competiciones de robótica desde la Universidad de Alcalá. Una de ellas me marcó el camino y estoy seguro que el de mucha más gente.

Es curioso, en mi cabeza los años tienen nombre de robot. Y cada robot lo asocio con la gente que formábamos el equipo que lo construimos. Mi agradecimiento a todos y cada uno por los buenos y malos momentos vividos, entre todos, los malos se soportaron mejor y lo buenos fueron más grandiosos. Especialmente, mi agradecimiento a aquellos que sentí más cerca y con los que compartí más momentos: Diego Alonso, Kike, Jose, Saleta, Oscar, Marcelo, Sergio y Nilo.

Agradecer también a mis últimos compañeros del equipo Eurobotics Engineering: Carlos, Rubén, Silvia y Javi. Formamos un buen equipo y vivimos grandes aventuras, especialmente el último año. Las ganas e ilusión de todos vosotros me ayudaron a seguir en momentos de flaqueza. Silvia, Javi, agradecí mucho tener un apoyo y programar mano a mano con vosotros. Sobre todo en el último año, gracias Javi por tu espíritu de equipo, tus ganas, iniciativa y constantes mensajes de ánimo al equipo.

Me gustaría hacer una mención muy especial a Diego y Mario. Sin ellos, no hubiera sido posible hacer todo lo que hemos hecho y llegar hasta donde hemos llegado en todos estos años. Gracias a Diego por tus ganas, esfuerzo e ímpetu sincero que nos hizo llegar alto, los mismos que ahora te hacen recorrer montañas por el mundo. Gracias a Mario que siempre, siempre, siempre, has estado apoyando e implicándote en todos los robots, sacando tiempo de donde no lo tenías. Gracias amigo.

Un recordatorio especial a Guillermo, gracias por tu apoyo incondicional durante todos estos años. Sólo tu capacidad de asombro por los robots que hemos hecho, hacía que mereciese la pena todo el esfuerzo. Se te recordará siempre.

Por último, agradecer a las personas que directa o indirectamente han sufrido mis ausencias por mi dedicación a los robots, mi familia y amigos. Especialmente agradecer a Jana su apoyo, paciencia y esfuerzo por comprenderme.



# Resumen

Este trabajo de fin de carrera trata sobre el desarrollo de robots para la competición de robots Eurobot [1]. Pretende ser un manual de referencia para todo aquel que quiera construir un robot para participar en Eurobot. El trabajo está basado en la experiencia adquirida en el desarrollo de este tipo de robots entre los años 2003 y 2015. Especialmente en el periodo entre 2010 y 2015.

Eurobot es una competición internacional cuyas reglas y temática el juego cambian cada año. El libro describe la mecánica, hardware y software relativos al desarrollo e implementación de las diferentes funcionalidades de un robot de Eurobot. El desarrollo e implementación de éstas, se han dividido en dos partes principales: la *plataforma robótica base* y los *sistemas mecánicos* de manipulación de elementos del juego.

La plataforma robótica se basa en un *sistema de tracción diferencial* que implementa un *posicionamiento por odometría* y un *control de posición polar*. La plataforma, además, utiliza sensores para detectar obstáculos y un *sistema de balizas* para la medir la posición de los robots oponentes, a partir del cual, se implementa la *evitación de obstáculos*.

Mediante el estudio del modelo dinámico de la plataforma robótica, se determinan los límites de aceleración y velocidad, a partir de los cuales, dimensionar y especificar los elementos mecánicos y estructura de la misma.

El desarrollo *hardware* implementa una *arquitectura* que permite ser reutilizada en el desarrollo de nuevos robots de Eurobot. Se trata de un sistema embebido basado en *microcontroladores dsPIC*, que reparte sus recursos entre la implementación de la plataforma robótica y los sistemas mecánicos.

El desarrollo *software* también está basado en una *arquitectura* pensada para ser reutilizada. Ésta, se organiza en capas y/o módulos funcionales con diferentes niveles de abstracción. El desarrollo software, incluye además, un *simulador de los robots y del campo de juego*. El entorno de simulación permite visualizar los movimientos de los robots sobre un campo de juego virtual y simular robots oponentes.

**Palabras clave:** robot, eurobot, control, posicionamiento, odometría, electrónica, mecánica, programación, microcontrolador, simulador, sistemas embebidos.



# Abstract

This End of Degree Assignment is about the development of robots for the Eurobot contest [1]. It aims to be a reference manual for whom wants to make a robot for participating on Eurobot contest. It's based on the gathered experience on the development of this kind of robots between the years 2003 and 2015. Specially on the period from 2010 to 2015.

Eurobot is an international contest whose theme and rules change every year. The book describes the mechanics, hardware and software related to the implementation of the different funicionalities of a robot. The development and implementation of these functionalities has divided in two main parts: the *robotic base platform* and the *mechanical systems* for the handling of game elements.

The robotic platform is based on a *differential drive system* which implements a *position dead reckoning by odometry* and a *polar position control*. The platform uses also sensors for the obstacles detection and a *beacon system* in order to measure the opponents robots position. Using the information of the beacon system the robot implemnts an *obstacle avoidance algorithm*.

The acceleration and speed limits of the robotic platform have been determined via the study of its dinamic model. These limits are used to specify the mechanical elements and the structure of the robotic platform.

The *hardware* development implements an *architecture* which can be reused on others developments of Eurobot robots. It's a embedded system based on *dsPIC microcontrollers*. The system distributes its resources between the implementations of the robotic platform and the mechanical systems.

The developed *software* is also based on an *architecture* aimed to be reused. This architecture is organized in layers and/or functional modules with different abstraction levels.

The software development includes also a *simulator of the robots and of the game field*. This simulator is implemented by the compilation of the robot source code on a GNU/Linux host and the emulation of the hardware and mechanical elements of the robot. The simulation enviroment allow to visualize the robot movements on a virtual game field and simulate opponents robots.

**Keywords:** robot, eurobot, control, positioning, odometry, hardware, mechanics, software, simulator, embedded systems.



# Resumen extendido

Este trabajo de fin de carrera trata sobre el desarrollo de robots para la competición de robots Eurobot [1]. Pretende ser un manual de referencia para todo aquel que quiera construir un robot para participar en Eurobot. Todo lo que se expone en este libro se sustenta en la experiencia adquirida en el desarrollo de robots participantes en Eurobot entre los años 2003 y 2015. Especialmente en el periodo entre 2010 y 2015.

El libro trata de describir el desarrollo de este tipo de robots utilizando patrones de implementación y esquemas de funcionamiento que puedan ser utilizados como punto de partida por aquellos no iniciados en este *arte*.

Eurobot es un concurso internacional de robótica creado en 1998 a partir de la Copa de Francia de la robótica [2] (Coupe de France de robotique). Las reglas y la temática de Eurobot cambian cada año, sin embargo, la base es siempre la misma: tamaño de los robots, dimensiones del campo de juego y duración de los partidos.

Los robots que se presentan en este proyecto han sido desarrollados por el equipo *Eurobotics Engineering* [3], un equipo participante en Eurobot, perteneciente a la Asociación de Robótica de Coslada (ARC), y que tuvo su origen en antiguos equipos de la Universidad de Alcalá (UAH), creados por el Departamento de Electrónica de la UAH.

Un robot de Eurobot ha de ser completamente autónomo y ser capaz de:

- Desplazarse hasta las diferentes zonas del campo, donde se sitúan los elementos del juego o donde se almacenan.
- Evitar chocar con elementos del campo de juego y con otros robots oponentes durante los desplazamientos entre zonas del campo.
- Resolver la temática del juego, lo cual implica el manejo de los elementos o piezas propias del juego.

En base a esto, el desarrollo de un robot de Eurobot se ha dividido en las siguientes partes funcionales principales:

1. Desplazamiento y localización.
2. Evitación de obstáculos.
3. Comunicación entre robots (en caso de jugar con 2 robots).
4. Manipulación de elementos de juego.
5. Estrategia de juego.

Desde la experiencia, se ha descrito y sintetizado lo que implica el desarrollo de cada una de estas partes, aportando ejemplos y casos reales. También, se reflexiona sobre los objetivos que llevan a desarrollar este tipo de robots y a participar en Eurobot, desde el punto de vista del equipo y sus integrantes. Igualmente, se tratan las diferentes fases de las que se compone el desarrollo de un robot de Eurobot.

Se presenta un desarrollo formado dos partes fundamentales: la **plataforma robótica base** y los **sistemas mecánicos de manipulación de elementos de juego**. La plataforma robótica está formada por los tres primeros bloques funcionales mencionados anteriormente, cuyo desarrollo, es independientemente de la temática de la prueba de Eurobot. De esta forma, la arquitectura mecánica, hardware (HW) y software (SW) de la plataforma robótica base, puede ser reutilizado y/o adaptada en futuros desarrollos de robots de Eurobot (o similares). En cambio, el resto de las partes, como los sistemas mecánicos y la estrategia, dependen de la temática de la prueba e implica un desarrollo específico.

La función de **desplazamiento y localización** de la plataforma robótica, se implementa a partir de un *control de posición polar* y un *sistema de posicionamiento por odometría*, a partir del los cuales, se gestiona la *generación de trayectorias*.

Mecánicamente, la plataforma robótica está basada en un sistema de tracción diferencial. Éste, se compone de un bloque motor formado por dos motores unidos mediante una reductora mecánica a dos ruedas motrices (ver figura 2). Además, sobre el eje motriz, en sus extremos, el robot tiene dos ruedas libres conectadas a sensores tipo encoder que se utilizan para implementar el sistema de posicionamiento por odometría y el control de posición del robot.

El estudio del modelo dinámico de la plataforma robótica, permite dimensionar y especificar los elementos que componen el sistema de tracción diferencial (la reductora y los motores de tracción), y las características físicas de la estructura del robot (puntos de apoyo y centro de gravedad).

El control de posición polar está formado por sendos controladores de posición en distancia y ángulo. Ambos controladores, implementan un perfil de velocidad trapezoidal, cuya velocidad y aceleración, vienen determinados por el modelo dinámico de la plataforma robótica. El posicionamiento por odometría se obtiene mediante la aproximación lineal del modelo del sistema de encoders libres de la plataforma robótica. La medida y corrección de los errores sistemáticos del sistema, permite reducir el error acumulativo de odometría.

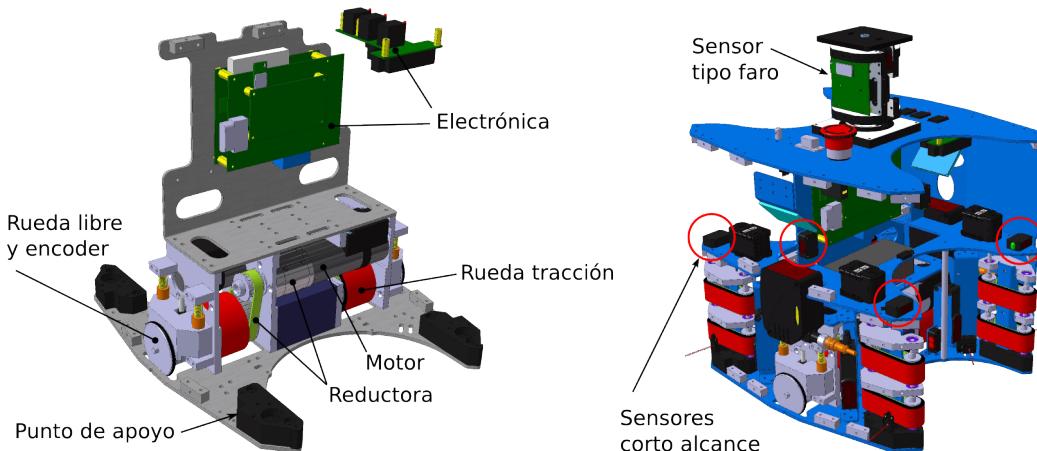


Figura 2: Elementos de la plataforma robótica base

La **evitación de obstáculos** se realiza a partir de sensores de distancia situados en las caras del robot (ver figura 2), y a partir de un *sistema de balizamiento* que hace uso de los soportes y espacios destinados a las balizas de la competición Eurobot. Éste sistema, está compuesto por un sensor tipo

faro situado en el robot, y por balizas reflectantes situadas en el oponente (ver figura 2). Además, está específicamente desarrollado para medir la distancia y ángulo de los oponentes, relativos a la posición de la plataforma robótica.

El sensor de la baliza tipo faro gira sobre el plano  $xy$  y emite una luz que es reflejada por balizas reflectantes, cuando el sensor se encuentra enfrentado con ellas. La medida de distancia se basa en el principio por el cual, un objeto cercano se ve más grande que uno lejano. De forma que, a una distancia cercana le corresponde un ángulo relativo de detección  $\alpha_d$ , mayor que el de una distancia más lejana. Por otro lado, la medida de ángulo, se obtiene a partir del ángulo durante el cual se detecta la baliza reflectante.

El desarrollo HW presentado, implementa una **arquitectura HW** que contempla la implementación, tanto de la plataforma robótica, como de los sistemas mecánicos del robot. La arquitectura implementa un sistema embebido basado en microcontroladores dsPIC, que tiene sus recursos repartidos entre la implementación de la plataforma robótica y de los sistemas mecánicos. Debido a que los recursos destinados a los sistemas mecánicos dependen de la temática de la prueba, la arquitectura tiene una interfaz I2C destinada expansión de recursos de entrada/salida (E/S) y a la implementación de un procesamiento distribuido.

Por otro lado, la arquitectura HW tiene una interfaz de depuración RS-232 a través de la cual poder controlar, monitorizar y probar el hardware y el software desarrollado. Esta interfaz, forma una cadena serie con el resto de microcontroladores que permite acceder a cualquiera de ellos mediante un *bypass* de los datos. Además, la arquitectura cuenta con una interfaz inalámbrica Bluetooth, mediante la cual se implementa la **comunicación con la baliza tipo faro y el robot secundario**. Dependiendo de las características del robot, esta arquitectura HW ha sido implementada en una o varias tarjetas electrónicas, como se muestra en varios ejemplos de robots desarrollados.

Se describe un desarrollo SW que basado en una **arquitectura SW**, cuya estructura, contempla la implementación funcional de la plataforma robótica, los sistemas mecánicos y las funcionalidades propias de la temática y estrategia de juego del robot de Eurobot. Así mismo, se tratan ejemplos de robots desarrollados que implementan esta arquitectura SW.

La implementación SW de la plataforma robótica se ha realizado a partir de las librerías Aversive [4], desarrolladas por el equipo de Eurobot Microb Technology [5]. Las librerías Aversive constituyen un marco de trabajo para el desarrollo de sistemas basados en microcontroladores AVR de Atmel. A partir de estas librerías, se crearon las librerías Aversive4dspic, una migración de las librerías Aversive a los microcontroladores dsPIC de Microchip, utilizados por el HW desarrollado.

La arquitectura SW propuesta se organiza en capas o módulos funcionales, de forma que aquellos de nivel superior implementan funciones más abstractas que los inferiores. El módulo *plataforma robótica* incluye las funcionalidades de desplazamiento, localización y evitación de obstáculos. Al mismo nivel, se encuentra el módulo *sistemas mecánicos*, encargado de controlar los mecanismos utilizados por el robot para manipular los elementos de juego. Ambos módulos, se integran y se sincronizan en la capa de *temática de juego*, que implementa las funcionalidades propias de la prueba de Eurobot. El nivel más abstracto lo implementa la capa *estrategia de juego*, la cual decide cuándo y qué acciones del juego realizar en cada momento durante un partido de Eurobot.

Por último, el desarrollo SW incluye además un **simulador de los robots y del campo de juego**. Éste simulador, se implementa a partir de la compilación del código fuente de los robots en un *host* GNU/Linux, y de la emulación del HW y elementos mecánicos del robot. El entorno de simulación, permite visualizar los movimientos de los robots sobre un campo de juego virtual y simular robots oponentes.



# Índice general

<b>Resumen</b>	<b>ix</b>
<b>Abstract</b>	<b>xi</b>
<b>Resumen extendido</b>	<b>xiii</b>
<b>Índice general</b>	<b>xvii</b>
<b>Índice de figuras</b>	<b>xxi</b>
<b>Índice de tablas</b>	<b>xxiii</b>
<b>1 Introducción al proyecto</b>	<b>1</b>
1.1 Presentación . . . . .	1
1.2 La competición Eurobot . . . . .	1
1.3 El equipo <i>Eurobotics Engineering</i> , historia y antecedentes . . . . .	2
1.4 La robótica educativa . . . . .	5
1.5 Aprendiendo de gigantes. . . . .	7
1.6 Motivación y objetivos . . . . .	8
1.7 Organización del libro . . . . .	9
<b>2 El robot de Eurobot</b>	<b>11</b>
2.1 Especificaciones del robot de Eurobot . . . . .	11
2.2 Elementos del juego y tareas de un robot de Eurobot . . . . .	13
2.3 Partes principales en el desarrollo de un robot de Eurobot . . . . .	15
2.3.1 Estrategia de juego . . . . .	16
2.3.2 Desplazamiento y localización . . . . .	16
2.3.3 Evitación de obstáculos . . . . .	17
2.3.4 Comunicación entre robots . . . . .	20
2.3.5 Manipulación de elementos de juego . . . . .	22
2.4 Objetivos del desarrollo de un robot de Eurobot . . . . .	22
2.5 Desarrollo del robot de Eurobot . . . . .	25

2.5.1	Fabricación del campo de juego . . . . .	26
2.5.2	Diseño de la estrategia de juego . . . . .	26
2.5.3	Realización de prototipos mecánicos y pruebas de concepto . . . . .	30
2.5.4	Diseño y fabricación de la mecánica y de la electrónica . . . . .	31
2.5.5	Montaje del robot y cableado . . . . .	32
2.5.6	Implementación software de las funcionalidades del robot . . . . .	32
2.5.7	Implementación software de la estrategia de juego . . . . .	33
2.5.8	Puesta en marcha y verificación . . . . .	34
<b>3</b>	<b>Plataforma robótica base</b>	<b>35</b>
3.1	Descripción general de la plataforma robótica . . . . .	35
3.2	Sistema de tracción diferencial . . . . .	37
3.3	Sistema de posicionamiento y control de posición . . . . .	38
3.4	Estudio de la dinámica de robots de tracción diferencial . . . . .	39
3.4.1	Dinámica básica en desplazamientos lineales sobre un plano horizontal . . . . .	39
3.4.2	Coeficiente de adherencia de las ruedas y su cálculo experimental . . . . .	40
3.4.3	Determinación de la aceleración máxima en las fases de aceleración y frenado . . . . .	42
3.4.3.1	Robots con ruedas atrás . . . . .	43
3.4.3.2	Robots con ruedas en el medio . . . . .	45
3.4.3.3	Medida del centro de gravedad del robot . . . . .	47
3.4.4	Perfil de velocidad trapezoidal . . . . .	47
3.4.5	Dimensionamiento de la reductora . . . . .	48
3.4.6	Calculo de la velocidad máxima . . . . .	52
3.5	Control en posición de tipo polar . . . . .	56
3.5.1	Implementación del perfil de velocidad trapezoidal . . . . .	60
3.5.2	Sistema de unidades del robot . . . . .	60
3.5.3	Ajuste del controlador de posición . . . . .	62
3.6	Posicionamiento mediante odometría . . . . .	67
3.6.1	Calculo de la posición del robot . . . . .	67
3.6.2	Medida y corrección de errores sistemáticos . . . . .	69
3.6.2.1	Test del cuadrado bidireccional . . . . .	71
3.6.2.2	Método utilizado . . . . .	73
<b>4</b>	<b>Sistema de balizas tipo faro</b>	<b>77</b>
4.1	Principio de funcionamiento . . . . .	77
4.1.1	Medida de distancias y ángulos . . . . .	78
4.2	Implementación mecánica . . . . .	80
4.2.1	Baliza de cuerpo giratorio . . . . .	80

4.2.2	Baliza de espejo giratorio . . . . .	81
4.3	Implementación hardware y software . . . . .	82
4.3.1	Control de velocidad de giro . . . . .	83
4.3.2	Procesamiento de medidas y calculo de la posición . . . . .	83
4.3.3	Interfaz con la plataforma base . . . . .	85
4.4	Ajustes, modelado y calibración . . . . .	85
4.4.1	Ajustes del rango de medida . . . . .	85
4.4.1.1	Ajuste de la distancia mínima de detección . . . . .	85
4.4.1.2	Ajuste del rango de medida de distancia . . . . .	86
4.4.2	Modelado y calibración de la medida de distancia . . . . .	86
<b>5</b>	<b>Desarrollo hardware</b>	<b>89</b>
5.1	Sobre sensores y actuadores . . . . .	89
5.2	Arquitectura hardware . . . . .	91
5.2.1	Plataforma robótica base . . . . .	93
5.2.2	Sistemas mecánicos . . . . .	94
5.2.3	Esquema de alimentación . . . . .	96
5.3	Implementación HW del robot principal . . . . .	98
5.3.1	Tarjeta principal . . . . .	98
5.3.2	Tarjeta de sensores digitales . . . . .	101
5.3.3	Tarjeta de sistemas de vacío . . . . .	101
5.3.4	Tarjeta de control de alimentación . . . . .	102
5.3.5	Tarjeta de alimentación . . . . .	102
5.4	Implementación HW del robot secundario . . . . .	102
<b>6</b>	<b>Desarrollo software</b>	<b>107</b>
6.1	Librerías Aversive y Aversive4dspic . . . . .	108
6.1.1	Herramientas matemáticas . . . . .	109
6.1.2	Gestión de <i>buffers</i> de datos . . . . .	109
6.1.3	Planificador de eventos . . . . .	109
6.1.4	Medida de tiempo . . . . .	110
6.1.5	Mensajes de depuración . . . . .	110
6.1.6	Interfaz de línea comandos . . . . .	110
6.2	Arquitectura software . . . . .	111
6.3	Implementación funcional . . . . .	113
6.3.1	Plataforma robótica base . . . . .	114
6.3.1.1	Gestión de trayectorias . . . . .	114
6.3.1.2	Control de posición . . . . .	116

6.3.1.3	Odometría . . . . .	116
6.3.1.4	Detección de bloqueos . . . . .	116
6.3.2	Sistemas mecánicos . . . . .	117
6.3.3	Temática de juego . . . . .	120
6.3.4	Estrategia de juego . . . . .	122
6.4	Simulador de robots y campo de juego . . . . .	124
<b>Bibliografía</b>		<b>129</b>
<b>A Planos eléctricos</b>		<b>131</b>
A.1	Tarjeta <i>dspic33_protoboard</i> . . . . .	132
A.2	Tarjeta <i>mainboard_v02</i> . . . . .	140
A.3	Tarjeta <i>mainboard_v03</i> . . . . .	150
A.4	Tarjeta <i>switchboard</i> . . . . .	157
A.5	Tarjeta <i>sensorboard</i> . . . . .	159
A.6	Tarjeta <i>beaconboard</i> . . . . .	162
A.7	Tarjeta <i>vacuumboard</i> . . . . .	165
<b>B Código fuente</b>		<b>167</b>
B.1	Librerías Aversive4dspic . . . . .	167
B.2	Código fuente de robots . . . . .	167

# Índice de figuras

1.1	Plataforma robótica <i>Dummy robot</i> . . . . .	8
2.1	Altura de balizas y sensores . . . . .	12
2.2	Soportes para balizas del campo de juego de Eurobot . . . . .	13
2.3	Partes principales del desarrollo de un robot de Eurobot . . . . .	15
2.4	Estrategia de juego: variables a tener en cuenta en la toma de decisiones . . . . .	16
2.5	Representación cartesiana del campo de Eurobot 2015 y ejemplo de generación de trayectorias	17
2.6	Desplazamiento y localización: diagrama de bloques de las partes a desarrollar . . . . .	18
2.7	Ejemplo de evitación de obstáculos en trayectorias . . . . .	19
2.8	Ejemplo de sistema de medida de la posición del robot oponente basado en balizas . . . . .	20
2.9	Ejemplo de sistema de detección de obstáculos en trayectoria . . . . .	21
2.10	Diagrama de bloques del sistema de comunicación entre robots . . . . .	22
2.11	Mecanismos del sistema mecánico de <i>stands</i> del robot P.Tinto (Eurobot 2015) . . . . .	23
2.12	Diagrama de bloques del las partes a desarrollar en un sistema mecánico . . . . .	24
2.13	Robots Topolino y Trompetero . . . . .	25
2.14	Elementos y campo de juego utilizado para discutir las primeras estrategias de juego . . . . .	27
2.15	Estrategia de los robot P.Tinto y Tirantes en Eurobot 2015 . . . . .	29
2.16	Prototipo recolección maquinas palomitas de Eurobot 2015 . . . . .	31
2.17	Sistema de recolección maquinas palomitas de Eurobot 2015 . . . . .	31
3.1	Plataforma robótica base: partes del sistema de tracción diferencial y del sistema de control de posición . . . . .	36
3.2	Plataforma robótica base: partes del sistema de evitación de obstáculos . . . . .	36
3.6	Modelo dinámico simple de un robot de tracción diferencial . . . . .	40
3.32	Ánalysis de la influencia de errores sistemáticos durante el giro del robot. . . . .	71



# Índice de tablas

1.1	Robots desarrollados para la competición Eurobot (2003-2015) . . . . .	3
3.1	Resultados de la medida del coeficiente de adherencia de las ruedas. La obtención de un valor mayor que 1,0 para el caso del robot Zamorano se debe a la tolerancia de los objetos utilizados en el test del cubo. En ambos casos el valor se puede aproximar a la unidad. . . . .	42
3.2	Ejemplo de aceleraciones máximas en función del centro de gravedad . . . . .	45
3.3	Unidades SI y del sistema robot . . . . .	60
5.1	Tipos de actuadores y recursos E/S del microcontrolador . . . . .	92
5.2	Tipos de sensores y recursos E/S del microcontrolador . . . . .	92
5.3	Recursos de la familia de microcontroladores dsPIC33FJXMC . . . . .	94
5.4	Elementos de carga representativos relacionados con el esquema de alimentación un robot de Eurobot . . . . .	96
5.5	Estimación de la potencia máxima del robot P.Tinto (Eurobot 2015) . . . . .	97
5.6	Estimación de la potencia máxima del robot P.Tinto (Eurobot 2015) . . . . .	97



# Capítulo 1

## Introducción al proyecto

*Todo el mundo es capaz de hacer cosas extraordinarias, cada uno a su manera. Algunos son perfectamente felices haciendo cosas simples de buen humor; otros, sin embargo, se concentran en los detalles. Somos todos diferentes, y no importa realmente si te centras en viajes espaciales o trabajas en el campo. Lo que importa es hacer lo que verdaderamente quieres.*

Alexander Kemurdzhian

### 1.1 Presentación

Este trabajo de fin de carrera trata sobre el desarrollo de robots para la competición de robots Eurobot [1]. Pretende ser un manual de referencia para todo aquel que quiera construir un robot para participar en Eurobot<sup>1</sup>. Todo lo que se expone en este libro se sustenta en la experiencia adquirida en el desarrollo de robots participantes en Eurobot entre los años 2003 y 2015. Especialmente en el periodo entre 2010 y 2015.

### 1.2 La competición Eurobot

Eurobot es un concurso internacional de robótica creado en 1998 a partir de la Copa de Francia de la robótica [2] (Coupe de France de robotique). El concurso está abierto a equipos de jóvenes, organizados ya sea en proyectos de estudiantes o en clubes independientes. Eurobot se lleva a cabo en Europa, pero da la bienvenida a los equipos de todos los continentes.

Cada año, las reglas se publican alrededor de octubre. Los equipos tienen entonces varios meses para diseñar y construir sus robots y competir entre abril y junio en el concurso nacional, en el cual se pueden seleccionar sólo 3 equipos por país. Los mejores equipos de cada país se reunirán después para competir en las finales de Eurobot.

Las reglas cambian cada año, sin embargo, la base es siempre la misma:

- Los robots tienen aproximadamente el tamaño de un cubo de 30 cm.

---

<sup>1</sup>O para desarrollar un robot de similares características para cualquier otro cometido

- El campo de juego tiene una dimensión de 2 metros de ancho y 3 metros de largo.
- El juego tiene 90 segundos de duración.
- Los robots tienen que llevar a cabo varias tareas en el campo con el fin de ganar tantos puntos como sea posible.
- Los robots tienen que evitarse unos a otros y evitar los choques.
- Los robots son completamente autónomos y deben hacer todas las acciones por ellos mismos.

Tal y como está diseñado, Eurobot constituye un reto de ingeniería que permite a alumnos de electrónica, mecánica e informática aplicar sus conocimientos o ampliarlos y compartirlos. También hace, por ejemplo, que alumnos de electrónica aprendan mecánica y/o informática. Dada la complejidad de los robots suelen ser construidos por equipo formados de al menos 2 personas, lo cual implica cierta organización, división de tareas y trabajo en equipo. Un equipo numeroso permite desarrollar robots con partes más especializadas pero implica una gestión y coordinación mayor. Por otro lado, Eurobot implica confeccionar un producto completo, original y funcional desde cero<sup>2</sup> (o como diríamos en Inglés *from scratch*) en cosa de 6 o 7 meses. Esto implica una elección y compra de materiales, componentes, diseño electrónico y mecánico, construcción de prototipos, desarrollo de software, ... y en definitiva una gestión del tiempo. Tiempo que suele escasear porque los participantes, generalmente alumnos, disponen de él escasamente<sup>3</sup>.

La comunidad de Eurobot se comunica principalmente a través de sus foros [6]. Hay tres foros principales: un foro relacionado con las reglas de cada año, el foro de Eurobot Open [7] y el foro de la Copa de Francia de la robótica [8]. Este último es el foro más activo dado que la Copa nacional de Francia es la de mayor tradición y la que reúne el mayor numero de participantes. Este foro, aunque está escrito en francés, es el que acumula la mayoría de la información acerca de la competición y de temas técnicos. Además, cada equipo suele tener una página web, blog o canal de Youtube donde comparte los avances de los robots que construyen cada año.

Eurobot está organizada gracias la asociación internacional de Eurobot y a la asociación francesa *Planet Sciences* [9] y a muchas otras asociaciones, escuelas, universidades y personas voluntarias de los países participantes. Es increíble la cantidad de personas que apoyan Eurobot y trabajan de forma voluntaria sin ninguna remuneración más que la satisfacción de estar implicados y formar parte de esta fiesta que es Eurobot. Y esto incluye a los participantes, ya que ganar Eurobot no implica ningún premio económico. En total, un montón de gente que trabaja para disfrutar y divertirse con la robótica.

### 1.3 El equipo *Eurobotics Engineering*, historia y antecedentes

*Eurobotics Engineering* [3] es un equipo participante en Eurobot formado por entusiastas de la robótica. Está formado por estudiantes de la Universidad de Alcalá (UAH): ingenieros de telecomunicación, de electrónica, de telemática y de electrónica industrial.

La historia de este equipo se remonta al año 2003, año en el que el Departamento de Electrónica de la UAH decide montar un equipo patrocinado a partir de la iniciativa de un grupo de alumnos de participar en Eurobot 2002. Los integrantes del equipo se renuevan cada año excepto algunos que se mantienen. Exceptuando el año 2009, el resto de años el equipo participó en Eurobot. En el año 2010 Diego Salazar

---

<sup>2</sup>No siempre se ha de empezar desde cero, un desarrollo modular y estructurado permite partir de una base mínima.

<sup>3</sup>Y parece ser que menos aún con los actuales planes de estudios Bolonia

y Javier Baliñas crean el equipo de Eurobot llamado *Eurobotics Engineering* y en el año 2011, junto con Nilo Masias, fundaron la Asociación de Robótica de Coslada (ARC).

El equipo Eurobot Engineering participó en Eurobot desde el año 2010 hasta el año 2015 (exceptuando el año 2013). Durante estos años ha contado con el apoyo del Dpto. de Electrónica de la Universidad de la UAH [10, 11] y el Ayuntamiento de Coslada, además del patrocinio de Ayudas Hidráulicas [12], Carpintería Inglés, Ro-botica [13] (2010-2012), el Consejo de estudiantes de la UAH [14] (2014-2015) y Mobile Minds [15] (2015).

La siguiente tabla muestra un resumen de los robots desarrollados para la competición Eurobot desde el año 2003 hasta el año 2015, en los que ha participado el autor del libro junto con otras personas implicadas en su desarrollo.

Tabla 1.1: Robots desarrollados para la competición Eurobot (2003-2015).

Año	Prueba	Robots	Integrantes equipo
2003	Heads or Tails	Husillo	Julio Pastor Juan Carlos García Diego Alonso García Enrique Moraleja García Jose Manuel Gómez Sanchez Alberto Sanchez Rodríguez Gustavo Tapias Javier Peletier Jesús Oliva Santiago Cobreces Alvarez Daniel Pizarro Perez Pedro Yagüe Pedro Jiménez Mario Inglés Garcés Javier Baliñas Santos

Continúa en la página siguiente

**Tabla 1.1 – continúa en la página anterior**

Año	Prueba	Robots	Integrantes equipo
2004	Coconut Rugby	Electrococo y Joselito	Julio Pastor Oscar Gónzález Santiago Cobreces Diego Alonso García Enrique Moraleja García Jose Manuel Gómez Sanchez Raúl Sanchez de la Montaña Javier Antunez Santiago Castillo Saleta Nistal Javier Rodriguez David Mollejo Noelia Requena Till Zenthofer Cesar Guerra Cesar Lozano Jordi Polo Jorge Fernandez Mario Inglés Garcés Javier Baliñas Santos
2005	Bowling	Campanolo	Julio Pastor Mario Inglés Garcés Javier Baliñas Santos
2007	Recycling	Mr. Proper	Julio Pastor Sergio Arroyo Sierra Marcelo Salazar Acucci Diego Salazar Acucci Adrian Díaz Collazo Rubén Arcos Moreno Mario Inglés Garcés Javier Baliñas Santos
2008	Mission to Mars	Topolino	Sergio Arroyo Sierra Marcelo Salazar Acucci Diego Salazar Acucci Mario Inglés Garcés Javier Baliñas Santos
2010	Feed the world	Trompetero	Mario Inglés Garcés Diego Salazar Acucci Javier Baliñas Santos
2011	Chess'Up!	Zamorano	Nilo Masiás Carranza Mario Inglés Garcés Diego Salazar Acucci Javier Baliñas Santos
Continúa en la página siguiente			

**Tabla 1.1 – continúa en la página anterior**

Año	Prueba	Robots	Integrantes equipo
2012	Treasure island	Crispín y Automático	Sergio Arroyo Sierra Nilo Masias Carranza Mario Inglés Garcés Diego Salazar Acucci Silvia Santano Guillén Javier Baliñas Santos
2013	Happy birthday	-	-
2014	Prehistobot	Grosnik y Seskäpa	Mario Inglés Garcés Diego Salazar Acucci Silvia Santano Guillén Javier Rodriguez Puigvert Miguel Ortiz Carlos de la Rubia Rubén Espino San José Javier Baliñas Santos
2015	Robomovies	P.Tinto y Tirantes	Mario Inglés Garcés Silvia Santano Guillén Javier Rodriguez Puigvert Carlos de la Rubia Rubén Espino San José Javier Baliñas Santos

## 1.4 La robótica educativa

A día de hoy la robótica constituye una herramienta educativa muy potente. Permite aprender y poner en práctica diferentes ramas de la ciencia. Lo hace a través de la ingeniería y las áreas de la ciencia en las que se sustenta (matemáticas, física, química, entre otras) y utilizando la tecnología actual. Este tipo de educación se denomina STEM [16] acrónimo del inglés *Science, Technology, Engineering and Mathematics* (Ciencia, Tecnología, Ingeniería y Matemáticas). Además permite potenciar la creatividad de las tecnologías digitales, más allá de la simple explotación de ellas [17].

Además si la robótica se hace en equipo, como suele ser habitual en Eurobot, la robótica permite desarrollar habilidades y competencias [18] como el trabajo en grupo, la motivación personal, gestión de retos y objetivo, y el manejo del fracaso y las emociones que implican los errores, entre otras.

Este trabajo de fin de carrera constituye en sí un ejemplo de la robótica como herramienta educativa y de sus resultados. Los robots desarrollados han ayudado adquirir experiencia, conocimientos y experiencias de gran valor añadido a los estudios universitarios de los diferentes integrantes del equipo *Eurobotics Engineering* y anteriores.

En el campo de la ingeniería electrónica y mecánica, se ha ganado experiencia en el proceso de diseño, fabricación y montaje de productos que implican una electrónica y una mecánica (tarjetas electrónicas, piezas mecánicas, mecanismos, cableado, etc). Se han aprendido a trabajar con diferentes tipos de mate-

riales (madera, plástico, aluminio, polímeros, etc), a utilizar herramientas y a fabricar piezas mecánicas por diferentes procesos de fabricación (impresión 3D, fresado, plegado, inyección en silicona, etc.).

El simple hecho de que los robots se muevan y que lo hagan hacia lugares concretos implica la aplicación de las matemáticas y la física. Así, para ir de un punto  $A$  a un punto  $B$  se utiliza la trigonometría. Esto es, el robot necesita mirar a  $B$  (girar un ángulo  $\alpha$ ) y luego avanzar una distancia  $d$ . Y además si se desea el tiempo en ir de  $A$  a  $B$  sea el menor posible es necesario utilizar las leyes de Newton para el cálculo de la aceleración y velocidad máxima, que permitan al robot no derrapar ni pasarse de frenada.

Especialmente, dado que un robot en última medida es controlado desde un microcontrolador o procesador similar, la robótica implica adquirir conocimientos sobre la ingeniería del software y el desarrollo de sistemas embebidos[19]. Seguramente algún momento se quiera utilizar una librería o código desarrollada por otras personas pero que resulta muy útil. Se tendrá que aprender a manejar y confiar en librerías de terceros y si son de código abierto quizás permita colaborar añadiendo nuevas funcionalidades o solucionar bugs. Se aprende mucho analizando y comprendiendo el código de otros. Además seguramente estas librerías utilizarán algún sistema de control de versiones distribuido como Git, Mercurial o centralizado como SVN o incluso sistemas más antiguos como CVS. La mayoría de los desarrolladores de software son reacios al principio a utilizar un sistema de control de versiones, suele ser una decisión personal que normalmente cae por su propio peso cuando se trabaja con proyectos de código extensos.

A medida que el código desarrollado vaya creciendo surgen planteamientos de como organizarlo, cómo reutilizar código de un año a otro o incluso cómo reutilizar código del microcontrolador que se está utilizando para poder utilizarlo en otros proyectos. Esto lleva a desarrollar código de una forma más modular y estratificada. Quizás más adelante, el microcontrolador utilizado hasta el momento se quede escaso de recursos, se quiera cambiar a otro de la misma familia pero con más recursos y reutilizar las librerías que se han desarrollado hasta entonces y mantener la compatibilidad con el antiguo microcontrolador. Esto por ejemplo se puede lograr mediante la compilación condicional.

Al ir ganando experiencia, se es capaz de detectar las partes fundamentales y comunes a la hora de implementar una aplicación o funcionalidad. Si se observa como lo ha resuelto el resto del mundo se cae en la cuenta que hay ciertas cosas que ya están muy estudiadas y aparecen en cualquier aplicación que desarrollemos. Así se podrá detectar y aplicar patrones que facilitan desarrollo.

Quizás llegue un momento en que el tiempo que lleva realizar la depuración del robot sobre el entorno real es mucho y es un proceso lento, o quizás no se disponga de dicho entorno porque no queda otra opción que desarrollar de forma remota. Para todo eso, es muy útil el uso de simuladores o bancos de pruebas<sup>4</sup> que permiten emular las interfaces de entrada y salida del robot (sensores y motores) o sus estímulos (obstáculos).

La robótica se disfruta, es divertida, pero también se sufre y requiere de un indudable ejercicio de abstracción y concentración. Es necesario investigar sobre cosas nuevas o reforzar conocimientos ya adquiridos, aplicarlos y madurarlos. Son más comunes los errores que los aciertos, nada sale a la primera, y la *ley de Murphy*<sup>5</sup> suele no fallar. Todo esto lleva al aprendizaje y aplicación del *método científico* para experimentar o para resolver problemas encontrados, como comportamientos no esperados del robot.

Durante el desarrollo de la robótica, puede parecer que es una actividad poco agradecida o agradecida en momentos muy cortos. Pero merece la pena y además se demuestra que más tarde, en el mundo laboral o de la investigación, es cuando la robótica tiene gran parte su retorno<sup>6</sup>.

---

<sup>4</sup>También conocidos en la literatura anglosajona como *sandbox*.

<sup>5</sup>"Si algo puede salir mal, saldrá mal"

<sup>6</sup>El autor de este proyecto está convencido de ello y lo ha experimentado.

## 1.5 Aprendiendo de gigantes.

Hoy en día es casi impensable que una sola persona pueda realizar avances en los diferentes campos de la ciencia equiparables a los descubrimientos de los grandes científicos de la historia. Así la investigación y el desarrollo tecnológico son llevados a cabo por grupos de personas cada cual experta en ciertos tema concretos. Por otro lado la tecnología y sus aplicaciones cada vez son más complejas y eso es en gran parte a que los nuevos desarrollos se apoyan en desarrollos anteriores que permiten abstraerse de aquello que se encuentra fuera del alcance de un objetivo concreto.

Quizá el ejemplo más conocido a día de hoy es la plataforma de desarrollo Arduino [20] que permite desarrollar sistemas embebidos partiendo de un conocimiento muy básico o nulo sobre microcontroladores. Esto es gracias a que la plataforma de desarrollo Arduino tiene una vasta librería de funciones que abstraen la programación de bajo nivel del microcontrolador de forma que no se *escribe un bit* de uno o varios *registros* de nombres aparentemente sin significado, si no que se *detiene*, se *cambia el sentido* o se *cambia la velocidad* de un *motor*. Además todas esas funciones se encuentran documentadas y son accesibles [21] para su modificación o consulta. Así, si una funcionalidad de una librería no existe, es posible estudiar como está implementada y añadir nuevas funcionalidades.

En cierta forma, esta manera de abordar el aprendizaje y el desarrollo de aplicaciones tecnológicas es una forma natural y acorde con el desarrollo tecnológico de la sociedad actual. Hoy en día la tecnología que utilizamos en el día a día es compleja y muy abstracta por lo que no tiene sentido que el aprendizaje de la misma comience desde los niveles más básicos. Plataformas como Arduino permiten comenzar en un punto intermedio y poder crecer hacia niveles de abstracción mayores (grano grueso) o hacia niveles inferiores más específicos (grano fino).

Pues bien este proyecto no es una excepción y también se ha apoyado en hombros de gigantes. El código fuente de los microcontroladores de algunos de los robots desarrollados (Husillo, Joselito o Campanolo) fue escrito desde cero. Otros (Electrococo, Mr.Proper y Topolino) se ayudaron del código escrito en años anteriores para otros robots. Y finalmente a partir del año 2010 los robots (Trompetero, Zamorano, Crispín, Automático, Grosnik, Seskäpa, P.Tinto y Tirantes) se desarrollaron a partir del código abierto escrito por terceros.

El utilizar código abierto de terceros vino dado por el objetivo de evolucionar tecnológicamente los robots a desarrollar para Eurobot. Para ello se estudió los fundamentos técnicos y tecnológicos de los robots por aquella época obtenían muy buenos resultados como RCVA [22], Microb Tecnology [5], RCA Archen [23] o Turak [24]. Al final de dicho estudio se establecieron los requisitos de la mecánica, hardware y software necesarios, y se desarrollo una plataforma robótica base (el *Dummy robot*, ver 1.1) con una mecánica y una electrónica de acuerdo a los requisitos buscados.

Al final del proceso de estudio se descubrieron las librerías Aversive desarrolladas por el equipo *Microb Technology* para microcontroladores AVR de Atmel. Por aquél entonces, la electrónica que se estaba diseñando estaba basada en microcontrolador dsPIC de Microchip. Se podría haber migrado el HW y utilizar microcontroladores AVR, pero al final se optó por migrar las librerías Aversive a la arquitectura de dsPIC, lo que se denominó Aversive4dsPIC<sup>7</sup>.

La decisión de utilizar estas librerías y portarlas permitió no solo aprender a utilizarlas desde la capa de aplicación (el robot) sino estudiar cómo estaban implementadas y adaptarlas para ser compatibles con una arquitectura diferente. En definitiva se reutilizó el código de terceros, lo cual implica mucho más que el simple hecho de aprovechar el trabajo realizado por otros. En este caso las librerías Aversive [4] son unas librerías libres desarrolladas por varias personas, versionada bajo un sistemas de control de

<sup>7</sup>Pronunciado *Aversive for dsPIC*, Aversive para dsPIC

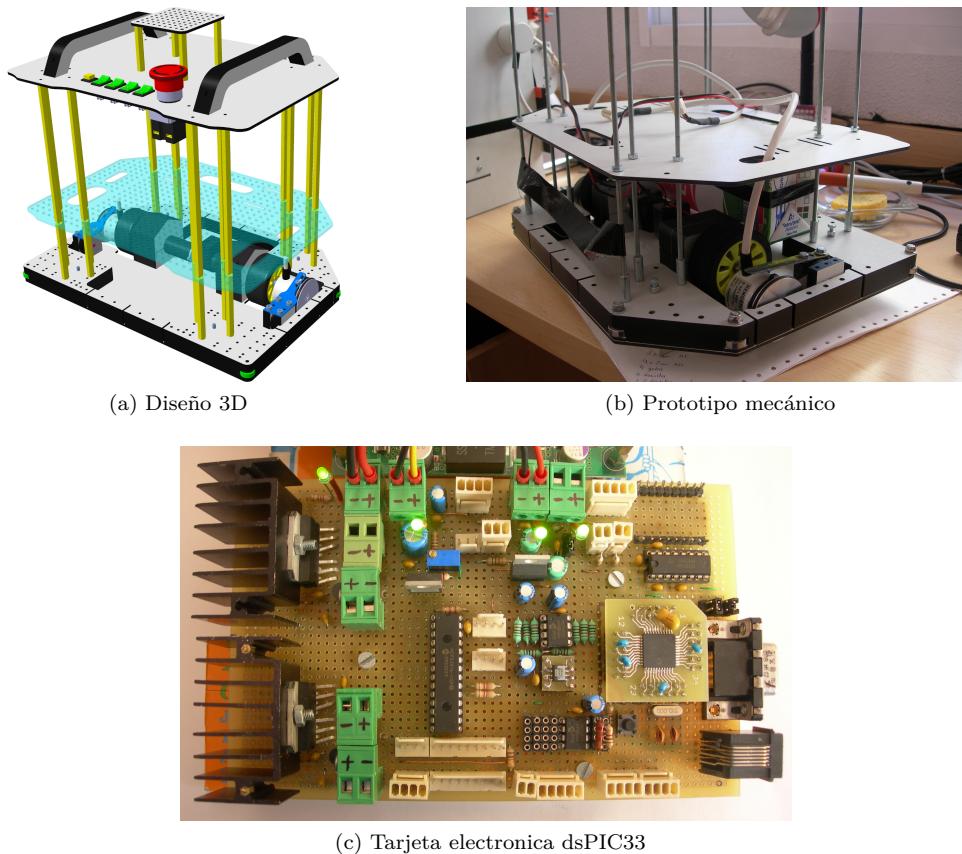


Figura 1.1: *Dummy robot*. Plataforma robótica base desarrollada en 2010 por el equipo Eurobotics Engineering. Diseño 3D, prototipo mecánico y electrónica basada en microcontrolador dsPIC33.

versiones, que están muy bien estructuradas, escritas siguiendo una guía de estilo común, que explota al máximo el lenguaje ANSI C, que cuenta con un *toolchain*<sup>8</sup> que utiliza herramientas libres, que implementan todo tipo de funcionalidades y muchas de ellas dirigidas al desarrollo de robots como los de Eurobot. La migración de las librerías Aversive se realizó siguiendo y manteniendo toda las características anteriormente mencionada, lo cual implicó el aprendizaje de las mismas y de todos los conceptos derivados y subyacentes de ellas.

## 1.6 Motivación y objetivos

Todo el trabajo realizado en el desarrollo de los diferentes robots viene principalmente motivado por la afición a la robótica, la electrónica, los sistemas embebidos y por la inquietud de inventar y crear cosas.

Por otro lado, el objetivo fundamental de este trabajo es la divulgación del conocimiento adquirido en la realización de robots que compiten en Eurobot, siendo los objetivos específicos de este proyecto los siguientes:

- Reflexionar y analizar diferentes aspectos indirectamente relacionados con el desarrollo de ingeniería (el equipo de trabajo, organización, ...)
- Analizar y describir los fundamentos de ingeniería implícitos (dinámica, cinemática, sistemas de control, ...).

<sup>8</sup>Cadena de herramientas: conjunto de programas mediante los cuales se construye el programa que se ejecuta en el microcontrolador

- Describir las partes fundamentales del desarrollo (movimiento, navegación, posicionamiento, estrategia ...) y diseño un robot de Eurobot (mecánica, electrónica, programación, ...).
- Presentar implementaciones y diseños reales de cada una de las partes fundamentales de un robot de Eurobot.
- Documentar todas aquellos diseños, desarrollos o algoritmos realizados que puedan servir de punto de partida o ser reutilizados en futuros desarrollos (diseños electrónicos, librerías software, herramientas de simulación, ...).

## 1.7 Organización del libro

El libro cuenta con 5 capítulos adicionales a este capítulo:

### **El robot de Eurobot**

Capítulo que describe el robot de Eurobot desde el punto de vista de su desarrollo: las especificaciones funcionales, las partes principales de un robot de Eurobot, las características de la mecánica, electrónica y software, y las fases de su desarrollo.

### **Plataforma robótica base**

Descripción de lo que se ha considerado plataforma robótica base, un robot base cuyas características y funcionalidades son comunes a cualquier juego de Eurobot y sobre el que desarrollar nuevos robots participantes. El capítulo se centra en la plataforma robótica desarrollada de tracción diferencial y control de posición polar mediante odometría. Además se describe el sistema desarrollado para la detección y evitación de los robots oponentes.

### **Sistema de balizas tipo faro:**

Este capítulo se centra en la medida de la posición del robot oponente respecto al robot apoyándose en los soportes de balizamiento proporcionados por las reglas del juego. Se presenta uno de los sistemas más utilizados en Eurobot basado en una baliza tipo faro. El capítulo describe los fundamentos de este tipo de sistema y presenta el desarrollo de dos balizas tipo faro de implementación mecánica diferente pero que comparten una misma arquitectura hardware y un mismo software.

### **Desarrollo hardware**

Se presenta la arquitectura hardware a partir de la cual se han desarrollado los diferentes robots desde el año 2010. Se describe cómo la arquitectura fue diseñada y pensada para poder ser reutilizada en cada desarrollo de un robot. También se describen las diferentes implementaciones realizadas de la arquitectura en diferentes tarjetas electrónicas en función de las necesidades de cada tipo de robot desarrollado.

### **Desarrollo software**

El capítulo sobre la arquitectura software implementada en los robots desarrollados. Se describe la organización y funcionamiento de los niveles o capas de los que se compone la implementación SW del robot.



# Capítulo 2

## El robot de Eurobot

*Somos como enanos subidos a hombros de gigantes.  
Nuestra mirada puede abarcar más cosas y ver más lejos  
que ellos. No porque nuestra vista sea más penetrante y  
nuestra estatura mayor sino porque nos ha elevado su  
altura gigantesca.*

Bernardo de Chatres, en *Metalogicon* de Juan de  
Salisbury

Cada robot que participa en Eurobot es único y diferente al resto, sin embargo todos tienen en común ciertas características o funcionalidades que vienen determinadas o derivadas de las reglas del juego. Como se ha comentado en el capítulo 1 cada año la prueba en la que competir o a resolver es diferente, esto tiene una implicación directa en el diseño mecánico especialmente en los mecanismos diseñados para manipular los elementos de juego propios de la prueba. Sin embargo, si se comparan las reglas del juego de varios años se observan características comunes que repercuten en el diseño de la mecánica, de la electrónica o del software.

### 2.1 Especificaciones del robot de Eurobot

Un robot de Eurobot tiene especificaciones independientes de la temática del juego y otras dependientes totalmente. Las primera permiten que una vez estén desarrolladas y validadas puedan ser reutilizadas o mejoradas en futuros diseños. Esto quiere decir que pueden ser un bloque totalmente modular, por ejemplo una tarjeta electrónica, o diseños reutilizables como el diseño electrónico de la tarjeta, cuyo PCB<sup>1</sup> puede diseñarse a medida del robot y en función del espacio disponible.

Por otro lado, los sistemas mecánicos a diseñar para la manipulación de los elementos de juego son en su mayoría específicos de cada prueba. Sólo una pequeña parte del desarrollo relacionado con estos sistemas es reutilizable, como es parte del desarrollo software destinado al control de dichos sistemas mecánicos.

Un robot de Eurobot ha de ser completamente autónomo y básicamente ser capaz de:

1. Desplazarse hasta las diferentes zonas del campo donde se sitúan los elementos del juego o donde se almacenan.

<sup>1</sup>Del inglés *Printed Circuit Board*, Tarjeta de Circuito Impreso

2. Evitar chocar con elementos del campo de juego y con otros robots oponentes durante los desplazamientos entre zonas del campo.

Además, si las reglas permiten dos robots por equipo: un robot principal y uno secundario, es recomendable que ambos robots sean capaces de comunicarse entre sí con el fin jugar de forma cooperativa.

Con el fin de facilitar la detección entre los robots y que estos sean capaces de evitar colisiones entre ellos, los robots han de reservar un espacio para sensores y balizas a una altura determinada como muestra la figura 2.1. Así, cada robot puede colocar una baliza o marca en el robot oponente y disponer de un espacio donde situar los sensores apropiados para detectar dicha baliza.

Estas balizas pueden ser activas o pasivas. Por ejemplo, un sistema compuesto por una baliza maciza y un sensor de ultrasonidos puede ser utilizado para detectar a qué distancia y dirección se encuentra un robot oponente antes de tomar la decisión de hacia dónde redirigirse.

Además el campo de juego cuenta con 3 soportes para balizas adicionales que están a disposición de cada equipo (ver figura 2.2). Estos soportes están pensados para facilitar la localización de los robots en el campo de juego. Por ejemplo, un sistema compuesto por una baliza reflectante colocada en el soporte de una esquina del campo y un sensor de luz reflexivo en el robot puede ser utilizado para localizar una canasta que se encuentre en dicha esquina del campo.

Las balizas pueden ser activas o pasivas y tiene como únicas restricciones sus dimensiones y peso:

- Balizas colocadas en el oponente: dimensiones máximas de 80x80x80 cm y peso máximo de 0,5 Kg.
- Balizas del campo de juego: dimensiones máximas de 160x80x80 cm.
- Espacio para sensores en cada robot: 80x80x80 cm.

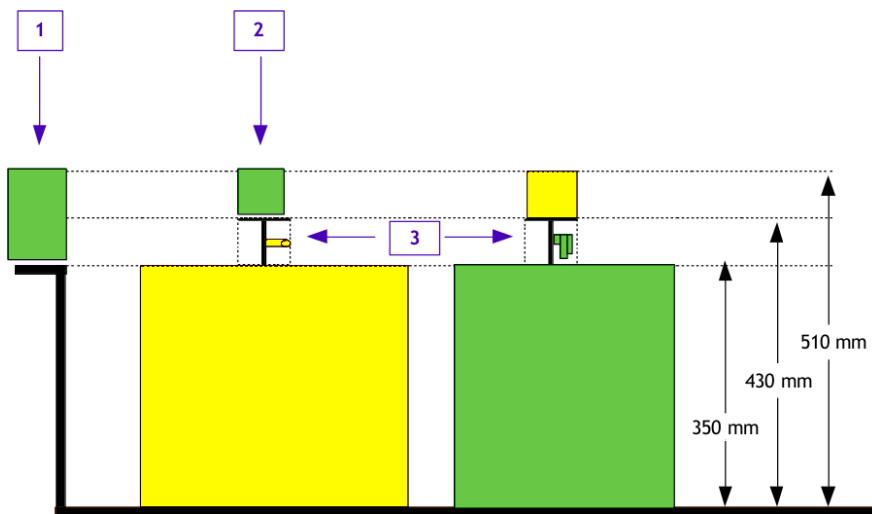


Figura 2.1: Altura de balizas y sensores. Vista de perfil de dos robots oponentes y el campo de juego. 1- Soporte para balizas del campo de juego. 2- Soporte para baliza del oponente. 3- Espacio para sensores de balizas.

Por otro lado, el robot ha de ser capaz de resolver la temática del juego lo cual implica el manejo de los elementos o piezas propias del juego. Por ejemplo, en Eurobot 2015, juego titulado como *Robomovies* (temática sobre el cine), una de las acciones por las que los robots obtenían puntos era por hacer una torre con piezas cilíndricas coronada con una pelota de tenis que simbolizaba un foco de cine.

Además, mecánicamente un robot ha de cumplir las siguientes restricciones dimensionales:

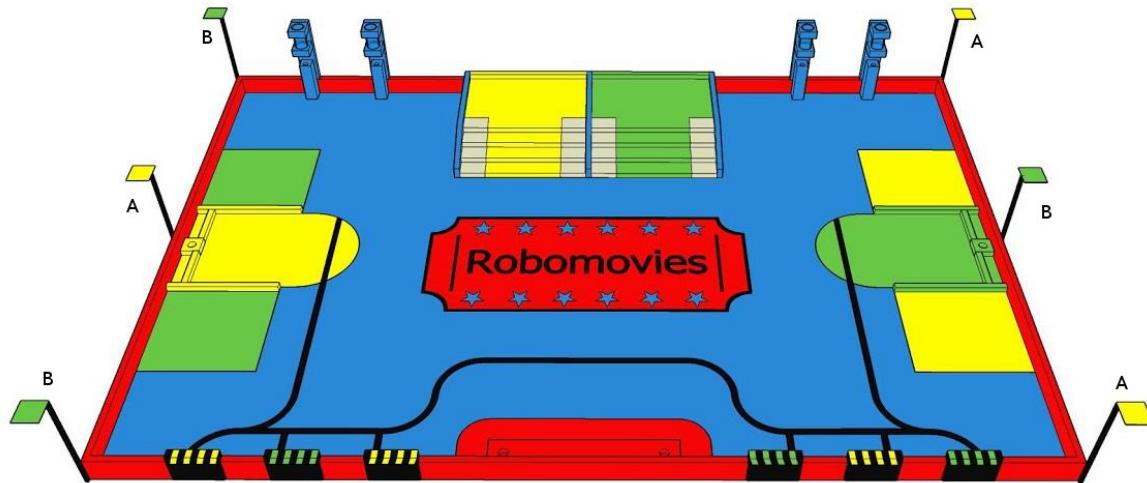


Figura 2.2: Campo de juego de Eurobot 2015. Soportes para balizas fijas de los equipos A y B.

- Perímetro máximo inicial 1200 mm (700 mm para el robot secundario)
- Perímetro máximo desplegado 1500 mm (900 mm para el robot secundario)
- Altura máxima de 350 mm

Por último, las reglas de Eurobot además especifican otros temas que deben de ser tenidos en cuenta en caso de que apliquen:

- Fuentes de alimentación y baterías
- Utilización de fuentes de luz y láseres
- Utilización de sistemas de comunicación de radio
- Utilización de sistemas de aire comprimido
- Botón de parada de emergencia
- Método de comienzo y finalización de un partido
- Procedimientos de un partido.

## 2.2 Elementos del juego y tareas de un robot de Eurobot

Los elementos de juego en Eurobot varían cada año con la temática. Se pueden identificar 2 tipos de elementos básicos:

- **Elementos no compartidos** Elementos que sólo pueden ser manipulados por cada robot contrincante y puntúan y pertenecen a cada robot. El número de elementos de este tipo es el mismo para cada robot. La manipulación de los elementos del robot oponente puede estar penalizada.
- **Elementos compartidos** Elementos comunes a los robots contrincantes de un partido. Ambos robots pueden manipularlos y conseguir puntos por ellos, pero no pertenecen a ninguno, pueden ser utilizados por el contrincante sin penalización alguna.

Atendiendo a la posición de los elementos se obtiene la siguiente clasificación:

- **Elementos de posición conocida:** La posición en el campo (coordenadas) es conocida a priori.
- **Elementos de posición aleatoria:** La posición en el campo es aleatoria entre varias posiciones posibles. Por ejemplo, en Eurobot 2016 la posición de las estrellas de mar era aleatoria. Previamente a la colocación de los elementos de juego por los árbitros echaban a suertes la posición de las estrellas de mar de cada robot.
- **Elementos aportados por los robots:** Algunos elementos pueden ser aportados por los robots al partido. Así al iniciar éste ciertos elementos pueden estar inicialmente en los robots. Por ejemplo, es el caso de unas pelotas de *scuash* que el Eurobot 2005 los robots podían disparar para derribar las torres de bolos del oponente.

Los elementos del juego suelen tener asociada una tarea por la cual se obtienen puntos en un partido. Dichas tareas simbolizan acciones relacionadas con la temática de la prueba. Se pueden identificar varios tipos de tareas atendiendo a su grado de complejidad:

#### Accionar o empujar una palanca o mecanismo:

Este tipo de tareas son las más sencillas, suele consistir en empujar un mecanismo simple. Por ejemplo, en Eurobot 2012 (*Treasure Island*) los robots tenían que empujar un mecanismo, situado en una de las paredes del campo, por el cual se desplegaba un mensaje en rollado. Esto simbolizaba la acción de mandar un mensaje en una botella

#### Recolectar y almacenar:

Ciertas acciones consisten en recolectar elementos de juego y llevarlos hasta un sitio donde almacenarlos. Por ejemplo, en Eurobot 2015 (*Robomovies*) los robots tenían que recolectar vasos de plástico con pelotas de poliestireno en su interior en su interior, que simbolizaban palomitas, y llevarlos hasta las zonas que simbolizaban los cines. Además, inicialmente los vasos no estaban completamente llenos de palomitas de forma que los robots podían ir hasta las máquinas de palomitas (unos dispensadores de pelotas) para rellenarlos y así conseguir más puntos.

#### Recolectar selectivamente y almacenar:

A veces, además de recolectar y almacenar elementos de juego, es necesario discriminar ciertos elementos que penalizan restando puntos. Es el caso de los frutos de Eurobot 2014 (*Prehistobot*), de entre los frutos (corchos morados) que colgaban de los árboles había 2 colocados de forma aleatoria que eran tóxicos (corchos negros).

#### Recolectar y construir:

Este tipo de tareas consisten en construir un elemento de juego a partir de piezas de éste que son recolectadas por el campo. Por ejemplo, en Eurobot 2009 (*Temples of Atlantis*) los robots tenían que recolectar partes de columnas (fichas cilíndricas) y linteles (fichas rectangulares) para construir templos de dos columnas lo más altos posibles.

#### De habilidad:

Algunos años los robots han tenido que demostrar cuánto de hábiles son desafiando la gravedad. Por ejemplo, en Eurobot 2015 (*Robomovies*) los robots obtenían puntos por subir unas escaleras en las que previamente habían extendido la alfombra roja. O por ejemplo en Eurobot 2011 (*Chess'Up!*), donde los robots tenían que acabar sentados sobre una de las fichas que simbolizaban peones de Ajedrez en su casilla de salida.

**Funny action:**

La *Funny action* es una acción que está pensada para ser realizada por los robots al finalizar el partido. Por ejemplo, en Eurobot 2016 (*The Beach Bots*) al finalizar el partido los robots tenían que desplegar una sombrilla o toldo que llevaban consigo para protegerse del sol.

## 2.3 Partes principales en el desarrollo de un robot de Eurobot

Teniendo en cuenta las especificaciones descritas en la sección anterior y la temática de la prueba de Eurobot, el desarrollo de un robot de Eurobot se puede dividir en las siguientes partes principales:

1. Estrategia de juego.
2. Desplazamiento y localización.
3. Evitación de obstáculos.
4. Comunicación entre robots (en caso de jugar con 2 robots).
5. Manipulación de elementos de juego.

Así, tal y como representa la figura 2.3, un único robot está compuesto por las tres primera partes y sólo en caso de contar con 2 o más robots es necesaria la parte de comunicación entre robots.

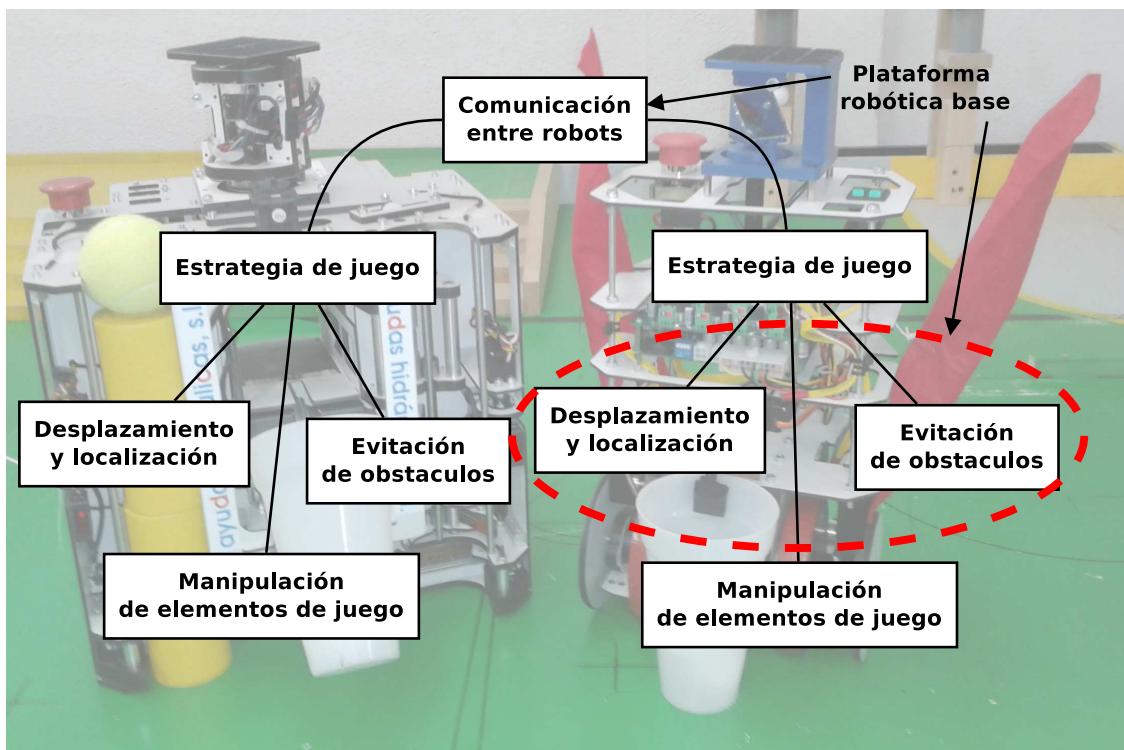


Figura 2.3: Partes principales del desarrollo de un robot de Eurobot. En el fondo robot P.Tinto y Tirantes (robot secundario) desarrollados para Eurobot 2015 por el equipo Eurobotics Engineering

Por otro lado, las partes 2, 3 y 4 forman lo que se ha denominado *plataforma robótica base*. Esta plataforma base está compuesta por las partes que permiten un desarrollo independientemente de la temática de la prueba de Eurobot. Así, esta plataforma robótica constituye un desarrollo reutilizable en la concepción de nuevos robots de Eurobot.

Cada una de estas partes principales de un robot de Eurobot pueden implicar un desarrollo mecánico, electrónico y de software, dependiendo del caso.

### 2.3.1 Estrategia de juego

La estrategia de juego constituye el nivel más abstracto del robot y tiene un desarrollo puramente software. La estrategia tiene en cuenta diferentes variables del juego como se indica en la figura 2.4(tareas por realizar, prioridad de tareas, posición del oponente, el tiempo transcurrido de partido, etc.) para tomar decisiones que maximicen la puntuación a conseguir y así ganar el partido.

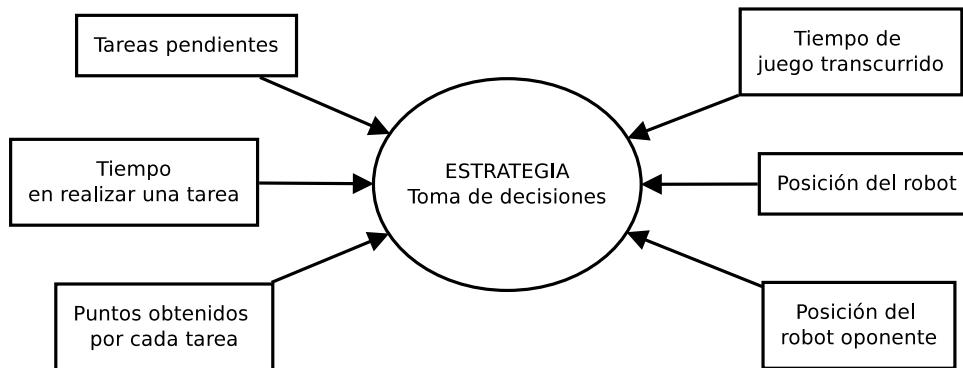


Figura 2.4: Estrategia de juego: variables a tener en cuenta en la toma de decisiones.

El desarrollo de la estrategia puede ser realizado en paralelo a otras partes del robot, pero ha de tener en cuenta los tiempos derivados de tareas que impliquen sistemas mecánicos, como por ejemplo coger una pelota y almacenarla o desplazarse de un punto del campo a otro. Dichos tiempos por lo general son mucho mayores (segundos o decenas de segundos) que el tiempo que tarda el algoritmo de estrategia implementado en tomar una decisión. Es por ello que dicho algoritmo estará la mayor parte del tiempo esperando a que otras partes terminen de realizar su función.

### 2.3.2 Desplazamiento y localización

Un robot de Eurobot está diseñado para funcionar en un entorno de dimensiones conocidas como es el campo de juego (un área de 2x3 m), o lo que es lo mismo un plano cartesiano. Así la posición del robot y otros objetos del campo, como elementos de juego o los robots oponentes, vendrá dada por sus coordenadas rectangulares ( $x, y$ ) y polares ( $d, a$ ) relativas a un origen de coordenadas (0,0) (ver figura 2.5).

La localización permite al robot conocer su posición ( $x_R, y_R$ ) en el campo y poder calcular las distancias (rectangulares o polares) de un punto de destino ( $x_i, y_i$ ) relativas a su posición. El desplazamiento permite recorrer dichas distancias a una velocidad  $v$  y aceleración  $a$  determinadas generando así *trayectorias*.

Mecánicamente es necesario desarrollar un sistema de tracción, que en el caso de los robots desarrollados es un *sistema de tracción diferencial*. Dicho sistema está compuesto por dos motores con reductora que mueven dos ruedas independientes que están en contacto con el campo de juego. Además para poder controlar velocidad y aceleración de cada rueda se utilizan sensores tipo *encoder* que miden cuánto gira cada rueda, o lo que es lo mismo su posición angular relativa. Utilizando la información medida por los

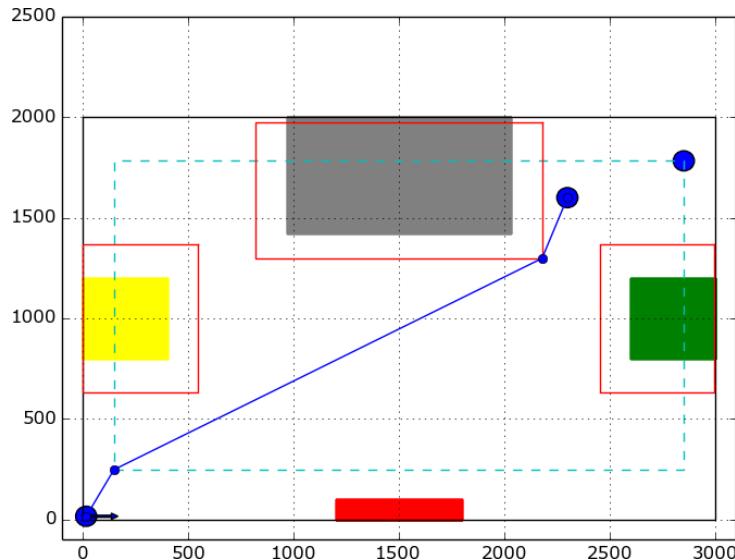


Figura 2.5: Representación cartesiana del campo de Eurobot 2015 (ver figura 2.5) y ejemplo de generación de trayectorias. En horizontal el eje  $x[mm]$ , vertical el eje  $y[mm]$ . Los cuadrados rojos representan obstáculos y el cuadrado azul de rayas la zona útil de desplazamiento del robot. El punto azul grande con flecha indica la posición de inicio y los puntos azules grandes sin flecha indican un punto de destino. Los puntos azules pequeños indican puntos de la trayectoria.

encoders (u otro sistema similar) es posible medir la posición del robot de forma relativa a una posición de inicio mediante *algoritmos de odometría*<sup>2</sup>.

Como muestra la figura 2.6 la electrónica asociada ha de contar con una etapa de electrónica de potencia que permita la excitación de los motores (*drivers* de motores), el acondicionamiento y decodificación de las señales de los encoders, y un microcontrolador donde se implementen los algoritmos de control de posición, velocidad y aceleración de los motores, y el algoritmo de odometría.

Dicho microcontrolador, además de implementar los algoritmos específicos de control de motores y odometría, ha de implementar funciones mucho más abstractas como la *generación de trayectorias*. De forma que en última medida, el robot pueda desplazarse a un punto del campo de juego de forma autónoma o seguir un camino compuesto por varios puntos como el representado en la figura 2.5.

### 2.3.3 Evitación de obstáculos

El robot de Eurobot se encuentra con dos tipos de obstáculos a evitar en sus trayectorias. Por un lado están los obstáculos fijos y de posición conocida como pueden ser partes del campo de juego o elementos de juego al inicio del partido. Y por otro lado, objetos de posición desconocida como son los robots oponentes o elementos de juego que han sido movidos de su posición inicial.

La evitación de obstáculos está directamente relacionada con la generación de trayectorias del robot, teniéndose en cuenta previamente al cálculo de la trayectoria a realizar y durante la misma. Los sistemas de detección de obstáculos a desarrollar dependen de las diferentes situaciones que se dan en un partido de Eurobot:

#### Obstáculo en trayectoria:

Es el caso en el que el robot quiere ir de una zona del campo a otra, por ejemplo para coger o dejar un elemento de juego. El camino a seguir hasta la zona de destino ha de tener en cuenta los espacios

---

<sup>2</sup>De forma complementaria se pueden utilizar sistemas de medida de posición absoluta para corregir los errores de las medidas relativas

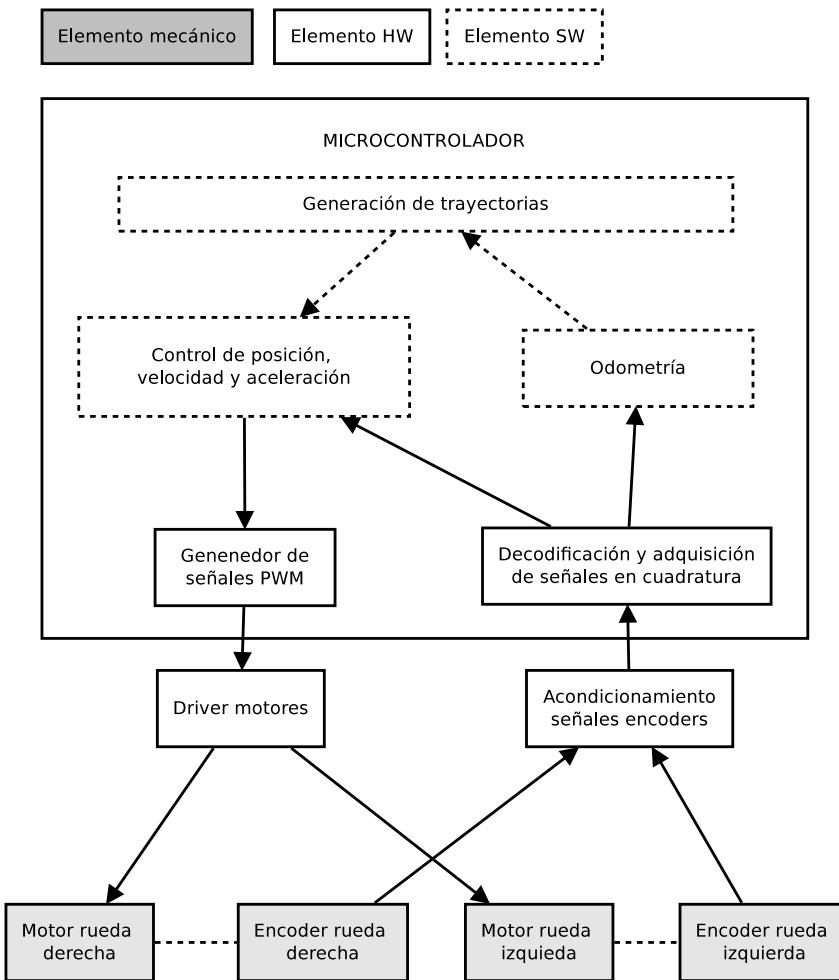


Figura 2.6: Desplazamiento y localización: diagrama de bloques del desarrollo mecánico, electrónico y software implicado.

libres del campo de juego y la posición de los robots oponentes para así seguir trayectorias libres de obstáculos. Como muestra el ejemplo de la figura 2.7, entre el punto origen y destino además de los objetos fijos del campo de juego se encuentra el robot oponente. La generación de trayectorias tiene en cuenta estos obstáculos y genera trayectorias que rodean los obstáculos siguiendo el camino más corto.

La evitación los obstáculo fijos del campo de juego se puede realizar utilizando la *localización del robot*. Si el robot conoce su posición es sencillo evitar chocar con paredes o elementos fijos del campo de juego.

Respecto a la evitación y detección de los robots oponentes es necesario desarrollar un sistema específico para ello. Este sistema normalmente utiliza los soportes de balizas especificado por las reglas de Eurobot (ver figura 2.1). Dependiendo del sistema de balizas utilizado [25] el desarrollo del mismo implicará en mayor o mejor medida un desarrollo mecánico, electrónico o software.

Sea cual sea el sistema utilizado para localizar al robot oponente, debe de aportar la información de la posición del éste de forma relativa a la posición del robot o absoluta respecto al origen de coordenadas del campo de juego (ver ejemplo de la figura 2.8). Además otras características deseables son un rango de detección que cubra todo el campo de juego y un tiempo de medición de la posición del oponente pequeño. El conocer en todo momento la posición del oponente constituye una información muy valiosa a tener en cuenta en la estrategia de juego. Así mismo, un tiempo de

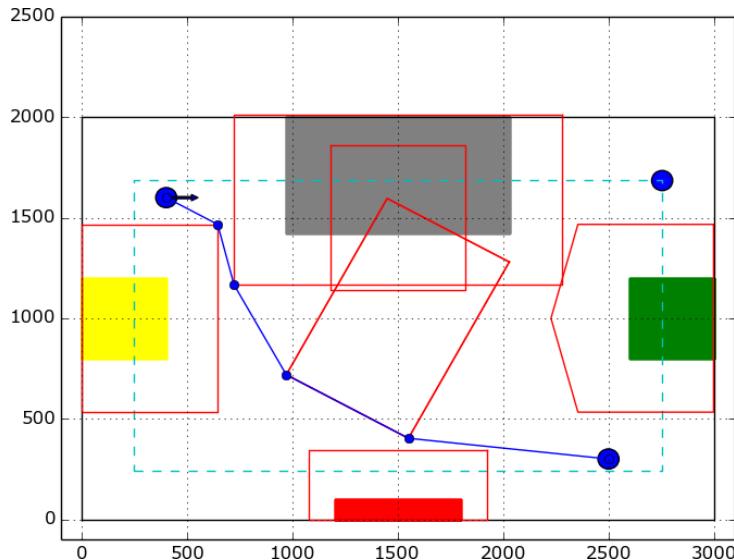


Figura 2.7: Evitación de obstáculos en trayectorias. En el centro un robot oponente. En horizontal el eje  $x[\text{mm}]$ , vertical el eje  $y[\text{mm}]$ . Los cuadrados rojos representan obstáculos y el cuadrado azul de rayas la zona útil de desplazamiento del robot. El punto azul grande con flecha indica la posición de inicio y los puntos azules grandes sin flecha indican un punto de destino. Los puntos azules pequeños indican puntos de la trayectoria.

medición pequeño permitirá un menor tiempo de reacción en caso de peligro de colisión.

#### Obstáculo inesperado en una trayectoria:

Iniciada una trayectoria, que a priori no tiene obstáculos, en cualquier punto de esta puede ocurrir que el robot oponente cambie su posición y se interponga en el camino del robot pudiendo producirse una colisión. En este caso, el robot ha de detectar este obstáculo inesperado y parar o modificar su trayectoria para evitar la colisión.

La detección de un objeto inesperado en una trayectoria se puede realizar utilizando el mismo sistema desarrollado para localizar al robot oponente, si el rango de medida cubre distancias cercanas al robot y el tiempo de medida es suficientemente pequeño respecto a la velocidad de desplazamiento del robot. Aún así, un segundo sistema más específico para estos casos aumenta la fiabilidad y permite hacer una doble comprobación si los rangos de detección de ambos sistemas se solapan.

El sistema a desarrollar permite detectar un objeto cercano (0 a 30cm aproximadamente) en las dirección de avance del robot (parte delantera y trasera) y opcionalmente detectar un objeto situado a los lados del robot (lado izquierdo y derecho). El rango depende de principalmente de la distancia de frenado del robot a velocidad máxima, de forma que la detección de un objeto permita evitar la colisión.

Suelen utilizarse sensores digitales de luz difusa (luz roja o infrarrojos) situados estratégicamente a una altura determinada. En parte delantera y trasera del robot tres sensores son suficientes para detectar un objeto (dos si el robot es estrecho), dos de ellos se sitúan en los extremos de del robot y si fuera necesario un tercero en el punto medio entre ellos (ver figura 2.9). Para la detección de objetos a izquierda y derecha del robot un sensor a cada lado es suficiente. El fundamento es simple, si cualquiera de los dos sensores de los extremos de robot detecta un objeto hay posibilidad de colisión.

#### Obstáculo que bloquea un elemento del juego:

Caso en el que el robot se encuentra en una zona desde donde necesita acceder a un elemento

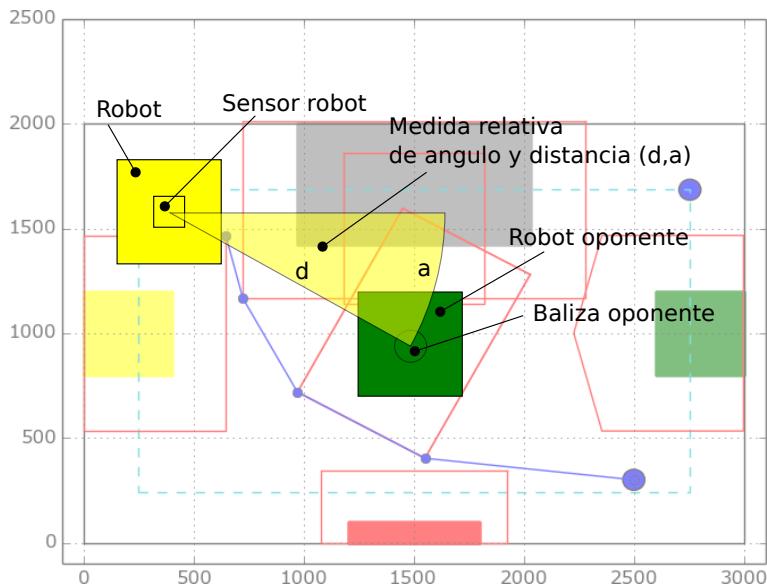


Figura 2.8: Ejemplo de sistema de medida de la posición del robot oponente que utiliza una baliza colocada en el oponente y un sensor en el robot. La medida obtenida es el ángulo y la distancia relativa al posición del robot.

de juego, por ejemplo un dispensador de fichas. Puede ocurrir que uno de los elementos de juego, por ejemplo una pelota u otro elemento, se encuentre entre el robot y el dicho dispensador. La probabilidad de que ocurra esto es mayor con los objetos esféricos y de poco peso, como por ejemplo pelotas de *ping pong*, la manipulación de los mismos puede hacer que éstos rueden a lo largo del campo aleatoriamente con facilidad<sup>3</sup>.

Otro caso, que se suele dar muy pocas veces, es que el robot oponente esté protegiendo un elemento de juego común de forma premeditada utilizando otro elemento de juego, con el fin de dificultar su acceso al otro robot.

En el caso de que un objeto se encuentre entre un elemento fijo del campo, como una pared, y el robot. Si el robot necesita utilizar dicha pared como referencia y un objeto se encuentra en medio, la referencia que medirá el robot será errónea, así como los movimientos que realice respecto a esa referencia. Esta situación se puede prevenir si el robot es capaz de limpiar previamente la zona a la que quiere acceder realizando secuencias de movimientos específicas para ello, o si el objeto bloqueante es detectado mediante sensores.

Una tercera forma de detectar un objeto bloqueante es utilizando la localización del robot, su posición y el error estimado de la misma. Si en el momento de tomar la referencia de la pared la posición medida por la localización del robot es mayor que el error estimado (en distancia o ángulo) algo está impidiendo llegar hasta la posición de referencia.

### 2.3.4 Comunicación entre robots

En caso de disponer de dos robots por equipo hay diferentes formas de implementar una estrategia coordinada:

1. Que cada robot tenga asignado un área del campo de forma que no se estorben.

<sup>3</sup>Este tipo de objetos también puede entrar dentro de los sistemas mecánicos del robot e inutilizalos o bloquearlos

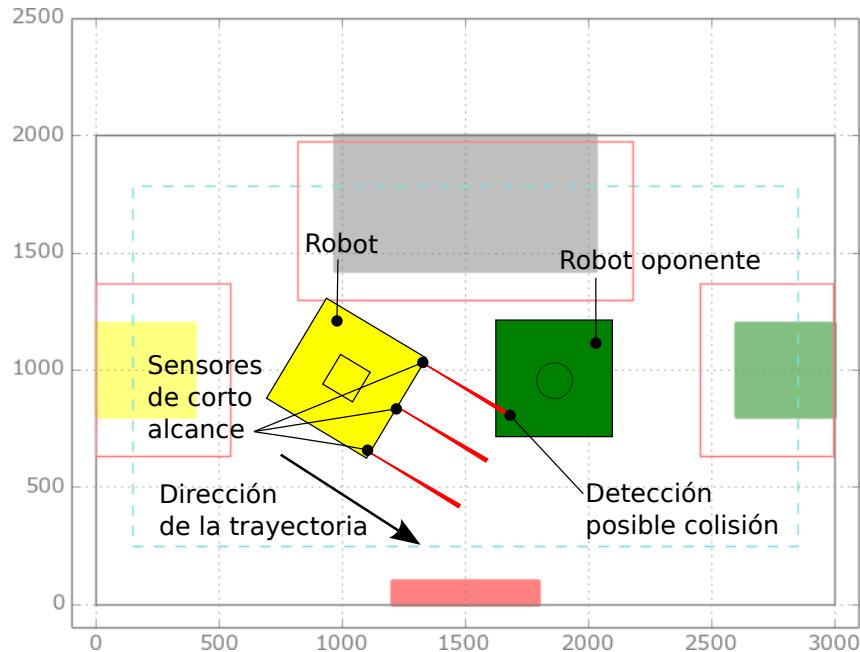


Figura 2.9: Ejemplo de sistema de detección de obstáculos en trayectoria. Tres sensores de corta distancia digitales detectan posibles colisiones.

2. Que cada robot se mueva libremente por todo el campo y considere al robot compañero como un obstáculo más.
3. Que exista una comunicación entre ambos de forma que cada uno conozca la posición del otro trabajen en cualquier zona del campo pero sin estorbarse.

Las dos primeras formas no implican un desarrollo añadido, los mismos sistemas de localización, desplazamiento y evitación de obstáculos desarrollados para un robot pueden replicarse en un segundo robot sin apenas coste añadido, excepto por aquel derivado de sus características mecánicas (diferente dimensiones, motores, sistemas mecánicos...).

La tercera forma necesita del desarrollo de un sistema de comunicación entre ambos robots. Esto implica un desarrollo electrónico y de software principalmente. Es recomendable utilizar una tecnología robusta y flexible que incluya capas de comunicación que aseguren una fiabilidad en la comunicación. Sobre todo porque el entorno en el que se desarrollan las competiciones suele tener muchas perturbaciones de radio frecuencia como equipos de audio y vídeo inalámbricos, redes Wifi o Bluetooth que pueden afectar a la comunicación entre los robots.

Como representa la figura 2.10, cada robot necesita de un módulo de comunicación conectado a un microcontrolador. Bluetooth y Zigbee son dos de las tecnologías más utilizadas en comunicación inalámbrica, existen en el mercado módulos de comunicación de estas tecnologías que permiten crear enlaces de comunicación de forma sencilla. Dichos módulos electrónicos suelen tener una interfaz serie de comunicación (UART, SPI, I2C, ...) de forma que una vez establecido el canal de comunicación inalámbrico la comunicación entre dos microcontroladores es transparente (similar a tener un canal cableado).

Por la parte del software, es necesario desarrollar un protocolo de comunicación, común a los dos robots, a partir del cual enviar y/o recibir mensajes entre ellos.

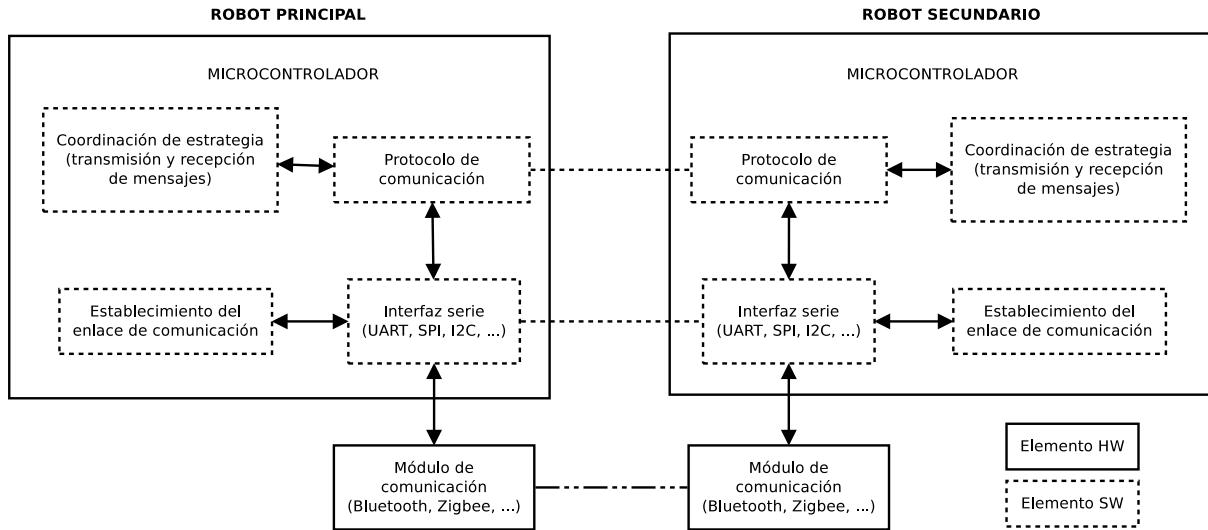


Figura 2.10: Sistema de comunicación entre robots. Ejemplo de diagrama de bloques de las diferentes capas HW y SW de la comunicación. De forma virtual existe una comunicación a nivel de cada capa software entre ambos robots.

### 2.3.5 Manipulación de elementos de juego

En una prueba de Eurobot la forma de conseguir puntos consiste en realizar ciertas tareas que implican la manipulación de elementos de juego. Las acciones a realizar con dichos elementos pueden ser recolectar, almacenar, clasificar y realizar construcciones, entre otras. Dicha manipulación suele ser realizada mediante sistemas mecánicos desarrollados expresamente para la prueba. Se ha considerado que cada sistema mecánico puede estar compuesto por uno o varios mecanismos, y estos a su vez formados por varios actuadores y sensores.

Por ejemplo como muestra la figura 2.11, el sistema de *stands* del robot P.Tinto (desarrollado para Eurobot 2015) estaba compuesto por un mecanismo de puertas, un mecanismo clasificador, 2 mecanismos tipo pinza, 2 tipo ascensor y un mecanismo tipo carro. Este sistema era capaz de clasificar y almacenar de forma apilada elementos tipo *stands* en dos torres y posteriormente construir una torre única. La torre junto con una pelota de tenis en su parte superior (bombilla) simbolizaba un foco de cine, cuanto mayor altura tenía el foco mayor era la puntuación obtenida.

Al igual de la estrategia de juego (ver sección 2.3.1), la manipulación de elementos de juego constituye un desarrollo específico de cada prueba de Eurobot. En este caso el desarrollo es principalmente mecánico, aunque necesita además de un desarrollo electrónico y de software. Dicho desarrollo se muestra representado en la figura 2.12.

La manipulación de elementos de juego necesita estar sincronizada con los movimientos de desplazamiento del robot, es por ello que es recomendable destinar un microcontrolador independiente para la gestión y control de los sistemas mecánicos.

## 2.4 Objetivos del desarrollo de un robot de Eurobot

Antes de nada hay que recordar que el desarrollo de un robot de Eurobot y la competición en sí misma tiene como objetivo la experimentación, el aprendizaje y compartición del conocimiento científico y de ingeniería que implica la robótica. Independientemente del resultado de la competición para un equipo, el beneficio que obtiene cada uno de los integrantes es el aprendizaje, la experiencia y el disfrute del

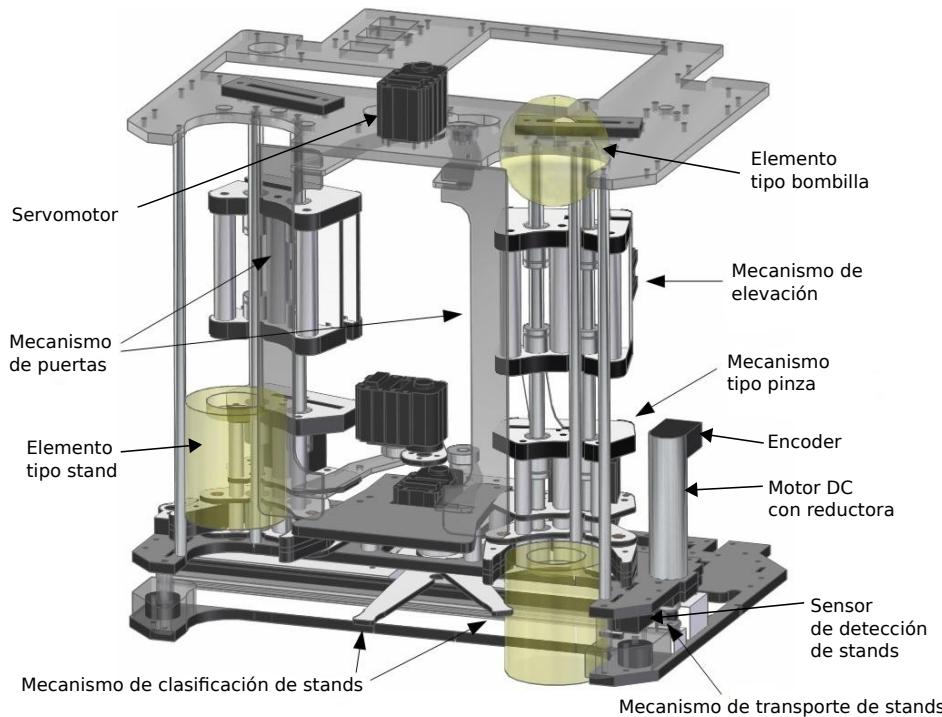


Figura 2.11: Sistema mecánico de *stands* del robot P.Tinto (Eurobot 2015) formado por varios mecanismos. Cada mecanismo esta formado por uno o varios actuadores (motores y servomotores) y sensores.

proceso de desarrollo del robot. El desarrollo de un robot de Eurobot no supone un desarrollo de un producto comercial o similar que lleve implícito una ganancia económica puesto que ni siquiera ganar la competición tiene un premio económico.

Es muy recomendable que el equipo que va a desarrollar los robots fijen unos objetivos comunes (a ser posible de forma escrita) de forma que focalicen sus esfuerzos en ellos. Unos objetivos realistas suelen tener resultados satisfactorios, consiguiendo disfrutar del proceso de desarrollo del robot y de la competición. La experiencia dice que esto nunca es así, ya que la escasez de tiempo suele implicar esfuerzos extra en el desarrollo lo cual implicar un menor disfrute. Aun así, intentar conseguir completar el 100 % de unos objetivos realistas con un 20 % menos de disfrute merece la pena.

Si no se dispone de suficiente experiencia, tiempo y recursos humanos, es recomendable empezar con un desarrollo que implique una mecánica sencilla y focalizar el desarrollo en las partes que forman lo que se ha denominado la *plataforma robótica base* (ver 2.3) y en la estrategia de juego. Una vez se tenga desarrollada una plataforma base reutilizable, el desarrollo puede centrarse más en la parte mecánica y de estrategia. En cualquier caso, aunque se desarrolle un robot que sea capaz de realizar todas las tareas del juego siempre quedarán partes que mejorar o nuevas tecnologías con las que experimentar y aprender.

Resulta muy interesante comparar el desarrollo de robot Topolino (2008) con el del robot Trompetero (2010) (ver figura 2.13). Ambos robots fueron desarrollados principalmente por las dos mismas 2 personas<sup>4</sup>: una persona dedicada a la mecánica y una dedicada a la electrónica y software. Topolino (figura 2.13a) se construyó sobre una una plataforma robótica desarrollada anteriormente<sup>5</sup>. Así se disponía casi por completo de las partes de desplazamiento y localización del robot, además de un chasis sobre el que construir la mecánica específica de la prueba de Eurobot. El desarrollo se centró en un diseño mecánico muy sencillo y de bajo coste, en la implementación de un sistema de evitación de obstáculos, en la

<sup>4</sup>Diego Salazar y Javier Baliñas

<sup>5</sup>Desarrollada por Diego Salazar, Marcelo Salazar y Sergio Arroyo

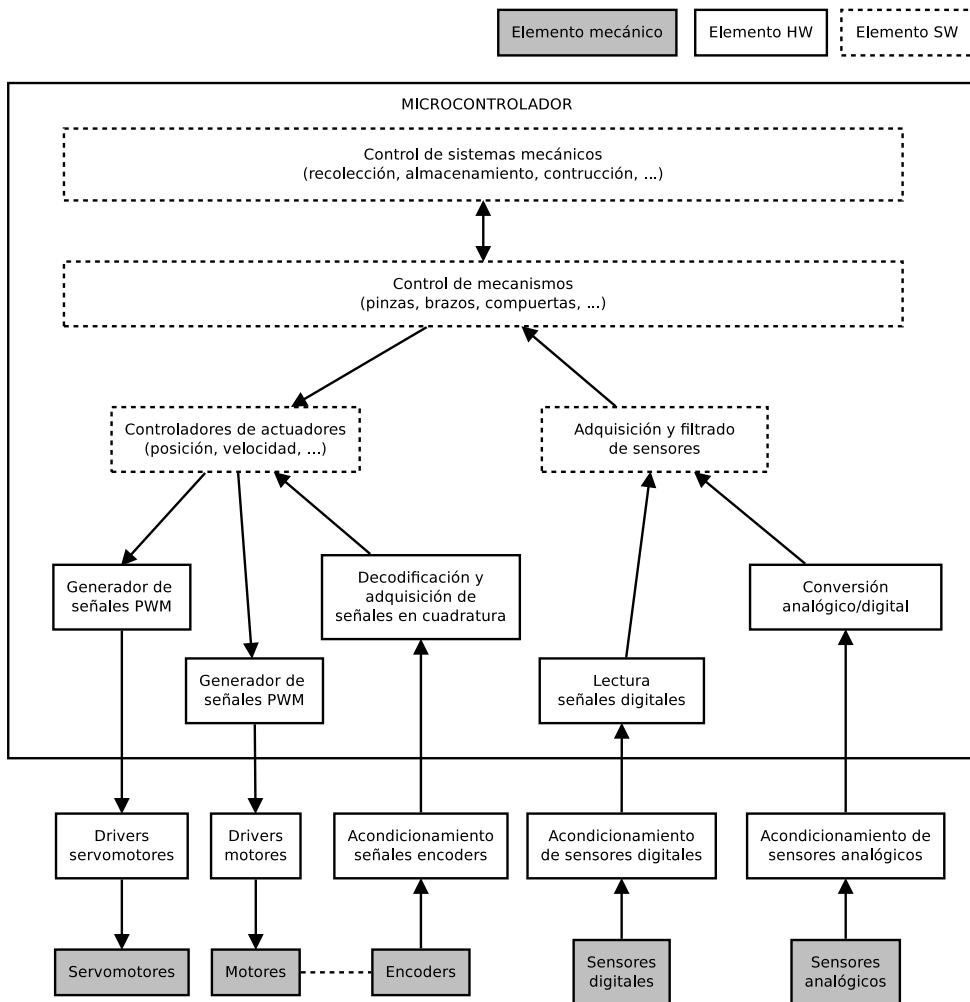
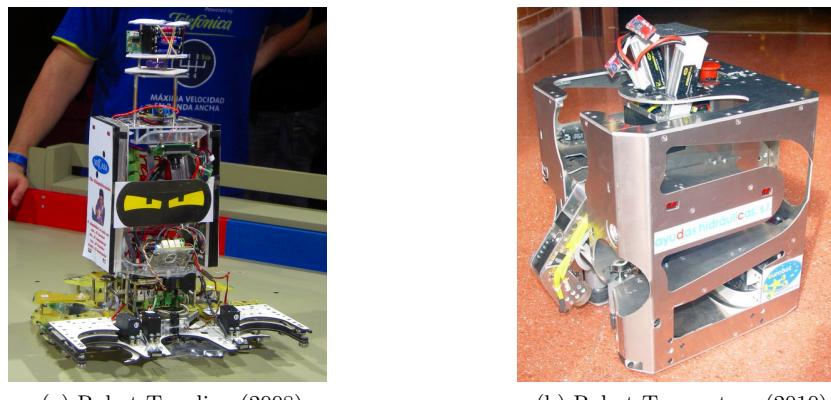


Figura 2.12: Diagrama de bloques genérico de las partes a desarrollar en un sistema mecanico destinado a la manipulación de elementos de juego.

estrategia y en la manipulación de los elementos de juego. El resultado fue muy satisfactorio, se llegó a realizar casi el 100 % de los objetivos del robot, se obtuvo una buena clasificación en la competición e incluso Topolino ganó el premio *Mejor concepto de robot*<sup>6</sup>. Y lo más importante, los desarrolladores disfrutaron del proceso hasta el final.

Trompetero (figura 2.13b) supuso un reto para los desarrolladores de Topolino, con el que se había alcanzado el techo de la tecnología con la que se construyó, y con Trompetero se quiso desarrollar un robot al nivel de los robots de primera linea de Eurobot. Trompetero partió de las librerías Aversive las cuales cuentan con partes del robot ya implementadas como el desplazamiento y localización. Aun así, Trompetero implicó el desarrollo de la mecánica, electrónica y del software del resto de partes, además de la migración de las librerías Aversive al HW basado en microcontroladores dsPIC33 y su aprendizaje. Trompetero tenía una mecánica compleja que abarcaba la manipulación de todos los elementos del juego de Eurobot 2010. En general un desarrollo muy ambicioso. Aunque el proceso de desarrollo de Trompetero supuso un gran aprendizaje y una evolución en el desarrollo de un robot de Eurobot, los imprevistos, la falta de tiempo y sueño hicieron que terminase en números rojos en términos de disfrute. Aunque se consiguieron desarrollar todas las partes del robot, estas tenían errores que no fueron solucionados a tiempo y al igual que la estrategia de juego. En consecuencia se consiguió una evolución a costa del disfrute.

<sup>6</sup>Premio otorgado a robots que implementan soluciones originales y recursos materiales modestos



(a) Robot Topolino (2008)

(b) Robot Trompetero (2010)

Figura 2.13: Robots desarrollados para Eurobot 2008 y 2010

Afortunadamente, la mayor parte del trabajo realizado en el desarrollo de Trompetero tuvo su retorno con el robot Zamorano (Eurobot 2011). Aprendida la lección, con Zamorano se volvió a la idea de una mecánica sencilla, un punto intermedio entre Topolino y Trompetero, centrando el desarrollo en la estrategia y en la optimización de la dinámica del robot. Aunque Zamorano tuvo un final amargo<sup>7</sup>, consiguió cumplir los objetivos planteados por el equipo y supuso una gran satisfacción para sus desarrolladores.

## 2.5 Desarrollo del robot de Eurobot

A la hora de afrontar el desarrollo de un robot de Eurobot lo ideal es desarrollar las partes que forman la plataforma robótica base (ver sección 2.3) antes de que las reglas de la temática del año sean publicadas. En otro caso, se recomienda focalizar los esfuerzos en desarrollar primero estas partes y una mecánica y estrategia sencilla que permitan participar en la prueba.

Una vez que se dispone de las reglas del juego lo más sencillo es empezar desde el diseño de la estrategia, siguiendo con la mecánica y por último el diseño electrónico y software. De esta forma, la estrategia es la que impondrá las restricciones de la mecánica y ésta a su vez fijará las necesidades de la electrónica, y toda ellas, las necesidades del software.

El desarrollo de un robot de Eurobot se puede dividir en las siguientes fases:

1. Fabricación del campo de juego
2. Diseño de la estrategia de juego
3. Realización de prototipos mecánicos y pruebas de concepto
4. Diseño y fabricación de la mecánica y de la electrónica
5. Montaje del robot y cableado
6. Implementación software de la funcionalidad del robot
7. Implementación software de la estrategia de juego
8. Puesta en marcha y verificación

<sup>7</sup>Habiendo ganado el partido de cuartos de final en las finales de Eurobot en Rusia, Zamorano fue descalificado por una decisión arbitral injusta.

En esencia, como ya se ha comentado un robot de Eurobot implica un desarrollo mecánico, electrónico y software. En cualquiera de estos es importante tener en cuenta el nivel de complejidad a desarrollar en cada caso sin perder de vista el objetivo final: un robot participante en Eurobot. Por ejemplo si centra el desarrollo en partes que pueden considerarse básicas, como un sensor, no se estará desarrollando un robot, si no un sensor. Se ha de valorar qué partes han de ser desarrolladas a medida y qué partes han de ser compradas. En este aspecto el tamaño de las partes juega un papel importante ya que el espacio donde han de convivir mecánica y electrónica no suele abundar. Lo más normal es que el diseño mecánico se adapte a los motores y sensores disponibles, y la electrónica se diseñe a medida de la mecánica.

### 2.5.1 Fabricación del campo de juego

Disponer de un campo de juego con todos sus elementos supone una gran ayuda en todo el proceso de desarrollo del robot, desde el diseño de la estrategia, pasando por la validación de prototipos mecánicos y hasta la fase de depuración y mejoras. El campo de juego es el entorno del robot de Eurobot y por tanto cualquier prueba ha de realizarse en él para garantizar que las condiciones van a ser las mismas que en otros campos de juego.

Es recomendable fabricar un campo de juego lo más parecido a cómo está especificado en las reglas del juego y contar con los elementos de juego reales lo antes posible.

### 2.5.2 Diseño de la estrategia de juego

Diseñar una estrategia es un proceso en el que puede participar cualquier persona y la parte más divertida ya que se cuenta con recursos ilimitados: la imaginación. Aun así es de mucha utilidad contar con los elementos del juego físicos y un campo de juego de dimensiones reales. Antes de nada es necesario y recomendable leer y estudiar bien las reglas del juego. Algunos equipos suelen organizar fines de semana en los que dedican el tiempo a estudiar las reglas y diseñar la estrategia<sup>8</sup>. La figura 2.14 muestra el campo y elementos de juegos construidos rudimentariamente durante un fin de semana *robótico* y que se utilizaron para simular las primeras estrategias de juego.

Algunos de los factores a tener en cuenta en la estrategia son los siguientes:

#### Puntuación objetivo:

Es la puntuación media a obtener por partido y con la que se estima ganar al oponente. Normalmente es un porcentaje de la puntuación máxima que puede obtener un equipo. El primer objetivo durante la competición es clasificarse entre los 16 primeros equipos. Durante la fase de clasificación cada equipo juega 5 partidos. La puntuación de corte que permite estar entre los 16 primeros varía con la temática de cada año y depende de la dificultad de las acciones a realizar por los robots.

#### Compromiso puntuación vs. complejidad

Es recomendable analizar la puntuación que supone cada elemento de juego o la tarea implicada frente a la complejidad o inconvenientes de llevarlo a cabo. Así una tarea que implique una complejidad grande y poca puntuación puede ser descartada. En cambio, aquellas tareas de baja complejidad y mucha puntuación han de tener una prioridad alta. Este tipo de estudio permite determinar qué tareas realizar para alcanzar una puntuación objetivo.

<sup>8</sup>Este tipo de actividades además ayudan a unir al equipo de Eurobot y a que sus integrantes se conozcan mejor



Figura 2.14: Elementos y campo de juego utilizado para discutir las primeras estrategias de juego de Eurobot 2015

#### **Tiempo de partido:**

El tiempo de partido es un factor importante a tener en cuenta. Al tiempo que cada robot necesita para llegar hasta los elementos de juego hay que añadir el tiempo que los robots dedicarán a evitar las colisiones entre ellos o a bloqueos desafortunados.

#### **Tiempo que implica realizar una tarea del juego:**

No es lo mismo tener que empujar un elemento para obtener puntos que tener que cogerlo, identificarlo, clasificarlo y almacenarlo. La estimación de dicho tiempo dependerá de la experiencia de los desarrolladores, pero ha de ser acotado mediante prototipos y pruebas de concepto.

#### **Acceso a los elementos de juego:**

Además hay que tener en cuenta cuánto de accesible es un elemento de juego. Por ejemplo si está muy cerca de las esquinas del campo, dependiendo de la forma de cada robot puede que el elemento no sea accesible o que sean necesarias muchas maniobras para llegar a él. Aunque en esta fase no se disponga del robot definitivo, es una buena idea construir uno de cartón de las dimensiones máximas especificadas por las reglas del juego. La simulación de los movimientos de este *robot de cartón* sobre un campo de juego de dimensiones reales permitirá comprobar si existe algún impedimento en realizar las trayectorias o movimientos pensados en la estrategia.

#### **Trayectorias y distancias hasta los elementos de juego:**

En los desplazamientos del robot la mayor parte del tiempo se destina a acelerar y a frenar. Es interesante que un robot tenga la capacidad de manipular elementos de juego cercanos y así optimizar las trayectorias. Lo ideal es que un robot realice una trayectoria continua de forma que sin detenerse sea capaz de recolectar o puntuar al pasar por las zonas de los diferentes elementos de juego como lo hacía el robot del equipo Microb en Eurobot 2010 [26]. Si esto no es posible, se han de tratar de realizar trayectorias que permitan llegar hasta los elementos del juego en

el menor desplazamiento y numero de movimientos. La optimización de las trayectorias permite ahorrar mucho tiempo y disponer de más tiempo para imprevistos del juego.

#### Caminos alternativos de estrategia:

El tiempo en un partido de Eurobot es escaso, es por ello que un robot tiene que tratar aprovecharlo. La estrategia debe de prever que no siempre el robot va a poder hacer cierta tarea en un momento dado, es por ello que en cada momento ha de preverse un camino o tarea alternativa. Sin embargo, a veces, cuando un robot se queda atrapado entre dos paredes del campo de juego y el robot oponente no queda otra que esperar a que éste libere el paso.

#### Riesgo de encontrarse al oponente:

A la hora de decidir que tareas o caminos seguir se recomienda tener en cuenta la probabilidad de encontrarse al oponente en la misma zona, de forma que la tarea se vea entorpecida o ralentizada.

#### Tareas de cada robot:

Si el equipo está formado por dos robots hay que definir que tareas realizarán cada uno de forma que resulte ventajoso para el resultado del partido.

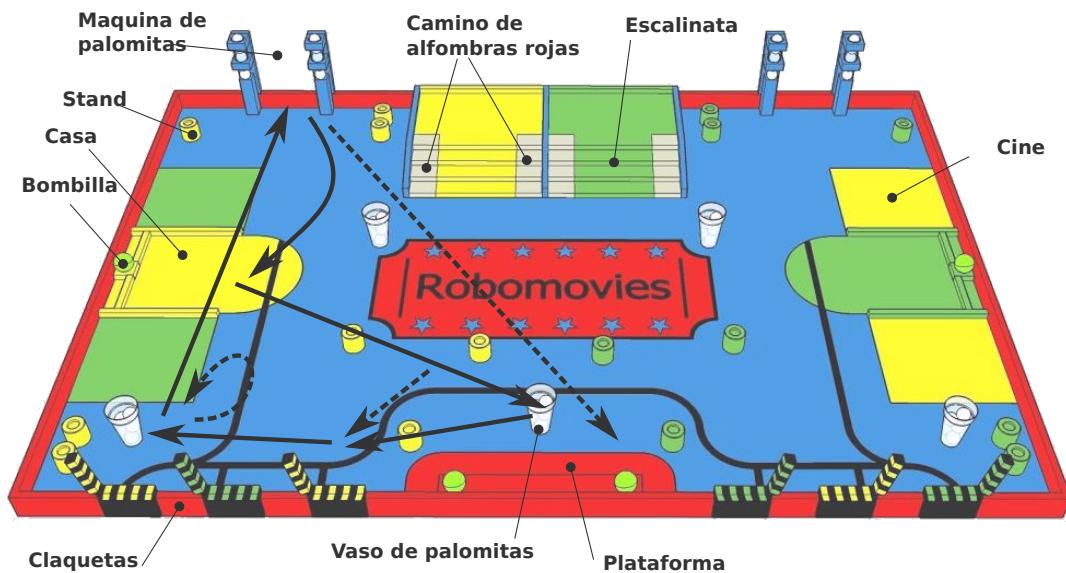
La figura 2.15 se representa la estrategia diseñada para Eurobot 2015 (*Robomovies*). Cada equipo jugaba por un color (amarillo o verde) y se permitían hasta dos robots por equipo, los cuales al principio de partido empezaban desde la zona llamada *Casa*. Cada robot podía partir con una bombilla (pelota de tenis) en su interior con la que construir un foco de cine. Se decidió crear dos robots: P.Tinto (principal) y Tirantes (secundario).

P.Tinto saldría primero al comenzar el partido y seguiría la siguiente estrategia:

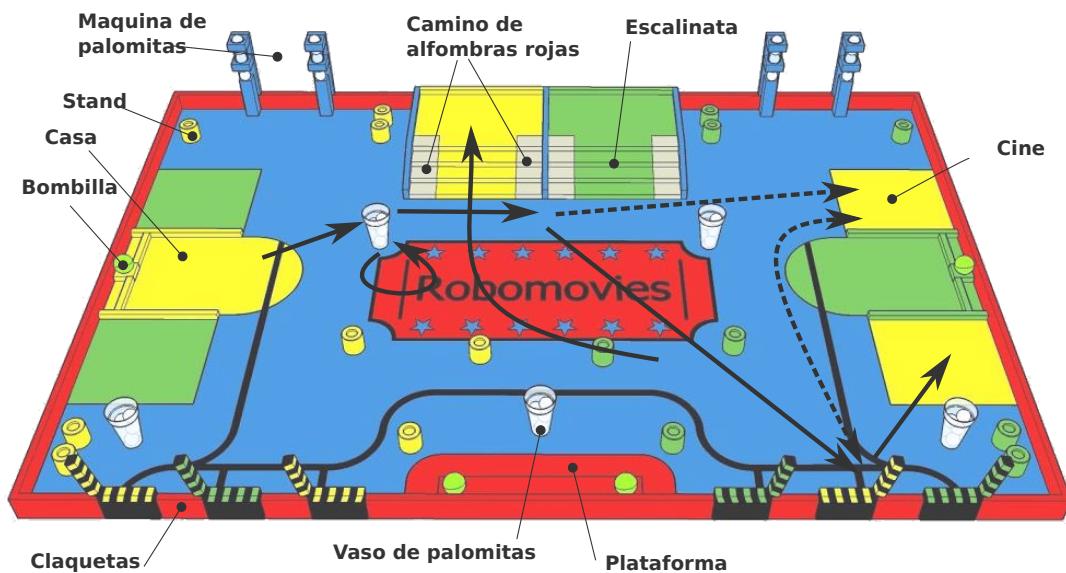
1. Se dirigiría hacia el *vaso de palomitas* del centro del campo para recolectar las palomitas de su interior y desechar el vaso. Los vasos de palomitas eran elementos compartidos por ambos equipos, además eran un número impar por lo que el vaso central tenía mucha probabilidad de ser un elemento muy disputado. En dicho camino recolectaría los dos *stands* de su color en la trayectoria.
2. Tanto si P.Tinto fuese capaz de llegar al vaso, como si no (robot oponente más rápido) a continuación se dirigiría a cerrar la *claqueta* de su color más cercana y en el camino recolectaría el *stand* que hay en la trayectoria.
3. Seguidamente P.Tinto recolectaría el *vaso de palomitas* de la esquina inferior, los dos *stands* y cerraría la *claqueta* de al lado.
4. Si P.Tinto encontrase un camino libre y el oponente no estuviese en la esquina superior derecha, se desplazaría hacia dicha zona para llenar el *vaso de palomitas* anteriormente recolectado con más palomitas y almacenar el sobrante en su interior. Posteriormente recolectaría el *stand* huérfano de la esquina superior derecha y los dos situados al lado de las escalinatas.
5. P.Tinto finalizaría volviendo a la *casa* donde dejaría el *vaso de palomitas* y las palomitas recolectadas y construiría un foco de cine con los *stands* recolectados.

Tirantes, más pequeño y ágil sería el encargado de defender el camino hacia la esquina superior derecha y hacer tareas en el otro lado del campo:

1. Tirantes comenzaría el partido en la *casa* detrás de P.Tinto y saldría después de él a recolectar el *vaso de palomitas* cercano a las escalinatas.



(a) Estrategia del robot P.Tinto



(b) Estrategia del robot Tirantes

Figura 2.15: Estrategia de los robots P.Tinto y Tirantes en Eurobot 2015. Las líneas continuas representan la estrategia principal y las líneas discontinuas caminos alternativos de estrategia.

2. Una vez capturado el vaso esperaría en esa posición a que P.Tinto hubiera terminado de recolectar los elementos de la zona interior. La máquina de palomitas era un elemento de juego común a ambos equipos, aunque lo más probable es que cada equipo fuese primero a recolectar palomitas de la máquina más cercana a su *casa*, la posición de Tirantes ayudaría a disuadir a los robots oponentes.
3. Posteriormente se dirigiría a poner las alfombras en sus caminos. Las alfombras rojas eran elementos del juego que portaban los robots desde el inicio en su interior. Si la zona de dejar las alfombras no se pudiese alcanzar Tirantes esperaría un tiempo antes de continuar con otra tarea.
4. A continuación Tirantes se dirigiría a cerrar la *claqueta* de su color del otro lado del campo. Si no pudiese se dirigiría a dejar el vaso de palomitas en el cine de arriba.
5. Seguidamente dejaría el vaso de palomitas en el *cine* de abajo o cerraría la *claqueta* que le quedaba por cerrar. Según se hubiesen desarrollado los acontecimientos.
6. Por último, Tirantes volvería a las escalinatas para subir las escaleras. Subir la escalinata era una acción de habilidad para los robots diseñados para desplazarse sobre una superficie lisa.

En esta estrategia se definieron los caminos alternativos descritos, en cualquier caso, si el robot no pudiese alcanzar alguna zona esperaría a que las condiciones fuesen favorables.

### 2.5.3 Realización de prototipos mecánicos y pruebas de concepto

Una vez más o menos clara la estrategia es el momento de investigar sobre los sistemas mecánicos a utilizar para la manipulación de los elementos de juego. No es necesario desarrollar los sistemas definitivos que llevará el robot para saber si una idea es válida. Es suficiente el desarrollo de prototipos mecánicos para validar una idea y comprobar que el concepto en el que se basa sirve para los propósitos buscados.

Antes de nada es imprescindible contar con los elementos de juego reales, para ello, será necesario adquirirlos o fabricarlos.

Esta fase de desarrollo se basa en el concepto de *prueba y error* es por ello que debería ser una fase de desarrollo muy dinámica que permite sacar conclusiones en un tiempo corto sobre el fundamento en el que luego se sustentarán los sistemas mecánicos definitivos. En un principio los prototipos suelen servir más para descartar posibles sistemas, será después de varios intentos cuando se llegará a un sistema que cumpla con los requisitos.

A la hora de desarrollar estos prototipos se recomienda plasmar las ideas sobre papel y utilizar materiales fáciles de manipular para fabricar los prototipos ideados. En función de las herramientas con las que se cuente, algunos de los materiales que se pueden utilizar son: madera, plástico, cartón y aluminio. La mayoría de esos materiales se puede trabajar utilizando herramientas de carpintería de madera o aluminio y de papelería. Una herramienta muy útil que recientemente se ha hecho muy popular son las impresoras 3D por aporte de plástico. Este tipo de herramienta es muy versátil ya que permite crear piezas de casi cualquier forma. Pero cuidado, si las piezas o piezas a imprimir para el prototipo son complejas el tiempo de diseño y fabricación de las mismas puede demorar esta fase del desarrollo del robot más de lo necesario.

La figura 2.16a muestra una de las ideas pensadas inicialmente para recolectar las palomitas de las maquinas de palomitas en Eurobot 2015. Esta se fundamentaba en utilizar una turbina para aspirar las pelotas de poliestireno (palomitas) y almacenarlas en columnas verticales y el exceso de palomitas en un

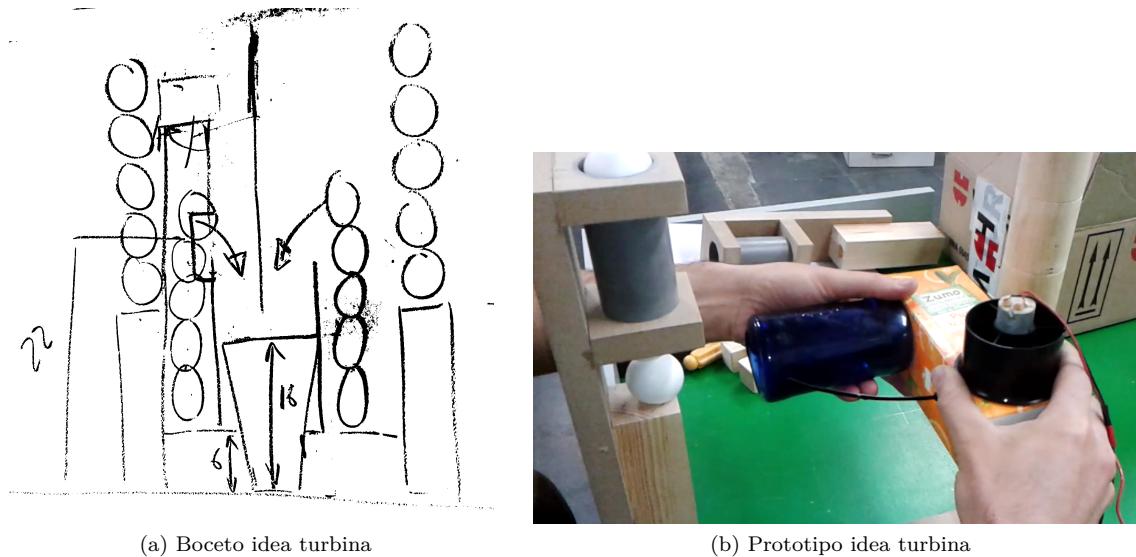


Figura 2.16: Prototipo de sistema de recolección de palomitas de las máquinas de palomitas en Eurobot 2015.

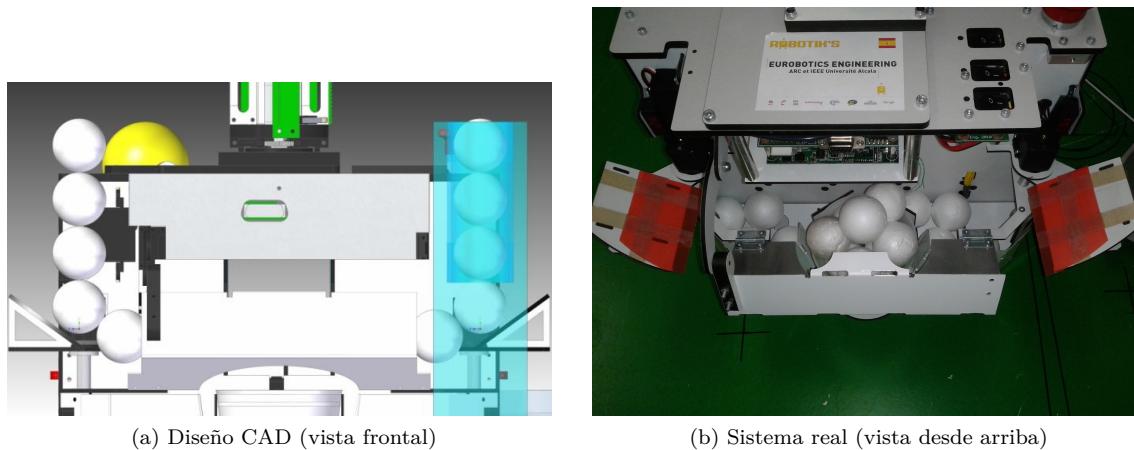


Figura 2.17: Sistema desarrollado para la recolección de palomitas de las máquinas de palomitas en Eurobot 2015.

vaso. La figura 2.16b muestra uno de los prototipos realizados para la validación de la idea, este está realizado a partir de un cartón de zumo, una botella de plástico y una turbina.

Ninguna de estas ideas fueron las que se llevaron a cabo, pero la idea final sí que contenía elementos de estas. El sistema final se muestra en la figura 2.17, dos brazos con forma de rampa empujan la primera pelota de los tubos donde se encuentran almacenadas las palomitas (máquina de palomitas). Una vez desplazada la primera pelota el resto de pelotas caen libres hacia dos compartimentos y un vaso de palomitas en el interior del robot P.Tinto.

#### 2.5.4 Diseño y fabricación de la mecánica y de la electrónica

Una vez validados los diferentes sistemas mecánicos del robot mediante prototipos o pruebas de concepto, es el momento de realizar un diseño conjunto del robot. Dependiendo del caso la electrónica necesaria para controlar la mecánica puede ser realizada a medida o mediante la integración de tarjetas desarrolladas anteriormente o comerciales. En el primer caso, la electrónica se podrá diseñar a medida de la mecánica

de forma que se situé en los huecos libres, o en el segundo caso, la electrónica deberá ser tenida en cuenta como un componente más. En cualquier caso el diseño mecánico y de la electrónica han de realizarse de forma sincronizada.

El diseño mecánico del robot no sólo ha centrarse en su funcionalidad, hay otros factores a tener en cuenta como la accesibilidad a los elementos, la facilidad de montaje y desmontaje, y el cableado de los elementos electrónicos. El robot a desarrollar no es otra cosas que un prototipo más o menos elaborado, es por ello que durante su desarrollo y en la competición de Eurobot es inevitable que existan fallos de diseño o averías. En estos casos, una mecánica modular que permita acceder de forma sencilla e independiente a las diferentes partes del robot permitirá realizar reparaciones o modificaciones con un desmontaje mínimo y en poco tiempo.

### 2.5.5 Montaje del robot y cableado

El montaje del robot y el cableado de los componentes electrónicos es una parte importante que no debe de ser subestimada. El tiempo de montaje puede llevar más o menos tiempo dependiendo de si durante la fase de diseño mecánico del robot se tuvo en cuenta como requisito la facilidad de montaje y desmontaje, y del número de piezas del que esté compuesto el robot. Así mismo la labor de cableado se simplificará si también en la fase de diseño mecánico se tuvo en cuenta el camino destinado a cables y conectores.

El cableado ha de ser claro, ordenado, jerárquico y respetando la coca de los cables utilizados. Las fijaciones de estos a la mecánica han de ser las mínimas necesarias de forma que los cables tengan cierto grado de libertad, un numero excesivo de fijaciones puede hacer que los cables se tensen o retuerzan debido al movimiento de las partes mecánicas móviles, o que en caso de reparación, la eliminación de la fijaciones supusiera un tiempo excesivo.

### 2.5.6 Implementación software de las funcionalidades del robot

Esta fase tiene como resultado un robot de Eurobot totalmente funcional, esto es, capacitado para competir en una prueba específica. Este es el punto que idealmente se debería alcanzar lo antes posible, una vez finalizado es como tener un *juguete* con el que divertirse diseñando estrategias de juego.

Hay que resaltar que una vez alcanzado este punto tan sólo el 20 % del desarrollo del robot estará terminado, el 80 % restante se corresponde al desarrollo software, la puesta en marcha y pruebas del robot el campo de juego.

La funcionalidad del robot es un nivel de abstracción que permite controlar el robot con funciones del tipo:

- ¿Está el oponente en la zona X?
- Ir a la zona de recolección del elemento X.
- Recolectar elemento X.
- Ir a la zona de almacenamiento X.
- Almacenar el elemento X.

Son varias las partes de un robot de Eurobot (ver sección 2.3) implicadas en su funcionalidad:

1. Desplazamiento y localización

2. Evitación de obstáculos
3. Comunicación entre robots
4. Manipulación de elementos de juego

Las tres primeras pertenecen a lo que se ha denominado plataforma robótica base (partes independientes de la temática de Eurobot), si estas han sido desarrolladas con anterioridad sólo necesitarán de modificaciones relacionadas con aspectos específicos de la temática del juego como pueden ser:

- Obstáculos fijos del campo de juegos
- Comandos de la comunicación entre los robots compañeros

Respecto a la última parte, está necesita ser desarrollada casi en su totalidad al depender enteramente de los sistemas mecánicos diseñados expresamente para una prueba concreta.

### 2.5.7 Implementación software de la estrategia de juego

La implementación de la estrategia supone ciertas funcionalidades del robot de Eurobot, entre ellas el desplazamiento y localización, la capacidad para manipular los elementos de juego y la detección de obstáculos y oponentes. Además, en caso de jugar con dos robots puede tener en cuenta las funcionalidades de ambos.

Un desarrollo software estratificado y modular permitirá implementar la estrategia de forma paralela al desarrollo del resto de las partes robot. La estrategia constituye la inteligencia del robot, siendo algunas de sus características deseables las siguientes:

#### **Robusted:**

Una estrategia suele tener varias ramificaciones dependiendo de las condiciones en cada momento. Cada uno de los posibles caminos han de volver siempre al camino principal de forma que no existan bloqueos o zonas muertas en la estrategia. Para ello, cada uno de los caminos implementados ha de estar probado para evitar dichas zonas muertas. Pero además, durante un partido de Eurobot se pueden dar situaciones muy diferentes dependiendo de la estrategia de los robots oponentes y del azar.

Una buena previsión de dichas situaciones evitara también bloqueos de la estrategia. Por ejemplo, la situación en la que un robot quiere llegar a una zona a la que se llega por dos caminos y uno está ocupado por el oponente y el otro por el robot secundario. Si no se ha tenido en cuenta esta situación de forma que el robot secundario libere el camino, la estrategia de partido puede quedar bloqueada hasta el final del partido y los robots no podrían sumar los puntos suficientes para ganar el partido<sup>9</sup>.

#### **Facilidad de modificación y flexibilidad:**

La estrategia dota de cierta inteligencia al robot que le permite ejecutarla de forma autónoma. Pero igual que un entrenador de baloncesto dirige a sus jugadores para seguir una estrategia concreta o adaptarla al contrincante, el equipo de desarrolladores han de poder hacer lo mismo con su robot de Eurobot. Así la implementación de la estrategia tiene que tener la flexibilidad suficiente para poder ejecutar variantes de esta o realizar modificaciones en tiempo real.

<sup>9</sup>Caso verídico por el cual los robots Crispín y Automático quedaron eliminados en cuartos de final en Eurobot 2012

**Ejecución simulada:**

La estrategia puede tener múltiples ramificaciones y variantes, y su ejecución en cada momento depende de múltiples variables, entre ellas variables dependientes del oponente. Las pruebas en el campo de juego implica reproducir cada momento y sus variables, necesita de uno o varios robots operativos y o varias personas que simulen el movimiento de robots oponentes.

Por todo ello, poder ejecutar y probar la estrategia en un entorno simulado constituye una característica de mucha utilidad. Un entorno de simulación software permite reproducir todas las variables de las que depende la estrategia en todo momento a un coste mucho menor que las pruebas realizadas en el campo de juego real con robots reales. De esta forma la verificación de la estrategia o modificaciones de ésta supone muchos menos recursos y tiempo.

### 2.5.8 Puesta en marcha y verificación

La puesta en marcha y verificación constituye la fase más importante si cabe de todas. Cada parte a desarrollar o mejorar del robot ha de ser probada en condiciones reales y verificada de forma que se asegure una repetitividad en su funcionamiento. Esto es así tanto para partes mecánicas y electrónicas, como para el funcionamiento software. Así pues esta es una fase implícita en cualquiera de las otras fases o partes a desarrollar de un robot de Eurobot.

Es recomendable realizar un desarrollo modular y estratificado desde los niveles más bajos y específicos hacia los más altos y abstractos. De esta forma cada parte desarrollada ha de contar con funciones o herramientas que permitan probar de forma independiente su funcionamiento y verificalo. Si además dichas funciones de test están accesibles durante todo el proceso de desarrollo pueden ser útiles para la depuración de funciones más abstractas o de averías relacionadas con el cableado o componentes electrónicos. Por ejemplo, en caso de fallo en un actuador tipo servomotor una función que permita mover dicho actuador a nivel de driver de señal PWM puede ayudar a verificar si el fallo es del actuador o de capas superiores al driver. O por ejemplo una función que permita comprobar el estado de los sensores digitales permitirá comprobar si hay un fallo en algún sensor o el cableado.

Por otro lado, aunque el funcionamiento del robot esté verificado es inevitable que durante los partidos surjan incompatibilidades en ciertas partes del desarrollo o funcionalidades que pueden ser mejoradas. Durante las competiciones, los partidos constituyen el mejor entorno de pruebas y son la mejor forma de pulir el funcionamiento y perfeccionar la estrategia del robot. Si es posible, se recomienda no limitarse a participar en la competición nacional del país de origen y participar en otras copas de Eurobot abiertas a participantes de otros países.

# Capítulo 3

## Plataforma robótica base

*La imaginación es más importante que el conocimiento.  
El conocimiento es limitado y la imaginación circunda  
el mundo.*

Albert Einstein, en *The Saturday Evening Post*

En el año 2010 se desarrolló una plataforma robótica base, cuyo diseño fue reutilizado y mejorado en el desarrollo de robots posteriores. Dicha plataforma incluye las funcionalidades básicas de un robot de Eurobot vistas en el capítulo 2:

- Desplazamiento y localización
- Evitación de obstáculos
- Comunicación entre robots

El desarrollo de cada una de estas funcionalidades constituye en sí una la labor de ingeniería muy interesante que implica la aplicación de fundamentos de diferentes campos de la ciencia y la ingeniería.

### 3.1 Descripción general de la plataforma robótica

La función de **desplazamiento y localización** de la plataforma robótica se implementa a partir de un *sistema de tracción diferencial* y un *sistema de posicionamiento por odometría*. Mediante la unión de estos dos sistemas implementa a su vez un *control de posición polar*, a partir del cual, se gestiona la *generación de trayectorias*.

Mecánicamente el sistema de tracción diferencial consiste en un bloque motor formado por dos motores unidos mediante una reductora mecánica a dos ruedas motrices (ver figura 3.1). Sobre el eje motriz, en sus extremos, el robot tiene dos ruedas libres conectadas a sensores tipo encoder que se utilizan para implementar el sistema de posicionamiento por odometría y, junto con el bloque motor, el control de posición del robot.

El **hardware desarrollado** de la plataforma robótica está compuesto por una tarjeta principal y varias tarjetas auxiliares que en conjunto tienen la capacidad de gestionar e implementar los sistemas mencionados anteriormente, así como los sistemas mecánicos específicos de la prueba de Eurobot, destinados a la manipulación de elementos de juego.

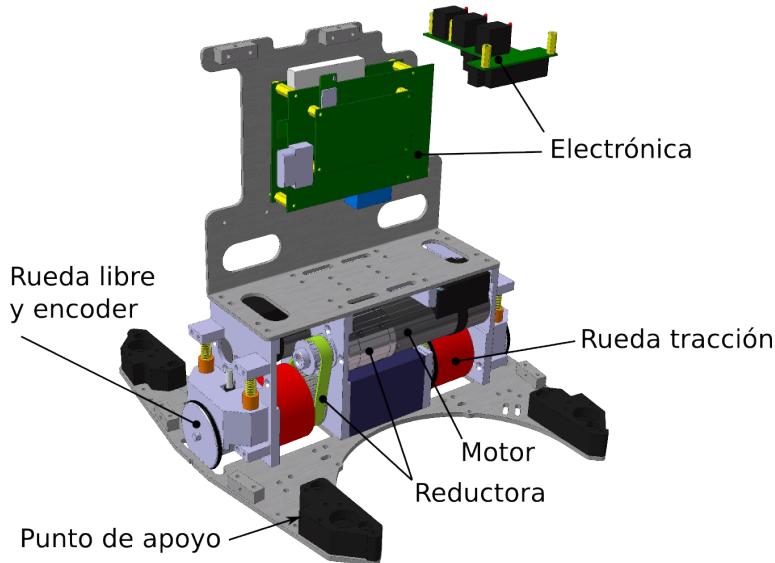


Figura 3.1: Plataforma robótica base: partes del sistema de tracción diferencial y del sistema de control de posición. Robot Zamorano (Eurobot 2011).

Respecto a la **evitación de obstáculos**, ésta se realiza a partir de sensores de distancia situados en las caras del robot (ver figura 3.2) y a partir un sistema de balizas específicamente desarrollado para la medida de la posición del oponente. Este sistema hace uso de los soportes y espacios destinados a sistemas de balizas. Concretamente, el sistema desarrollado está basado en un sensor tipo faro situado en el robot y balizas reflectantes situadas en el oponente (ver figura 3.2). El sistema de balizas es totalmente autónomo y se comunica mediante *Bluetooth* con la tarjeta principal la plataforma robótica.

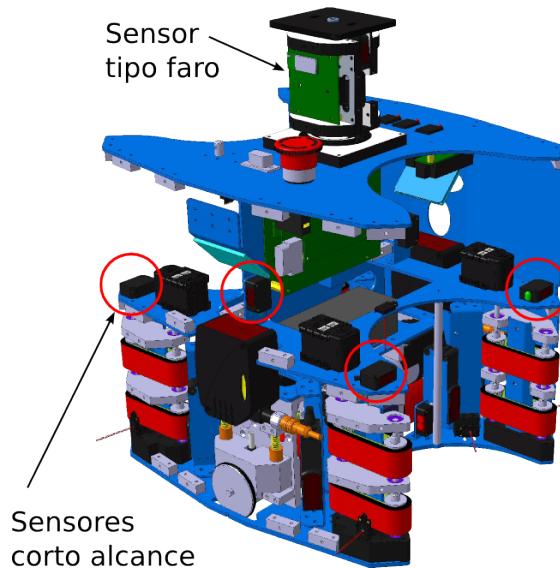


Figura 3.2: Plataforma robótica base: partes del sistema de evitación de obstáculos. Robot Zamorano (Eurobot 2011).

La **implementación software** de todos los sistemas y funcionalidades de la plataforma robótica ha sido realizada a partir de las librerías Aversive4dspic. Estas librerías han sido migradas de las librerías Aversive [4], desarrolladas para microcontroladores AVR, a microcontroladores dsPIC (utilizados por la tarjeta principal de la plataforma robot). Entre las características de las librerías Aversive y Aversive4dspic se encuentran módulos específicos para la implementación de sistemas de control y de robots como los de

Eurobot.

Por último, la plataforma robótica desarrollada permite que en caso de contar con **dos robots por equipo**, la misma arquitectura mecánica, hardware y software pueda ser utilizada en el robot principal y secundario. Y la comunicación entre ambos robots puede ser implementada mediante la interfaz *Bluetooth* de las tarjetas principales de cada robot.

## 3.2 Sistema de tracción diferencial

El sistema de tracción diferencial es uno de los más comunes en Eurobot, está compuesto por dos ruedas motrices unidas a motores mediante una reductora y varios puntos de apoyo (ver figura 3.1).

Dependiendo de la posición del eje motriz se dan 2 tipos de robots:

1. Robot con ruedas atrás: necesitan de dos puntos de apoyo delanteros.
2. Robot con ruedas en el medio: necesitan de puntos de apoyo delanteros y traseros.

Los robots con ruedas atrás suelen tener dos puntos de apoyo delanteros. En cambio, los robots con ruedas en el medio suelen tener dos puntos de apoyo delanteros y otros dos traseros. Aunque a veces en uno de los extremos sólo llevan un punto de apoyo, 3 puntos de apoyo en total.

Como se verá más adelante, interesa un robot con ruedas en el medio y que tenga una simetría total de forma que todo el peso del robot recaiga sobre el eje motriz. Además, los puntos de apoyo han de estar a la mayor distancia posible del eje motriz. Esta situación no siempre se cumple, ya que existe un compromiso con el resto de elementos mecánicos del robot.

Los puntos de apoyo utilizados normalmente son de tipo *rollon*. En caso de no disponer de espacio suficiente para ellos también es posible utilizar materiales de bajo coeficiente de rozamiento como el teflón (ver figura 3.3).

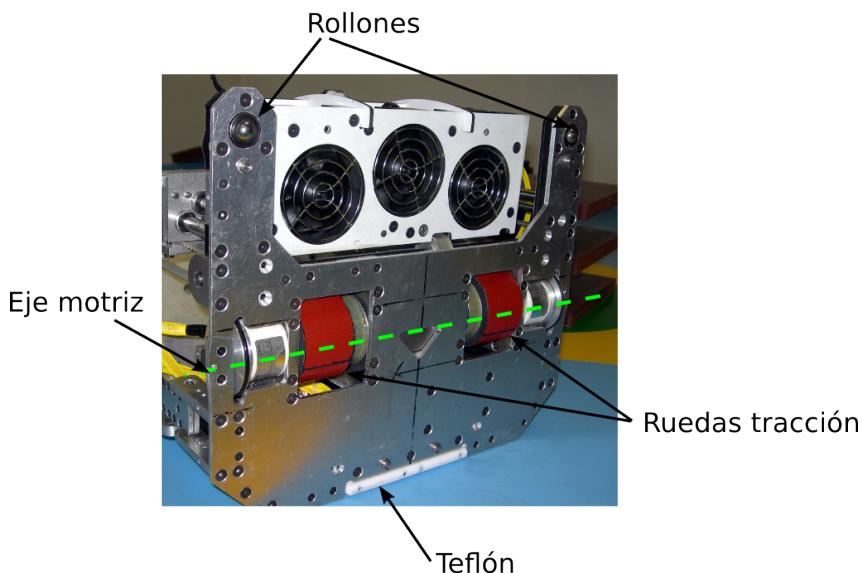


Figura 3.3: Sistema de tracción diferencial del robot Automático (Eurobot 2013) visto desde abajo.

Mecánicamente el conjunto de los motores, reductora y ruedas se denomina *bloque motor*. El bloque motor se reutiliza como una parte independiente en cada diseño de un nuevo robot de forma que se sitúa sobre la base del robot adaptándose al resto de la mecánica.

### 3.3 Sistema de posicionamiento y control de posición

La medida de la posición del robot se realiza mediante dos ruedas libres unidas a sensores tipo encoders. A partir de dicha medida de posición se implementa indirectamente el control de posición de los motores del sistema de tracción y de forma directa el sistema de posicionamiento por odometría del robot. Esta configuración difiere del esquema clásico en el que el encoder se encuentra conectado directamente al eje del motor, permite reducir los errores sistemáticos de odometría [27] y que un posible deslizamiento o derrape de las ruedas motrices no afecte al control de posición del robot.

Es posible que además los motores tengan un encoder conectado directamente su eje de forma que se implemente un doble lazo de control: un control de posición o velocidad de los motores y un control de posición del robot. Sin embargo, el uso de encoders incrementales de al menos 2000 pulsos por vuelta en las ruedas libres y el hecho de que el robot siempre está en contacto con el suelo, hace que los encoders de los motores resultarían redundantes e incrementen el procesado software y los recursos necesarios del hardware.

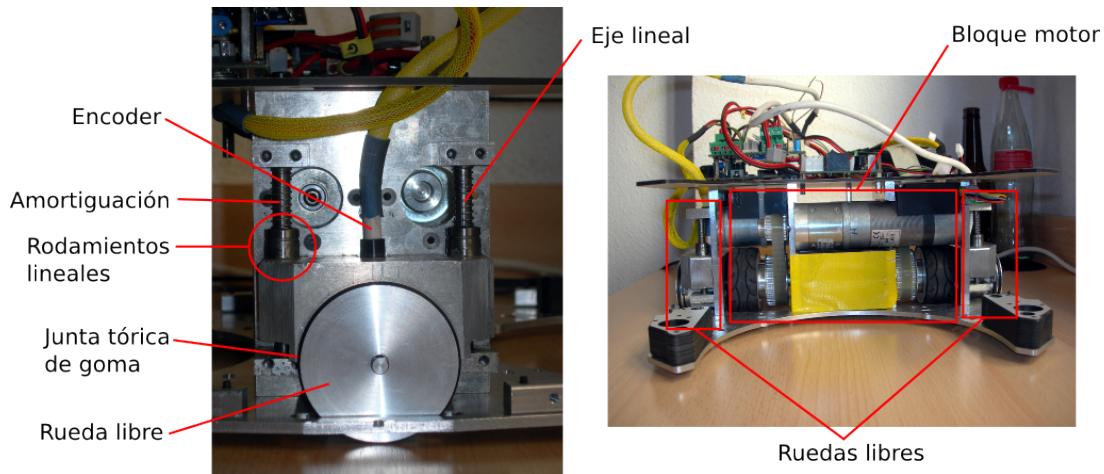


Figura 3.4: Sistema de ruedas libres lineal

Inicialmente, para el robot Trompetero (Eurobot 2010) se implementó un sistema de ruedas libres lineal como el de la figura 3.4. Este sistema está compuesto por una rueda que como neumático tiene una junta tórica de goma. Dicha rueda se encuentra conectada con un encoder incremental de dos canales en cuadratura. El conjunto rueda y encoder se encuentran fijados a la base del robot mediante un sistema de dos guías lineales con amortiguación. La amortiguación permite asegurar que ante cualquier desperfecto del campo la rueda siempre esté en contacto con el suelo. Cada guía tiene dos rodamientos lineales para disminuir el rozamiento. La implementación de este sistema mecánico es complejo y necesita de un ajuste de cierta precisión de forma que las dos guías lineales se encuentre perfectamente paralelas para que no existan rozamientos que impliquen errores no sistemáticos.

Un sistema mecánico más sencillo tanto en implementación y como en ajuste es el sistema de ruedas libres pivotante (ver figura 3.5). Este sistema fue implementado por primera vez en el robot secundario Seskapa (Eurobot 2014) dado el poco espacio disponible y posteriormente en los robots P.Tinto y Tirantes (Eurobot 2015). En este sistema, el conjunto rueda y encoder pivotan sobre un eje paralelo al eje motriz. Dicho eje se encuentra fijado a la base del robot mediante dos piezas con rodamientos de bolas separadas por un casquillo que evita que el eje se desplace. Este sistema tiene menos puntos de ajuste que el sistema lineal. Además la amortiguación del sistema es casi innecesaria debido al bajo rozamiento del eje con los rodamientos de bolas. El propio peso del conjunto rueda y encoder garantiza el contacto con el suelo. No obstante como se ve en la figura 3.5 se utilizó unas gomas elásticas para asegurar el contacto con el

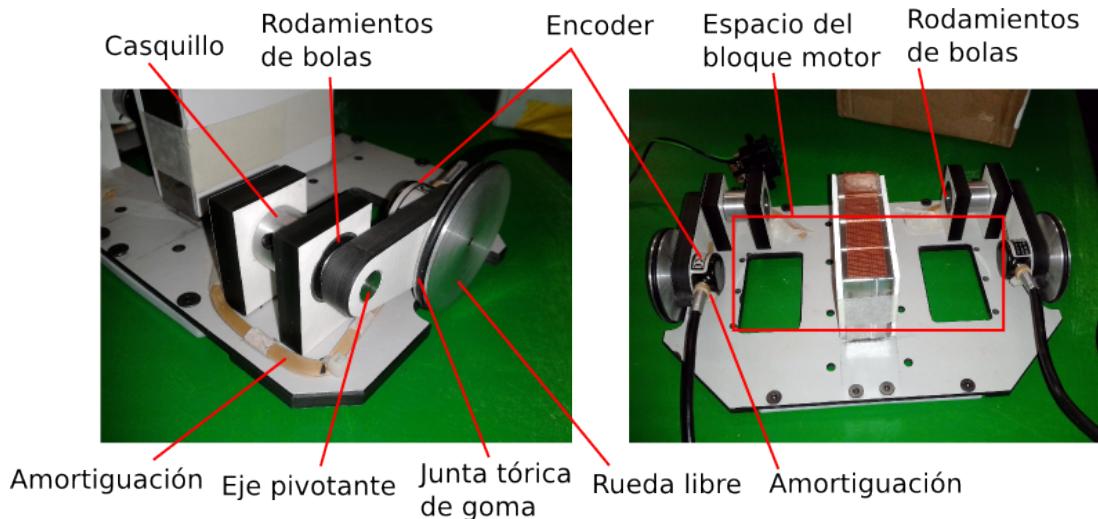


Figura 3.5: Sistema de ruedas libres pivotante

suelo. El inconveniente de este sistema se debe al propio pivote, si el suelo tiene muchas imperfecciones, al pivotar la rueda esta se desalinea respecto al eje motriz, lo cual puede generar errores de medida de posición. Este echo es despreciable frente a la simplicidad de implementación mecánica y a que el suelo del campo de juego es plano.

## 3.4 Estudio de la dinámica de robots de tracción diferencial

El estudio de la dinámica de un robot de tracción diferencial permite conocer los límites de aceleración y velocidad a partir de los cuales dimensionar la reductora y los motores que mueven el robot.

En las siguientes secciones se analizar la dinámica de un robot de tracción diferencial en desplazamientos lineales sobre el plano horizontal. El modelo dinámico para este caso es sencillo y permite deducir restricciones mecánica muy útiles a la hora de diseñar un robot de estas características.

No es objeto de este proyecto analizar el modelo dinámico del robot para desplazamientos angulares. El cálculo de dicho modelo resulta muy complejo comparado con la influencia de estos movimientos en un robot de Eurobot. Es en los desplazamientos lineales donde el robot ha de adaptar su velocidad y aceleración para evitar chocar con el oponente o alcanzar una posición del campo.

### 3.4.1 Dinámica básica en desplazamientos lineales sobre un plano horizontal

Es importante entender la dinámica de un robot de Eurobot y tenerla en cuenta durante el diseño mecánico del mismo. El movimiento más común de un robot de Eurobot son los desplazamientos entre dos puntos. Durante dichos desplazamientos el robot parte de parado, recorre una distancia y vuelve a detenerse. Así, a cada desplazamiento le corresponde una fase de aceleración, una fase de velocidad constante y otra fase de frenado.

Para el caso de un robot con ruedas atrás y dos puntos de apoyo, el modelo dinámico del robot, según describe el equipo RCVA en [28], queda representado por la figura siguiente:

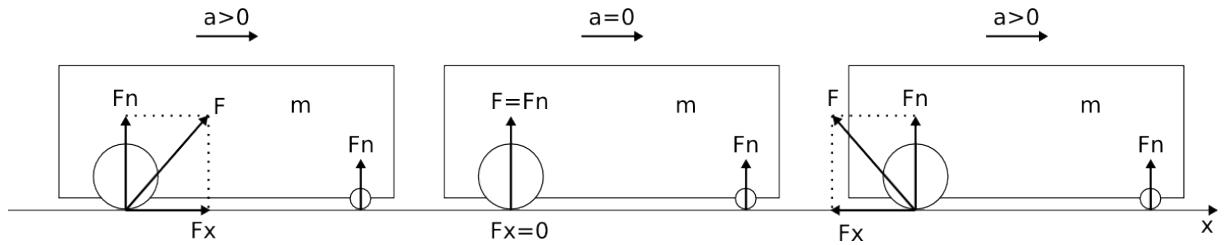


Figura 3.6: Modelo dinámico simple de un robot de tracción diferencial.

Siendo:

$F_n$ : fuerza normal o vertical del suelo (opuesta a la fuerza de apoyo debida al peso del robot)

$F_x$ : fuerza de propulsión de en la rueda

$F$ : fuerza resultante del suelo en la rueda

$a$ : aceleración del robot

$m$ : masa del robot sobre las ruedas.

Durante la fase de aceleración ( $a > 0$ ) se necesita una fuerza de propulsión  $F_x$  positiva de forma que la fuerza resultante  $F$  provoque un desplazamiento hacia delante. Una vez alcanzada la fase de velocidad constante ( $a = 0$ ), teóricamente no es necesario realizar ninguna fuerza de propulsión<sup>1</sup> y la fuerza resultante es igual a la fuerza normal sobre la rueda. En la última fase, la fuerza de tracción  $F_x$  ha de ser negativa de forma que la fuerza resultante se oponga al movimiento del robot provocando una desaceleración o frenado.

Para conseguir desplazamientos en un periodo de tiempo corto, las fases de aceleración y frenado han de durar lo menos posible. El valor de la aceleración durante esas fases va a depender por un lado del coeficiente de adherencia de las ruedas, y por otro, de la posición del centro de gravedad del robot.

### 3.4.2 Coeficiente de adherencia de las ruedas y su cálculo experimental

A partir del modelo dinámico descrito anteriormente, teniendo en cuenta que el robot dispone de dos ruedas (un motor por rueda) y a partir de la primera ley de Newton ( $F = ma$ ) se deduce que la aceleración sobre el eje horizontal del robot se corresponde con la expresión:

$$a = \frac{2F_x}{m} \quad (3.1)$$

La aceleración máxima vendrá dada por la fuerza máxima de propulsión de las ruedas  $F_{x\ max}$  a partir de la cual se produce el deslizamiento de las ruedas. Por lo tanto ha de cumplirse la condición

$$F_x < F_{x\ max} \quad (3.2)$$

siendo

$$F_{x\ max} = K_a F_n, \quad (3.3)$$

<sup>1</sup>En la práctica, dado que existen pérdidas por rozamiento  $F_x$  no es nula. Aun así la fuerza para mantener una velocidad constante es menor que la necesaria para acelerar el robot hasta esa velocidad. El efecto es el mismo experimentado al mover un armario, una vez vencida la fuerza de rozamiento inicial la fuerza necesaria para desplazarlo es mínima.

donde  $K_a$  es el coeficiente de adherencia de las ruedas contra el suelo y toma valores teóricos entre 0 y 1:

$K_a = 0$ : adherencia nula

$K_a = 1$ : adherencia teórica máxima<sup>2</sup>.

El valor de  $K_a$  depende del tipo de neumático utilizado y de la calidad y limpieza del campo de juego. Así para un neumático consistente en una junta tórica se consiguen valores de  $K_a$  inferiores a 0,5. En cambio un neumático de caucho de baja dureza (entre 20 y 30 shores) podría alcanzar valores de  $K_a$  cercanos a 1.

El valor del coeficiente de adherencia se puede medir experimentalmente mediante el método propuesto por Jacques Coulon en [28], conocido como el *método del cubo*.

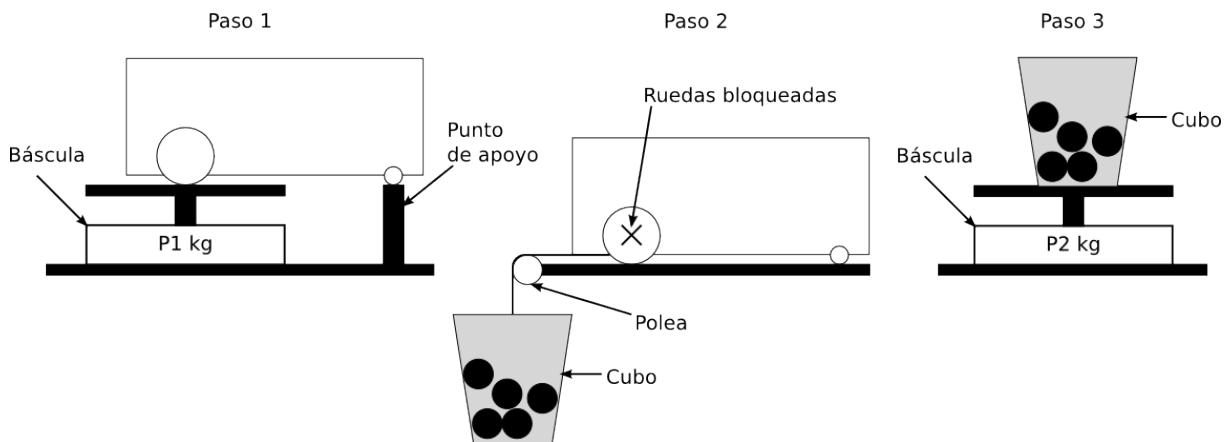


Figura 3.7: Medodo de medida del coeficiente de adherencia de las ruedas.

Dicho método se encuentra representado en la figura 3.7 y consta de tres pasos:

1. Apoyar las ruedas del robot sobre una bascula manteniendo el robot horizontal. Para ello, ayudarse de un soporte sobre los puntos de apoyo del robot. El peso  $P1$  medido por la bascula se corresponde con la fuerza con la que las ruedas apoyan en el suelo.
2. Bloquear las ruedas del robot<sup>3</sup> y fijar a la parte más baja y cercana al eje motriz una cuerda atada a un cubo. Hacer pasar la cuerda por una polea para disminuir el rozamiento y añadir peso al cubo hasta que las ruedas comiencen a deslizar.
3. Medir el peso  $P2$  del cubo, el cual se corresponde con la fuerza de tracción máxima en el límite de deslizamiento.

El coeficiente de adherencia viene dado por las medidas realizadas según la expresión

$$K_a = \frac{P2}{P1}. \quad (3.4)$$

Notar que el valor de  $K_a$  es independiente del peso del robot.

La validez de este método se ha comprobado en la práctica. El método se aplicó a robots de diferente peso y de ruedas idénticas obteniendo los resultados de la tabla siguiente:

<sup>2</sup>En la práctica el coeficiente de adherencia puede superar valores de uno, por ejemplo si existe una carga aerodinámica extra como ocurre en los coches de carreras de la F1.

<sup>3</sup>El bloqueo ha de ser un bloqueo mecánico, en caso de bloquear las ruedas manteniendo la consigna de posición del robot el motor puede resultar gravemente dañado.

Tabla 3.1: Resultados de la medida del coeficiente de adherencia de las ruedas. La obtención de un valor mayor que 1,0 para el caso del robot Zamorano se debe a la tolerancia de los objetos utilizados en el test del cubo. En ambos casos el valor se puede aproximar a la unidad.

Robot	$P1$	$P2$	$K_a$
Zamorano (2011)	14,5	15,6	1,076
Automático (2012)	17,6	17,3	0,983

Las ruedas de ambos robots se muestra en la figura 3.8. Están construidas a partir de ruedas de coche de radio control de espuma, 60mm de diámetro y 45mm de ancho. Además se les ha añadido una capa de silicona para mejorar la adherencia. Este tipo de ruedas ha sido utilizado en todos los robots desarrollados desde el año 2011 dado su alto coeficiente de adherencia.

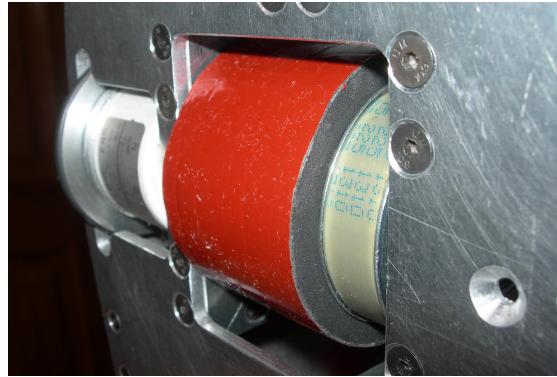


Figura 3.8: Ruedas utilizadas en los robots

### 3.4.3 Determinación de la aceleración máxima en las fases de aceleración y frenado

La aceleración máxima durante las fases de aceleración o frenado depende de dos condiciones. La primera es la condición para la cual las ruedas derrapen o deslicen. Y la segunda, en caso de robots con ruedas atrás, es la condición para la cual el robot pierde contacto con el suelo en su parte delantera (como un caballo que se pone sobre sus patas trasera), pudiendo volcar hacia atrás.

Por otro lado, el valor de la aceleración máxima para cada condición depende del coeficiente de adherencia de las ruedas ( $K_a$ ) y la posición del centro de gravedad del robot, respecto al eje motriz y los puntos de apoyo.

A continuación describe como determinar el valor máximo de aceleración para los siguientes tipos de robot:

1. Robot con ruedas atrás
2. Robot con ruedas en el medio

El desarrollo completo puede encontrarse en [28]. Aquí sólo se presentan los desarrollos que se han considerado importantes para el entendimiento la dinámica del robot y los resultados finales. Además, este trabajo añade el resultado para el caso de un robot con ruedas en el medio asimétrico, no estudiado en [28].

### 3.4.3.1 Robots con ruedas atrás

Se parte del modelo dinámico del robot representado por la siguiente figura:

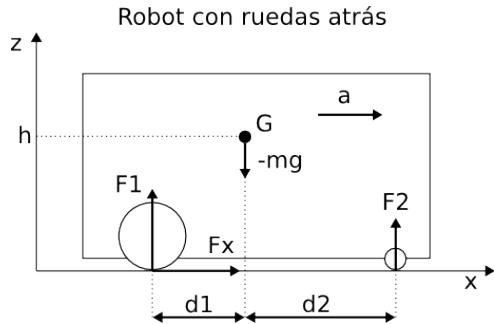


Figura 3.9: Modelo dinámico en fase de aceleración y frenado para robots con ruedas atrás.

Donde:

$G$ : centro de gravedad del robot.

$F_x$ : fuerza de propulsión de en la rueda.

$F1$ : fuerza normal del suelo en la rueda.

$F2$ : fuerza normal del suelo en el punto delantero.

$a$ : aceleración del robot

$-mg$ : fuerza de la gravedad aplicada en el centro de gravedad.

$d1$ : distancia entre el centro de gravedad y el eje motriz.

$d2$ : distancia entre el centro de gravedad y el punto de apoyo delantero.

$d$ : distancia entre el eje motriz y un punto de apoyo ( $d = d1 + d2$ ).

$h$ : altura del centro de gravedad del robot.

Las ecuaciones que garantiza que el sistema esta en equilibrio son las siguientes:

Suma de fuerzas verticales

$$2F1 + 2F2 - mg = 0 \quad (3.5)$$

Suma de fuerzas horizontales

$$2F_x = ma \quad (3.6)$$

Suma de torsiones aplicadas al punto G

$$F2d2 - F1d1 + F_x h = 0 \quad (3.7)$$

A partir las ecuaciones (3.5), (3.6) y (3.7) , se obtienen las ecuaciones que representan la distribución de pesos en las fases de aceleración y frenado:

$$(2F1)/mg = d2/d + (a/g)(h/d) \quad (3.8)$$

$$(2F2)/mg = d1/d - (a/g)(h/d) \quad (3.9)$$

Notar que el término  $(2F1)/mg$  representa la distribución de pesos en el eje motriz, mientras que el término  $(2F2)/mg$  representa el reparto de pesos en la parte delantera. Los términos  $d2/d$  y  $d1/d$  representan las distribuciones de pesos en estático, cuando la aceleración es nula ( $a = 0$ ). Y el término  $(a/g)(h/d)$  representa la transferencia de masa del robot durante la aceleración o frenado.

Así pues, en fase de aceleración ( $a > 0$ ) la transferencia de masa se produce hacia la parte trasera del robot por lo que el robot avanza y tiende a hacer *el caballito*. Mientras que en la fase de frenado ( $a < 0$ ) la transferencia de masa se produce hacia la parte delantera de forma que las ruedas tienen a deslizar o derrapar sobre el suelo.

Se deducen algunas conclusiones al respecto:

- Interesa tener el máximo apoyo en el eje motriz aumentando el término  $(2F1)/mg$  o el término estático  $d2/d$ , de forma que el centro de gravedad esté lo más cerca posible del eje motriz.
- Cuando el término de transferencia de masa dinámica  $(a/g)(h/d)$  cambia de signo debido a la aceleración, se consiguen una buena fase de aceleración pero una fase peor de frenado.
- Como los robots en un partido no hacer otra cosas que acelerar y frenar, interesa reducir lo máximo la transferencia de masa, la cual depende únicamente de la relación  $(h/d)$ . Por lo tanto interesa bajar el centro de gravedad lo máximo posible.

Continuando con el cálculo, la condición para no derrapar en la fase de aceleración viene dada por la ecuación (3.3) expresada como

$$F_x < K_a F1 \quad (3.10)$$

y para la fase de frenado como

$$|F_x| < K_a F1. \quad (3.11)$$

Por otro lado, la condición para que el robot no haga *el caballito* durante la fase de aceleración es

$$F2 > 0 \quad (3.12)$$

y para la fase de frenado es

$$F1 > 0 \quad (3.13)$$

A partir de las ecuaciones (3.8), (3.9), (3.6) y las condiciones mencionadas anteriormente se obtienen las aceleraciones máximas para cada fase y condición.

Fase de aceleración ( $a > 0$ ):

$$a_{max1} = g \frac{K_a d2}{d - (K_a h)} \quad (3.14)$$

$$a_{max2} = g \frac{d1}{h} \quad (3.15)$$

Fase de frenado ( $a < 0$ ):

$$a_{max1} = g \frac{K_a d2}{d + (K_a h)} \quad (3.16)$$

$$a_{max2} = g \frac{d2}{h} \quad (3.17)$$

Siendo  $a_{max1}$  la aceleración máxima para no derrapar y  $a_{max2}$  la aceleración máxima para no hacer el *caballito*. Para cada fase, la aceleración máxima vendrá dada por el valor mínimo de estas:

$$a_{max} = \min\{a_{max1}, a_{max2}\}. \quad (3.18)$$

Por ejemplo, la figura 3.10 representa el caso de dos robots iguales excepto por la altura de su centro de gravedad.

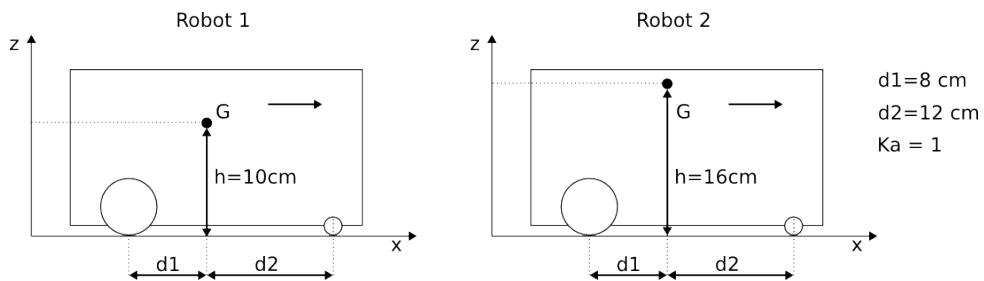


Figura 3.10: Ejemplo de cálculo de la aceleración máxima en las fases de aceleración y frenado

Las aceleraciones máximas calculadas para estos robots se muestran en la tabla 3.2.

Tabla 3.2: Ejemplo de aceleraciones máximas en función del centro de gravedad

	Aceleración R1	Frenado R1	Aceleración R2	Frenado R2
$a_{max1}$	$1,2g$	$0,4g$	$3,0g$	$0,3g$
$a_{max2}$	$0,8g$	$1,2g$	$0,5g$	$0,75g$
$a_{max}$	$0,8g$ ( $8 m/s$ )	$0,4g$ ( $4 m/s$ )	$0,5g$ ( $5 m/s$ )	$0,3g$ ( $3 m/s$ )

Observar que estos cálculos sólo se refieren a desplazamientos de translación y no a giros, permitiendo maximizar la aceleración. Que el robot sea capaz de alcanzar dicha aceleración va a depender de la elección de los motores y su reductora.

Por otro lado, hay que tener en cuenta que estos cálculos dependen únicamente del centro de gravedad del robot y del coeficiente de adherencia de las ruedas. El peso del robot no interviene.

### 3.4.3.2 Robots con ruedas en el medio

Un robot que tiene las ruedas en el medio puede ser que tenga su centro de gravedad exactamente sobre el eje motriz, de forma que el robot es simétrico. Si el centro de gravedad está desplazado se habla de un robot asimétrico.

Los modelos dinámicos para estos casos se corresponden con la siguiente figura:

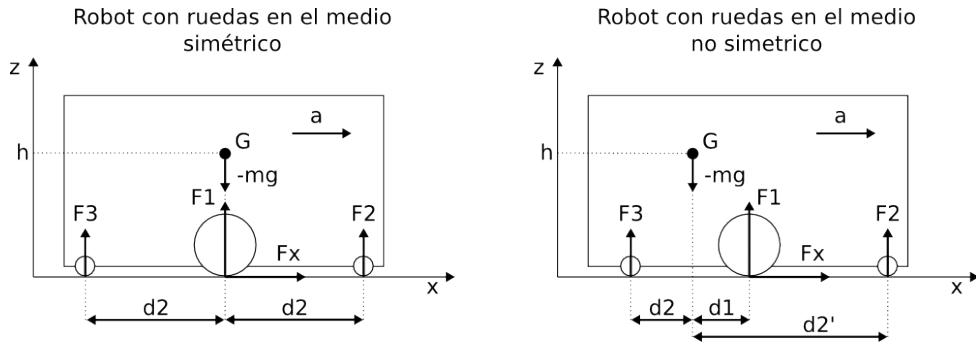


Figura 3.11: Modelo dinámico fase de aceleración y frenado para robots con ruedas en el medio.

Donde:

$G$ : centro de gravedad del robot.

$F_x$ : fuerza de propulsión de en la rueda.

$F_1$ : fuerza normal del suelo en la rueda.

$F_2$ : fuerza normal del suelo en el punto delantero.

$F_3$ : fuerza normal del suelo en el punto de apoyo trasero.

$a$ : aceleración del robot

$-mg$ : fuerza de la gravedad aplicada en el centro de gravedad.

$d_1$ : distancia entre el centro de gravedad y el eje motriz.

$d_2$ : distancia entre el centro de gravedad y el punto de apoyo trasero.

$d_2'$ : distancia entre el centro de gravedad y el punto de apoyo delantero.

$h$ : altura del centro de gravedad del robot.

Para el caso del robot simétrico de la figura 3.9 y siguiendo un análisis igual al realizado para robots con ruedas atrás, se obtienen las ecuaciones que definen las aceleraciones máximas. Debido a que el robot tiene 4 puntos de apoyo no es posible que éste haga *el caballito*, por lo que la aceleración máxima vendrá dada únicamente para la condición de derrape de las ruedas y es la misma para las fases de aceleración y frenado dada la simetría del robot:

$$a_{max} = g \frac{K_a}{1 + K_a (h/d2)} \quad (3.19)$$

Por otro lado, en el caso del robot asimétrico, el centro de gravedad cae entre el eje motriz y los puntos de apoyo de uno de los extremos del robot. Al igual que en el caso anterior la condición de hacer *el caballito* no existe al tener el robot 4 puntos de apoyo. Sin embargo, la aceleración máxima es distinta para la fase de aceleración y la de frenado debido a la asimetría del robot.

Fase de aceleración ( $a > 0$ ):

$$a_{max} = g \frac{K_a}{(d1 + d2)/d2 + K_a (h/d2)} \quad (3.20)$$

Fase de frenado ( $a < 0$ ):

$$a_{max} = g \frac{K_a}{(d2' - d1)/d2' + K_a (h/d2')} \quad (3.21)$$

Se comprueba que si  $d1 = 0$ , el modelo se corresponde con el de un robot simétrico y las ecuaciones (3.20) y (3.21) coinciden con la ecuación (3.19).

### 3.4.3.3 Medida del centro de gravedad del robot

Se ha visto que para los cálculos de la aceleración máxima del robot es necesario conocer la posición del centro de gravedad del mismo. Su valor puede ser medido fácilmente utilizando una regla y una barra lo suficientemente larga. Por ejemplo para medir la altura del centro de gravedad se procedería a apoyar el robot sobre la barra por una de sus caras como muestra la figura 3.12 hasta dar con una posición en la que el robot se encuentre en equilibrio. Una vez obtenida la posición de equilibrio se mide la distancia que hay de la base del robot al centro de la barra, la cual se corresponde con la altura del centro de gravedad.

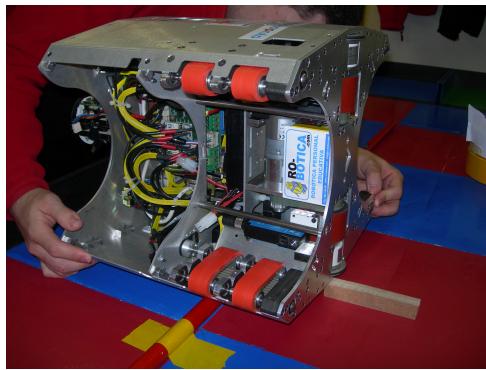


Figura 3.12: Ejemplo de medida de la altura del centro de gravedad del robot

La operación se repite sobre la base del robot situando la barra paralela al eje motriz, de forma que se determina la distancia del centro de gravedad a dicho eje. Y por último, se puede realizar la medida con la barra situada perpendicularmente al eje motriz con el fin de comprobar que el robot es simétrico en dicho eje y que el peso soportado por cada rueda es el mismo.

### 3.4.4 Perfil de velocidad trapezoidal

Como ya se ha visto, el desplazamiento de un robot sobre un plano horizontal se divide en una fase de aceleración, otra de frenado y una fase intermedia de velocidad constante. Dichas fases se suelen implementar mediante un perfil trapezoidal de velocidad como el representado en la figura 3.13.

Como se sabe, la optimización de dicho perfil pasa por la determinación de las aceleraciones y velocidad máximas. La aceleración máxima se corresponde con aquella para la que el robot se encuentre al límite del deslizamiento o de hacer *el caballito* y se calcula de la forma que se ha descrito en la sección anterior. En la práctica, nunca se trabaja con la aceleración máxima teórica y si no que se utiliza una aceleración de seguridad que suele ser el 80 % de la teórica:

$$a = 0,8 a_{max} \quad (3.22)$$

Respecto a la velocidad máxima, dependerá en última medida del motor y la reductora utilizada. No obstante, dadas las dimensiones del campo y que los robots han de ser capaces de evitar colisiones entre

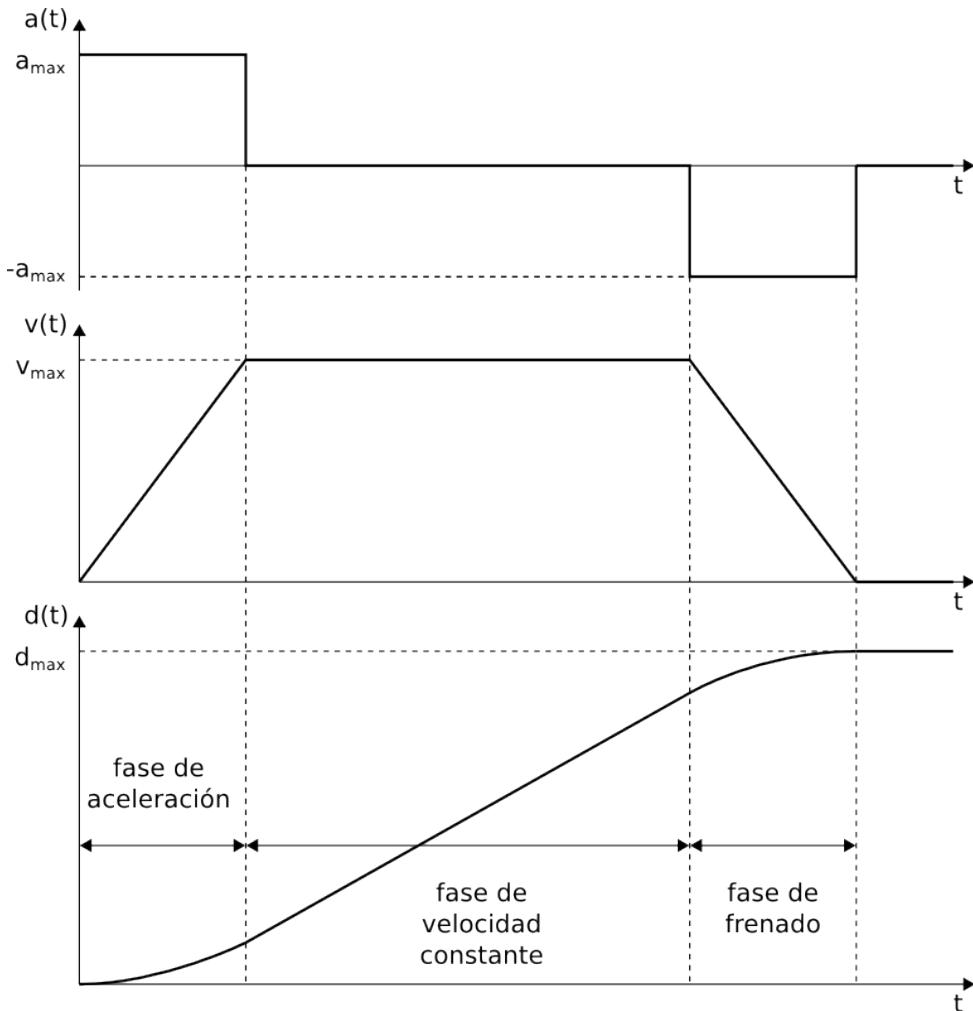


Figura 3.13: Fases de un perfil trapezoidal de velocidad: aceleración, velocidad y distancia recorrida

ellos, una velocidad de  $1m/s$  suele ser un buen valor de compromiso.

Resulta interesante el análisis que hacen en [28] sobre el valor de la velocidad máxima. En este análisis se comparan dos perfiles trapezoidales de distinta velocidad ( $1m/s$  y  $1,4m/s$ ), de igual aceleración ( $a = 2 m/s^2$ ) y para una misma distancia recorrida ( $1m/s$ ). El resultado es que el perfil con velocidad de  $1,4m/s$  tarda sólo  $0,1s$  menos en recorrer la distancia de  $1m$ . Un tiempo que es despreciable. Esto es debido a que el tiempo de la fase de velocidad constante del perfil de  $1,4m/s$  es muy pequeño respecto al tiempo que se invierte en acelerar y frenar. De esto se deduce que para distancias cortas no es tan importante la velocidad como la aceleración del robot.

Por último, respecto a la gráfica de distancia del perfil trapezoidal, se observa una evolución parabólica durante las fases de aceleración y frenado que permite un movimiento suave del robot. Como se verá en la sección 3.5, esta curva representa una consigna de distancia para el sistema de control de posición.

### 3.4.5 Dimensionamiento de la reductora

Como se sabe, las ruedas del eje motriz se encuentran conectadas al motor por medio de una reductora. El modelo mecánico de dicho sistema se muestra en la figura 3.14.

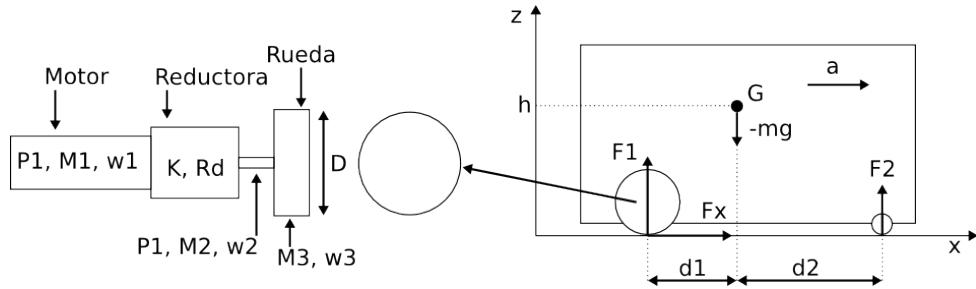


Figura 3.14: Modelo mecánico de uno de los motores del sistema de tracción diferencial.

Donde:

$\omega_i$ : velocidades angulares ( $rad/s$ )

$M_i$ : par o momentos de fuerza ( $Nm$ )

$P_i$ : potencias mecánicas ( $Nm/(rad/s)$ )

$K$ : relación de reducción de la reductora

$Rd$ : rendimiento de la reductora

$D$ : diámetro de la rueda

La potencia mecánica en el motor  $P1$  y en el eje de salida de la reductora  $P2$  viene dada como:

$$P1 = M1 \omega_1 \quad (3.23)$$

$$P2 = M2 \omega_2 \quad (3.24)$$

Dichas potencias se relacionan mediante la reductora de la siguiente manera:

$$\omega_2 = \frac{\omega_1}{K} \quad (3.25)$$

$$M2 = M1 Rd K \quad (3.26)$$

$$P2 = P1 Rd \quad (3.27)$$

Observar que la relación entre potencias en (3.27) no depende de  $K$ . Si  $K$  aumenta el par de la reductora  $M2$  aumenta pero la velocidad  $w2$  disminuye.

Por otro lado, del modelo del motor de corriente continua se sabe que la tensión de armadura del motor  $U$  y el par motor  $M$  vienen dados por las expresiones:

$$U = r i + K_v \omega \quad (3.28)$$

$$M = K_m i \quad (3.29)$$

Siendo:

$r$ : resistencia de la armadura ( $\Omega$ )

$i$ : corriente de la armadura (A)

$K_v$  : constante de fuerza electromecánica ( $V/(rad/s)$ )

$K_m$ : constante de par motor (Nm/A)

A la hora de dimensionar la reductora se busca determinar el valor de reducción ( $K$ ) que genere un par suficiente en la rueda para vencer las fuerzas aplicadas en la misma que se oponen al movimiento.

No obstante existe una condición más restrictiva a la hora de dimensionar la reductora. Es el caso en el que las ruedas queden bloqueadas debido a que el robot se encuentra con un obstáculo imprevisto. Por ejemplo una pared debido a un error de posicionamiento, o que haya un objeto inesperado entre el robot y la pared. En ese caso el controlador de posición reaccionará aumentando la tensión del motor hasta su valor máximo  $U_{max}$  (tensión de saturación  $v_{sat}$ ) y se corre el riesgo de que el motor se queme. Este caso es analizado en [28] a partir de modelo térmico del motor. El análisis determina que en caso de bloqueo de una rueda la vida útil del motor es de 2,3s, a menos que el driver del motor sirva de fusible y se rompa antes o que exista un mecanismo de protección.

Teniendo en cuenta este caso, un dimensionamiento más correcto de la reductora es aquel que permita que las ruedas derrapen antes de que la tensión del motor alcance su valor máximo.

Por un lado se calcula el par en el motor  $M1$  necesario para producir deslizamiento a partir del par en la rueda  $M3$ , el cual depende de a su vez la fuerza de propulsión máxima de las ruedas  $F_{x max}$ .

$$M1 = \frac{M3}{K Rd} \quad (3.30)$$

$$M3 = F_{x max} \frac{D}{2} \quad (3.31)$$

$$F_{x max} = \frac{K_a F1}{2} \quad (3.32)$$

La fuerza aplicada sobre las ruedas  $F1$  depende de la distribución de pesos sobre el eje motriz y se corresponde con el peso  $P1$  medido durante la medida experimental del coeficiente de adherencia  $K_a$ . De forma general  $F1$  se calcula como:

$$F1 = mg \frac{d2}{d} \quad (3.33)$$

El par en el motor  $M1$  resulta:

$$M1 = \frac{K_a mg (d2/d)}{K Rd D} \quad (3.34)$$

Por otro lado, a partir de las ecuaciones del modelo del motor (3.28) y (3.29), se obtiene el par máximo que puede dar el motor a la salida de la reductora  $M2$  para la situación de bloqueo ( $\omega = 0$ ):

$$U = U_{max} = v_{sat} \quad (3.35)$$

$$i|_{\omega=0} = \frac{v_{sat}}{r} \quad (3.36)$$

$$M2 = K_m i = K_m \frac{v_{sat}}{r} \quad (3.37)$$

La condición que garantiza el deslizamiento de las ruedas en una situación de bloqueo es que el par máximo del motor sea mayor que el par en el límite de deslizamiento de la rueda. Para asegurar dicha condición se añade un margen de un 20 % de forma que el deslizamiento sea apreciable:

$$M2 >= 1,2 M1 \quad (3.38)$$

El valor mínimo de la reductora  $K_{min}$  se obtiene al sustituir (3.30) y (3.37) en (3.38) y despejar  $K$ :

$$K_{min} = 1,2 \frac{K_a mg (d2/d) Dr}{4 K_m Rd v_{sat}} \quad (3.39)$$

Notar que  $K_{min}$  aumenta si los términos  $K_a F1$ ,  $D$  o  $r$  aumentan, o  $K_m$ ,  $U_{max}$  disminuye.

Por ejemplo, el sistema de tracción diferencial de la plataforma desatollada utiliza unos motores *brushless* Dunkermotoren BG40x50 alimentados a 24V de 30W. Aunque no se trata de un motor de corriente continua se sabe por [29] que se puede modelizar como tal y que su funcionamiento queda representado por las mismas curvas del motor de continua como muestra la figura 3.15

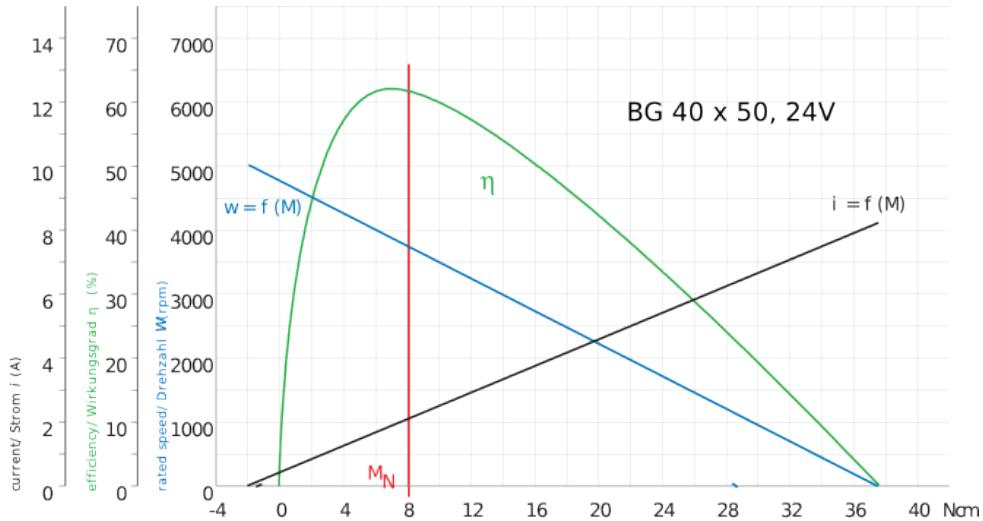


Figura 3.15: Curvas del motor Dunkermotoren BG40x50 24V dada por el fabricante.

Data / Leistungsdaten		BG 40x50
Rated voltage/ Nennspannung		24 VDC
Continuous rated speed/ Nendrehzahl	rpm*)	3640
Continuous rated torque/ Nendrehmoment	Ncm*)	8,0 (9.6***)
Continuous current/ Nennstrom	A*)	2.1
Starting torque/ Anlaufmoment	Ncm**)	37.4
Starting current/ Anlaufstrom	A**)	8.4
Rotor inertia/ Trägheitsmoment	gcm <sup>2</sup>	64
Weight of motor/ Motorgewicht	kg	0.6

\*) $\Delta\vartheta_W = 100$  K; \*\*)  $\vartheta_R = 20$  °C; \*\*\*) Depends on heat dissipation from the motor

Figura 3.16: Parámetros del motor Dunkermotoren BG40x50 24V dadas por el fabricante.

A partir de los parámetros y curvas dadas por el fabricante (ver figura 3.16 y 3.15) se deducen las variables necesarias para el cálculo de la reductora. La resistencia equivalente se obtiene de (3.28) a partir de la tensión nominal  $U_n$  y la corriente de arranque  $i_o$  como

$$r = \frac{U_n}{i_o} = \frac{24V}{8,4A} = 2,86\Omega$$

Por otro lado la constante mecánica  $K_m$  se obtiene a partir de la pendiente de la recta  $i = f(M)$  de la figura 3.15 los puntos  $(M_n, i_n)$  y  $(M_o, i_o)$  como:

$$K_m = \frac{M_o - M_n}{i_o - i_n} = \frac{(37,4 - 8)[Ncm]}{(8,4 - 2,1)[A]} \times \frac{1[m]}{100[cm]} = 46,7mNm/A$$

Además según el fabricante las reductoras compatibles para éste motor tienen un rendimiento  $Rd$  de un 90 % hasta valores de  $K = 8$

Para el caso del robot P.Tinto (Eurobot 2015) sus datos son:

$$\begin{aligned} m &= 15Kg \\ D &= 0,6m \\ d &= 0,131m \\ d2 &= 0,112m \\ K_a &= 1 \\ V_{bat} &= 24V \end{aligned}$$

Por último, sustituyendo en (3.39) se obtiene un valor de

$$K_{min} = 6,37.$$

Puesto que es menor que 8 el rendimiento utilizado de la reductora es válido. La reductora del fabricante de valor cercano por exceso es la reductora PLG42S 8:1.

En el caso real, ya se disponían de estos motores y además ya venían con una reductora PLG42S de factor de reducción 4:1<sup>4</sup>. Se optó por añadir una segunda etapa de reducción. Esta segunda etapa se implementó mediante una reducción por correa dentada, cuyo rendimiento es del 90 %. Recalculando  $K_{min}$  para un rendimiento de 0,9<sup>2</sup> se obtiene una

$$K_{min} = 7.$$

A partir de este valor, se escogió un factor de reducción 2:1 para la segunda etapa obteniendo un factor de reducción total de 8:1.

### 3.4.6 Calculo de la velocidad máxima

Asumiendo que los motores del robot tienen asociado un controlador de posición o velocidad, el dimensionamiento de estos se realiza de forma que el controlador no llegue nunca a la tensión de saturación. Esto es, la tensión  $U$  de los motores ha de estar por debajo de la tensión de saturación  $v_{sat}$ . Interesa conocer cual es el margen entre estas dos tensiones necesario para realizar un control sobre los motores sin bloqueos.

El análisis a realizar parte de la premisa de utilizar un perfil de velocidad trapezoidal. Para ello se considera que el perfil de velocidad constituye una consigna de velocidad para el controlador [28].

---

<sup>4</sup>Estos motores se compraron en el año 2010 a un vendedor de eBay. Los motores nuevos son caros y por que entonces no había presupuesto suficiente para comprar unos nuevos a demanda de los requisitos del robot.

El análisis parte del perfil de velocidad representado en la figura 3.17 y de las ecuaciones que describen el modelo del motor de continua

$$U = r i + K_v \omega, \quad (3.40)$$

$$M = K_m i \quad (3.41)$$

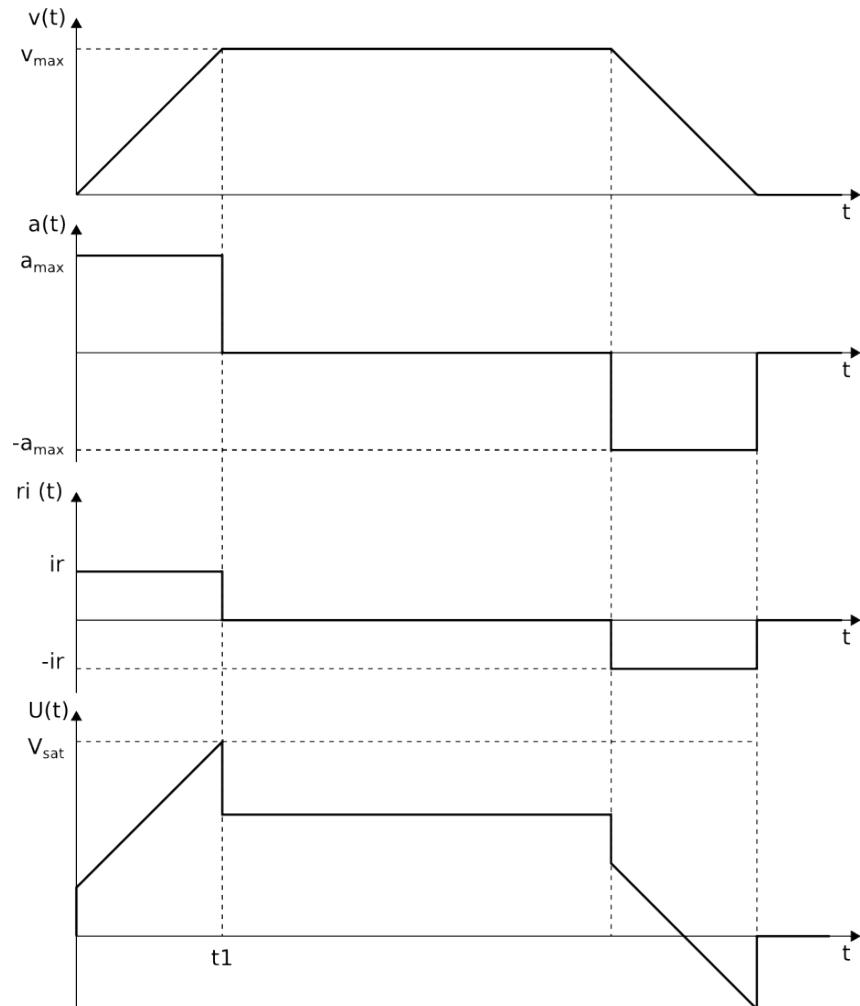


Figura 3.17: Tensión de armadura del motor en el límite de saturación para un perfil de velocidad trapezoidal.

Se observa que durante las fases de aceleración y frenado la aceleración del robot es constante, por lo que el par motor<sup>5</sup> y la corriente son constantes<sup>6</sup>. La tensión de armadura  $U$  del motor es la suma del término óhmico  $r i$  y del término de velocidad angular  $K_v \omega$ . Por lo tanto durante las fases de aceleración y frenado, el término óhmico será constante y el término de velocidad proporcional a esta. De esta forma el valor máximo de la tensión de armadura del motor  $U$  se alcanza, al final de la fase de aceleración, en el instante  $t_1$ .

Por otro lado, hay que recordar que el cálculo de la reductora también se realiza en base a la tensión de saturación  $V_{sat}$ , para la cual las ruedas comenzarían a deslizar para evitar el bloqueo del motor. Es por ello que el instante  $t_1$  constituye el punto más crítico de las fases del perfil trapezoidal.

El límite de funcionamiento sin saturación es dado por la ecuación (3.40), valida en el instante  $t_1$ .

<sup>5</sup>  $M = F \times d = m a \times d$ . Si  $a = \text{cte}$ ,  $M = \text{cte}$

<sup>6</sup>  $M = K_m i$ . Si  $M = \text{cte}$ ,  $i = \text{cte}$

Por un lado, la corriente de armadura  $i$  se puede expresar en función de la aceleración  $a$  del robot del siguiente modo:

$$\begin{aligned} i &= \frac{M1}{K_m} \\ M1 &= \frac{M3}{K Rd} \\ M3 &= Fx \frac{D}{2} \\ 2Fx &= ma \end{aligned} \quad (3.42)$$

Por otro lado, la velocidad angular  $\omega$  se puede expresar en función de la velocidad  $v$  del robot como:

$$\begin{aligned} \omega_1 &= K \omega_3 \\ v &= \frac{2\omega_3}{D} \end{aligned} \quad (3.43)$$

Sustituyendo los desarrollos (3.42) y (3.43) en (3.40) se obtiene<sup>7</sup>:

$$U = \left( \frac{r m D}{4K Rd K_m} \right) a + \left( \frac{2K_m K}{D} \right) v \quad (3.44)$$

Representando la velocidad  $v$  en función de la aceleración  $a$  se obtiene la ecuación de la recta

$$v = v_o - \frac{v_o}{a_o} a \quad (3.45)$$

Donde

$$v_o = \frac{UD}{2KK_m} \quad (3.46)$$

$$a_o = \frac{4KURdK_m}{rDm} \quad (3.47)$$

Dicha recta se representa en la figura 3.18, donde se puede observar que la aceleración máxima  $a_{max}$  define la zona útil de funcionamiento sin saturación y deslizamiento.

Recapitulando y simplificando (3.45), la velocidad máxima del robot  $v_{max}$  al límite de saturación y deslizamiento para una aceleración dada se obtiene a partir de la expresión

$$v_{max} = v_o \left( 1 - \frac{a_{max}}{a_o} \right) \quad (3.48)$$

$$v_o = \frac{UD}{2KK_m} \quad (3.49)$$

$$a_o = \frac{4KURdK_m}{rDm} \quad (3.50)$$

Siguiendo con el ejemplo del dimensionamiento de la reductora del robot P.Tinto, a partir de los datos ya conocidos del motor Dunkermotoren BG40x50 y de las expresiones (3.49) y (3.50), se obtiene la recta velocidad-aceleración representada en la figura 3.19 donde

$$v_o = 1,93m/s \quad (3.51)$$

$$a_o = 11,29m/s^2 \quad (1,15g) \quad (3.52)$$

<sup>7</sup> $K_v$  ha sido sustituido por  $K_m$  ya que coinciden si ambas constantes se expresan en  $[V/(rad/s)]$  y  $Nm/A$ , respectivamente

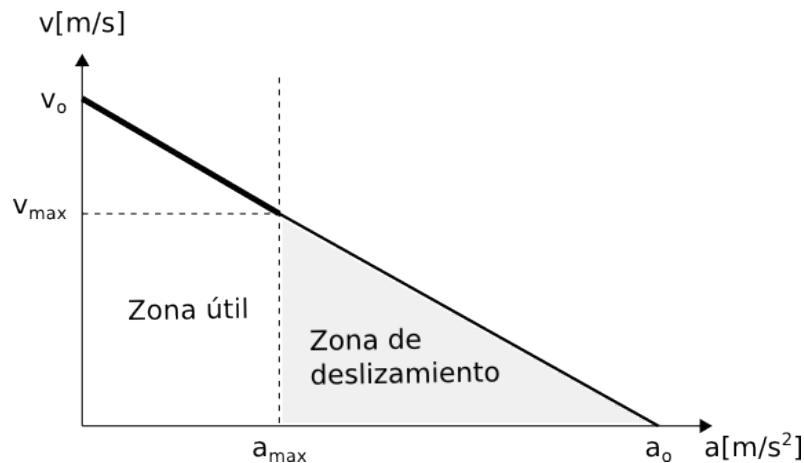


Figura 3.18: Recta de velocidad máxima para una aceleración dada y zona útil de funcionamiento sin saturación y deslizamiento.

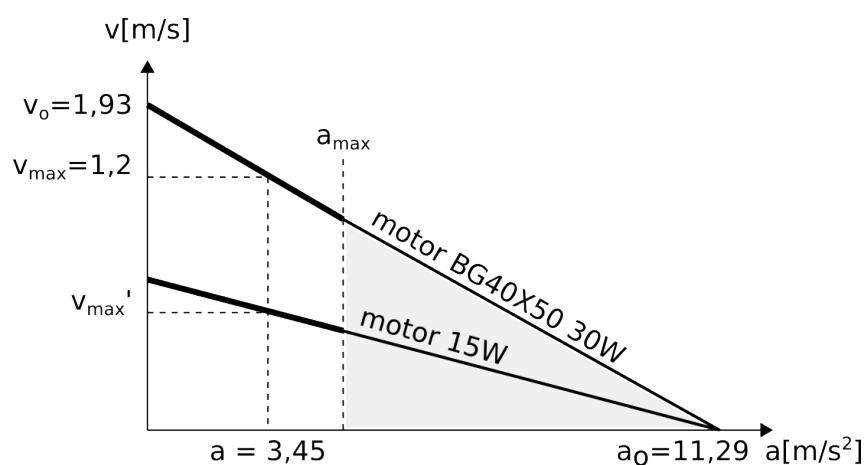


Figura 3.19: Velocidad y aceleración del robot P.Tinto para un funcionamiento sin saturación y deslizamiento.

P.Tinto es un robot tipo con ruedas en el medio no simétrico como el de la figura 3.11. A partir del coeficiente de adherencia de las ruedas  $K_a = 1$  y las medidas del centro de gravedad respecto al eje motriz:

$$\begin{aligned} d_1 &= 19,18\text{mm} \\ d_2 &= 111,8\text{mm} \\ d_2' &= 139,68\text{mm} \\ h &= 122,9\text{mm} \end{aligned} \tag{3.53}$$

Mediante las ecuaciones (3.21) y (3.20) se determina una aceleración máxima

$$a_{max} = 0,44g (4,32\text{m/s}^2) \tag{3.54}$$

Teniendo en cuenta un margen de seguridad del 20 % se fija la aceleración del robot a un valor de  $0,35g$  ( $3,45\text{m/s}^2$ ). A partir de la ecuación (3.48) se obtiene la velocidad correspondiente para un funcionamiento sin saturación y deslizamiento:

$$v_{max} = 1,2\text{m/s} \tag{3.55}$$

Se ha representado una segunda recta en la figura 3.19 correspondiente a un motor de la mitad de potencia. Observar que para una misma aceleración la velocidad es menor. En el primer caso la velocidad es superior a la velocidad deseada ( $1\text{m/s}$ ) y el motor BG40x50 es válido, mientras que en el segundo caso el motor no sería valido.

La gráfica de la figura 3.20 muestra el resultado de aplicar un perfil trapezoidal cuya velocidad es la máxima calculada. Observar que para la velocidad máxima calculada al final de la fase de aceleración (instante  $t_1$  de la figura 3.17) la tensión de control de los motores se encuentra muy cerca de su valor de saturación (63500 cuentas, que equivale al 100 % de ciclo de trabajo de una señal PWM). Si se disminuye la velocidad hasta  $0,8\text{m/s}$ , el valor de la tensión baja hasta el 80 % de su máximo, como muestra la figura 3.21.

### 3.5 Control en posición de tipo polar

La plataforma robótica base implementa un control de posición de tipo polar. Este tipo de control es muy utilizado por los robots de Eurobot y es el que mejor se adapta a este tipo de aplicación, ya que, en última medida el robot ha de ser capaz de llegar a posiciones específicas del campo donde manipular elementos de juego. Así mismo, este tipo de control de posición ha sido muy comentado en los foros de Eurobot [6], documentado [28] [30] e implementado por muchos equipos [5].

La posición del robot viene dada por una consigna de distancia  $d$  y de una consigna de orientación o ángulo  $a$ , correspondientes a un controlador de distancia y un controlador de ángulo. El control de posición polar permite obtener un mejor rendimiento que un controlador de posición por rueda del robot (ver figura 3.22).

El diagrama de bloques del controlador polar se muestra en la figura 3.23. En cada controlador error de posición  $e[n]$  se obtiene a partir de la resta de la consigna  $x[n]$  y de la medida instantánea de los encoders  $y[n]$  en cada periodo de muestreo  $T_s$ . Para el caso del control en distancia,  $y[n]$  se corresponde con la distancia recorrida por el robot y se obtiene a partir del valor medio de la distancia recorrida por cada encoder. Mientras que en el caso del controlador en ángulo,  $y[n]$  se corresponde con el ángulo instantáneo del robot y se obtiene a partir de la diferencia de la distancia recorrida por los encoders.

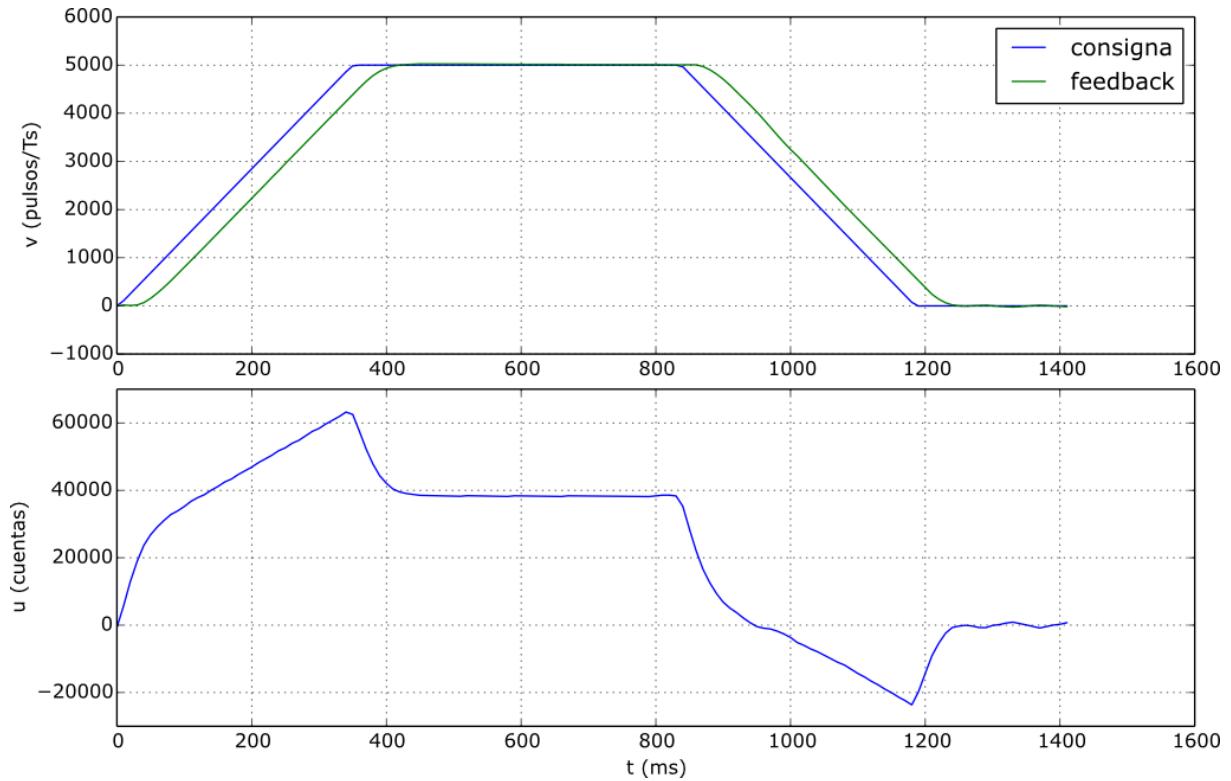


Figura 3.20: Señal de control de los motores para un perfil de velocidad trapezoidal de  $1,2\text{m/s}$  y una aceleración de  $3,45\text{m/s}$

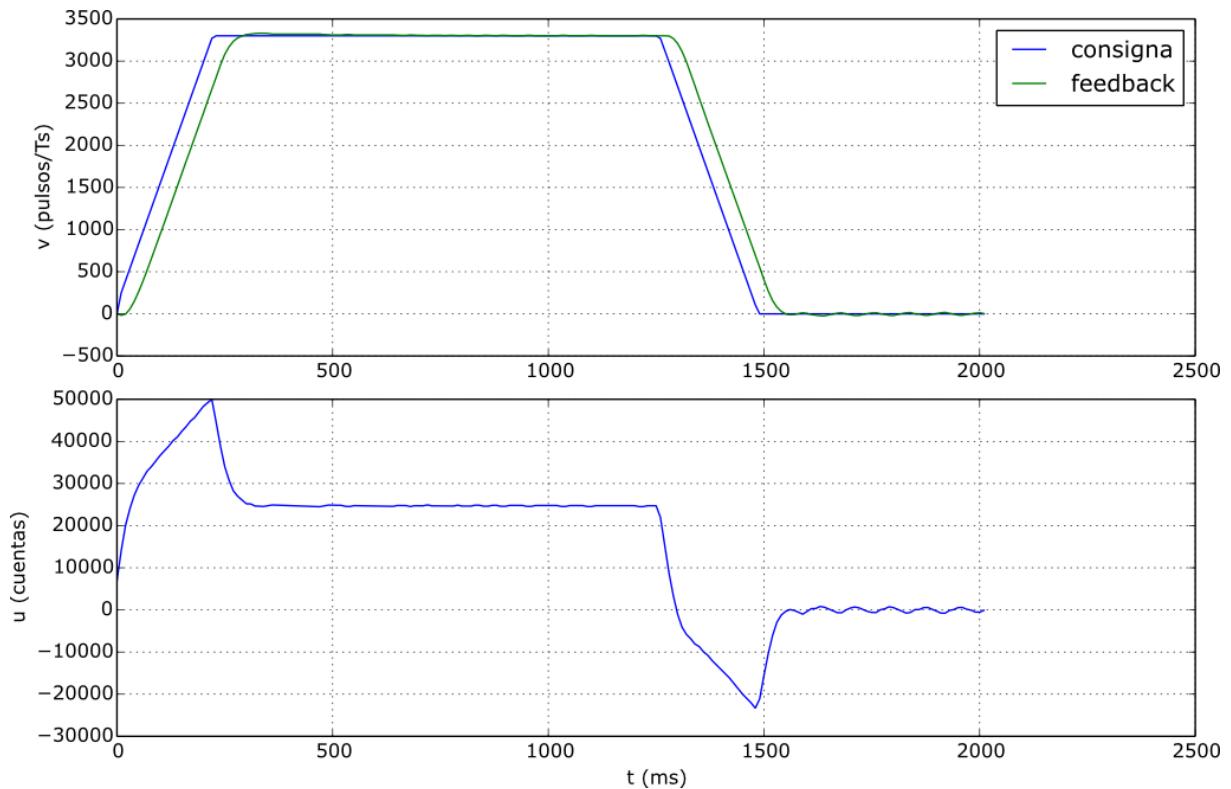


Figura 3.21: Señal de control de los motores para un perfil de velocidad trapezoidal de  $0,8\text{m/s}$  y una aceleración de  $3,45\text{m/s}$

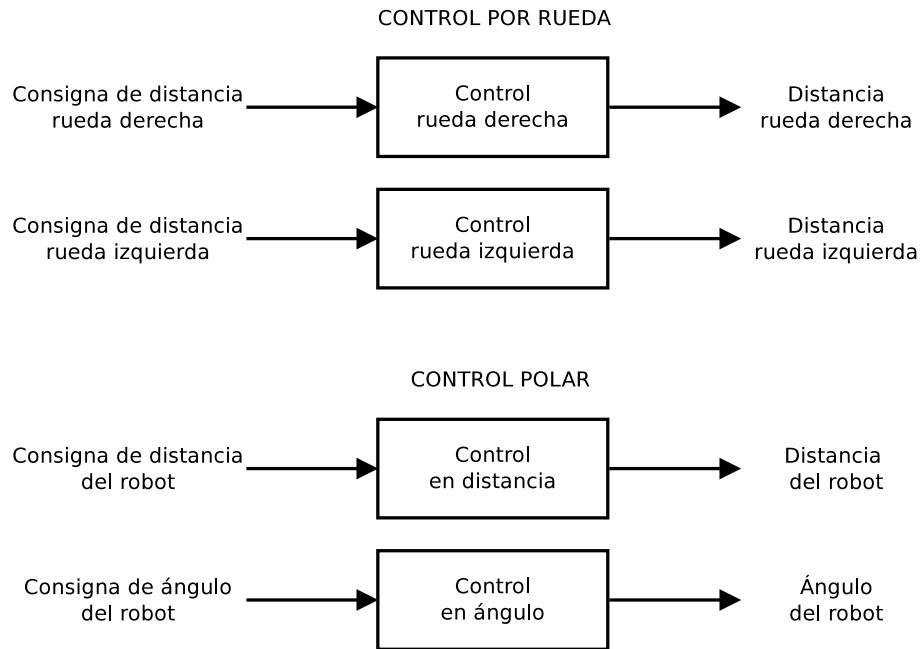


Figura 3.22: Control de posición polar vs. control independiente de posición por rueda

El error de consigna es procesado por un controlador PID (Proporcional Integral Derivativo) generando a su salida la señal de control  $u[n]$  para los motores. Notar que en el caso del controlador de distancia la señal de control es la misma para ambos motores, mientras que en el caso del controlador de ángulo, el signo de la señal de control del motor izquierdo se encuentra invertida.

En la práctica, se ha utilizado un controlador PD frente al clásico PID, esta elección viene dada por el balance entre las ventajas e inconvenientes del término integral.

Ventajas:

- Mejor precisión en régimen permanente
- Mejor respuesta a perturbaciones

Desventajas:

- Tendencia a desestabilizar el bucle de control
- Introduce el fenómeno de *windup*<sup>8</sup>.

El término integral suele ser utilizado en procesos industriales donde la respuesta a perturbaciones es muy importante. Sin embargo, para un robot de Eurobot su uso puede ser cuestionable tal y como se argumenta en [28]. Esto es debido a las posibles reacciones bruscas del efecto *windup* y a que en Eurobot una perturbación significa que el robot se ha bloqueado debido a que ha chocado con un oponente o una pared. En dicho caso, si la reductora del robot se ha dimensionado correctamente, las ruedas derraparan, por lo que el término integral no tendrá ningún efecto.

Al utilizar una arquitectura PD el sistema de control es tipo 1 [31]. Por un lado, esto implica que un ajuste más sencillo de los controladores. Hay que tener en cuenta que un controlador tipo 1 siempre tendrá un mejor rendimiento y fiabilidad que un controlador tipo 2 mal ajustado. Por otro lado, un sistema de

<sup>8</sup>Efecto por el cual el controlador puede saturarse dando como resultado respuestas bruscas no lineales en el momento que la perturbación cesa

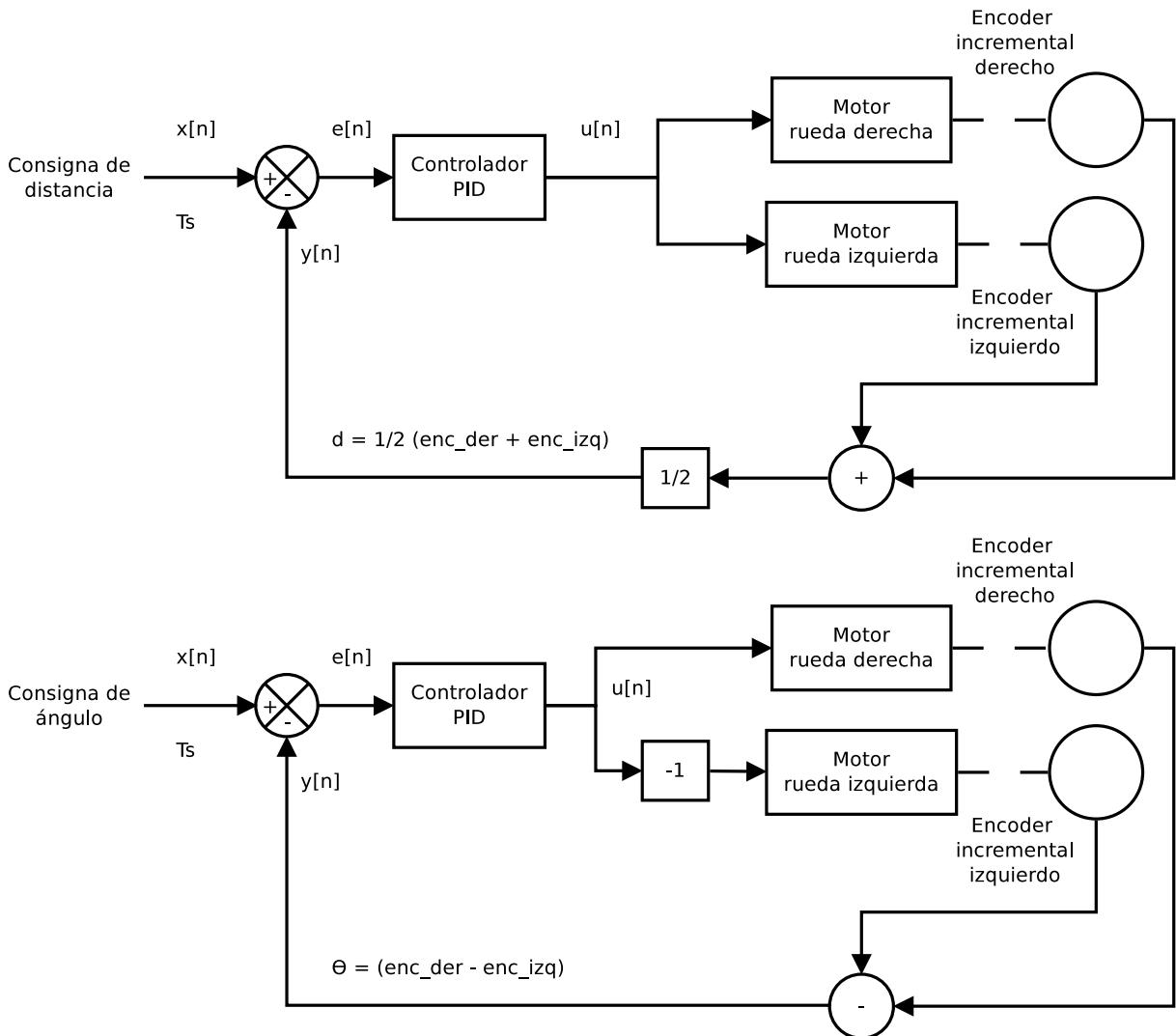


Figura 3.23: Diagrama de bloques del control de posición polar

control de tipo 1 implica un error de posición nulo ante una entrada tipo escalón y un error de velocidad constante ante una entrada tipo rampa. Teniendo en cuenta que las consignas de los controladores de posición vendrán dadas a partir de perfiles de velocidad trapezoidal, durante las fases de aceleración y frenado el controlador seguirá a la consigna a la misma velocidad pero con un retardo constante, mientras que durante la fase de velocidad constante o en estado de reposo (robot parado) el error de posición será nulo. Dado que es en distancia donde se requiere un control preciso, el retardo en velocidad es un factor asumible.

Respecto a la acción derivativa de un controlador PID o PD, esta suele introducir cierto ruido en la señal de control de los motores  $u[n]$  y da como resultado un controlador más nervioso. Este ruido puede reducirse mediante un controlador del tipo PI-D [31] donde la corrección derivativa se realiza directamente a partir de la medida de los encoders. Este caso ha sido estudiado en [28], donde se comprueban las ventajas de un controlador P-D frente a una arquitectura clásica PD. Efectivamente, la señal de control  $u[n]$  es menos ruidosa y de un valor de pico menor, lo que repercute en un control más suave de la velocidad del robot. Además se observa que el error  $e[n]$  del controlador P-D es menor que en el PD. Por contra, se observa un retardo mayor en las fases de aceleración y frenado para en controlador P-D, consiguiendo la misma aceleración para ambas arquitecturas.

Otra opción para disminuir el efecto del término integral consiste en asignar un periodo de muestreo

para el termino derivativo  $T_d$  menor que el periodo de muestreo del bucle  $T_s$ , de forma que, el error derivativo quede promediado y se consiga una respuesta más suave a la salida del controlador.

### 3.5.1 Implementación del perfil de velocidad trapezoidal

La implementación del perfil de velocidad trapezoidal se consigue mediante un filtro de consigna previo a la consigna de distancia y ángulo de los controladores (ver figura 3.24). Dado que lo que se quiere limitar es la aceleración y la velocidad con la que varia la consigna de posición se necesita de un filtro de segundo orden. Dicho filtro evalúa la primera y segunda derivada de la consigna de posición  $x[n]$ , correspondientes a la aceleración y velocidad máximas. En caso de superarse alguna de ellas la consigna de posición a la salida del filtro  $x'[n]$  es limitada al valor de la aceleración o la velocidad máxima, según aplique.

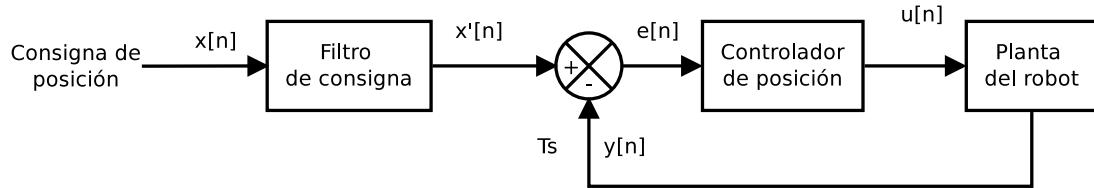


Figura 3.24: Implementación del perfil de velocidad trapezoidal mediante un filtro de consigna

### 3.5.2 Sistema de unidades del robot

Los controladores de posición a implementar trabajan en unidades diferentes a las del sistema internacional (SI). A la hora de establecer la consigna de los mismos o fijar la velocidad y aceleración máxima del filtro de consigna es necesario conocer los coeficientes de conversión entre ambos sistemas de unidades. La tabla 3.3 muestra las unidades de la distancia, velocidad y aceleración para el sistema internacional y el sistema de unidades del robot.

Tabla 3.3: Unidades SI y del sistema robot

Magnitud	Simbolo	Unidades SI	Unidades sistema Robot
Distancia	$d$	$m$	<i>pulsos</i>
Velocidad	$v$	$m/s$	$pulsos/T_s$
Aceleración	$a$	$m/s^2$	$pulsos/T_s^2$

Los coeficientes de conversión de distancia  $C_d$ , velocidad  $C_v$  y aceleración  $C_a$  entre sistemas se calculan como:

$$C_d = \frac{N \times M}{\pi D_n} \left[ \frac{\text{pulsos}}{m} \right] \quad (3.56)$$

$$C_v = C_d T_s \left[ \frac{\text{pulsos}/T_s}{m/s} \right] \quad (3.57)$$

$$C_a = C_d T_s^2 \left[ \frac{\text{pulsos}/T_s^2}{m/s^2} \right] \quad (3.58)$$

Siendo:

$N$  : Numero de pulsos por vuelta de encoder

$M$  : Factor de multiplicación de decodificación de señales en cuadratura del encoder

$D_n$  : Diámetro nominal del encoder

$T_s$  : Periodo de muestreo

De esta forma dada una distancia  $d$ , velocidad  $v$  y aceleración  $a$  en unidades del SI su valor en unidades del robot serian:

$$d_r = C_d d \quad (3.59)$$

$$v_r = C_v v \quad (3.60)$$

$$a_r = C_a a \quad (3.61)$$

$$(3.62)$$

Por ejemplo para los datos, correspondientes al robot P.Tinto (Eurobot 2015):

$$N = 3600 \text{ pulsos}$$

$$M = 4$$

$$D_n = 55 \text{ mm}$$

$$T_s = 5 \text{ ms}$$

Se obtienen las constantes

$$C_d = 83339,121$$

$$C_v = 416,696$$

$$C_a = 2,083$$

Así por ejemplo, para una distancia de  $1m$ , una velocidad de  $1m/s$  y una aceleración de  $3,6m/s^2$  se obtiene:

$$d_r = C_d 1 = 83339,121 \quad (3.63)$$

$$v_r = C_v 1 = 416,696 \quad (3.64)$$

$$a_r = C_a 3 = 7,501 \quad (3.65)$$

Se observa que los valores resultantes son decimales, lo cual implicaría trabajar con decimales en la implementación de los controladores. Esto no es nada óptimo desde el punto de vista de procesamiento digital. Una posible solución pasa por redondear dichos valores para poder trabajar con números enteros, pero para valores pequeños como el de la aceleración el error por redondeo podría ser considerable. Para

solucionar dicha perdida de precisión se añade un factor  $K_{imp}$ :

$$C_d = K_{imp} \frac{N \times M}{\pi D_n} \left[ \frac{\text{pulsos}}{m} \right] \quad (3.66)$$

$$C_v = K_{imp} C_d T_s \left[ \frac{\text{pulsos}/T_s}{m/s} \right] \quad (3.67)$$

$$C_a = K_{imp} C_d T_s^2 \left[ \frac{\text{pulsos}/T_s^2}{m/s^2} \right] \quad (3.68)$$

Por ejemplo, para un  $K_{imp} = 10$  el valor de la consigna de aceleración sería  $a_r = 75$  lo que equivale a un error menor del 1%.

Para que la conversión sea válida, el factor  $K_{imp}$  ha de integrarse en el controlador. Como muestra la figura 3.25, la integración se realiza en el lazo de realimentación de forma que el factor  $K_{imp}$  multiplica a la medida de posición de los encoders. Notar que la constante  $K_{imp}$  no tiene efecto sobre la medida de ángulo.

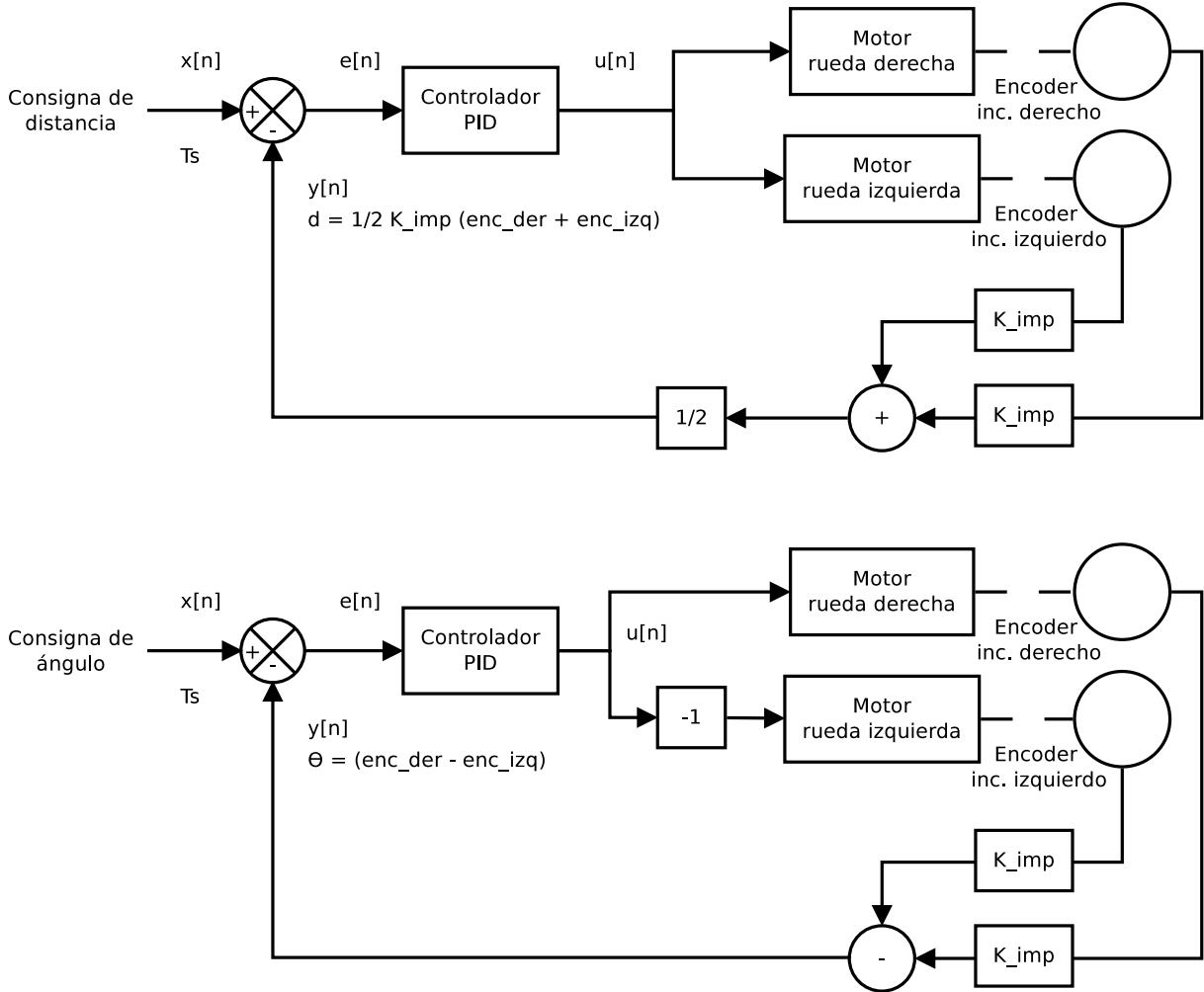


Figura 3.25: Integración del factor  $K_{imp}$  en el diagrama de bloques del control de posición polar

### 3.5.3 Ajuste del controlador de posición

El método de ajuste utilizado es experimental, al igual que otros métodos experimentales como los basados en las reglas de Ziegler-Nichols [31], permiten obtener una estimación razonable de los parámetros del

controlador y proporcionan un punto de partida para una sintonía fina.

Independientemente del método utilizado, para conseguir un ajuste preciso de los controladores es indispensable contar con una herramienta de depuración que permita adquirir las diferentes señales del diagrama de control y poder representarlas. Por ejemplo, la plataforma desarrollada cuenta con una interfaz serie RS-232 por la cual se vuelcan los datos durante el proceso de ajuste de los controladores.

El proceso de ajuste utilizado es el propuesto en [28]. A la hora de llevarlo a cabo es aconsejable primero en ajustar el controlador de ángulo y posteriormente el de distancia. Esto es debido a que durante el proceso de ajuste del controlador de ángulo se llevan a cabo ciertas secuencias de rotación, que luego son secuencias de desplazamiento en el caso del controlador de distancia. Si en este último caso el controlador de ángulo no está previamente ajustado, durante el proceso de ajuste del controlador de distancia habrá desviaciones laterales que dificultarían el proceso.

Primero para ajustar el controlador de ángulo se bloquea el controlador de distancia fijando una consigna de valor 0. Y luego, para ajustar el controlador de distancia se hace lo contrario, y se fija una consigna de ángulo igual a 0. El ajuste de ambos controladores se hace sin ningún tipo de perfil o rampa de velocidad, se busca observar la respuesta a una perturbación tipo escalón.

Para cada controlador se ajusta primero el término proporcional, luego el término derivativo y por último, si corresponde, el término integral. En este caso se presenta el proceso de ajuste de un controlador PD.

1. Ajuste del término proporcional  $K_p$ :

- (a) Anular la corrección derivativa  $K_d = 0$ .
- (b) Fijar una ganancia  $K_p$  inicial.
- (c) Saca al controlador de su posición de equilibrio. Desplazamientos entre 5 y 10 grados en ángulo, o 5cm en distancia.
- (d) Observar las gráficas de la respuesta del controlador.
- (e) Incrementar el valor de  $K_p$  hasta observar en las gráficas una respuesta amortiguada de duración entre 1 y 2 segundos.

2. Ajuste del término derivativo  $K_d$ :

- (a) Fijar una ganancia  $K_d$  inicial.
- (b) Saca al controlador de su posición de equilibrio. Desplazamientos entre 5 y 10 grados en ángulo, o 5cm en distancia.
- (c) Observar las gráficas de la respuesta del controlador.
- (d) Incrementar el valor de  $K_d$  hasta observar en las gráficas que se obtiene una respuesta sin oscilaciones amortiguadas correspondiente hasta llegar una respuesta subamortiguada con un sobreimpulso  $M_p$  cercano al 4,6% ( $\xi = 0,7$ ).

La figuras 3.26 y 3.27 muestra las gráficas obtenidas una vez realizado el ajuste de los términos del controlador de distancia.

Una vez ajustados ambos controladores es posible realizar un ajuste fino de los mismos con el fin de obtener un tiempo de establecimiento. Siguiendo el mismo procedimiento descrito para el ajuste, se incrementa el término  $K_p$  en un 20% y se vuelve a ajustar el término derivativo. Se repite esta operación sucesivamente hasta el momento en el que el término  $K_d$  no sea capaz de compensar el incremento de  $K_p$  poniendo atención en que la señal de control del motor  $u[n]$  no sature.

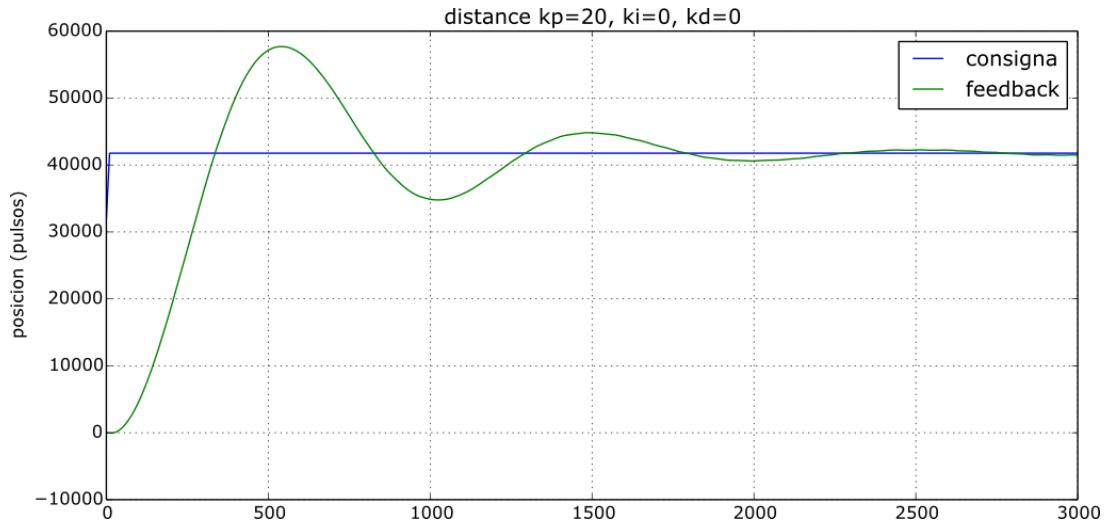


Figura 3.26: Resultado del ajuste del término proporcional  $K_p$  del controlador de distancia ante una consigna tipo escalón de 5cm

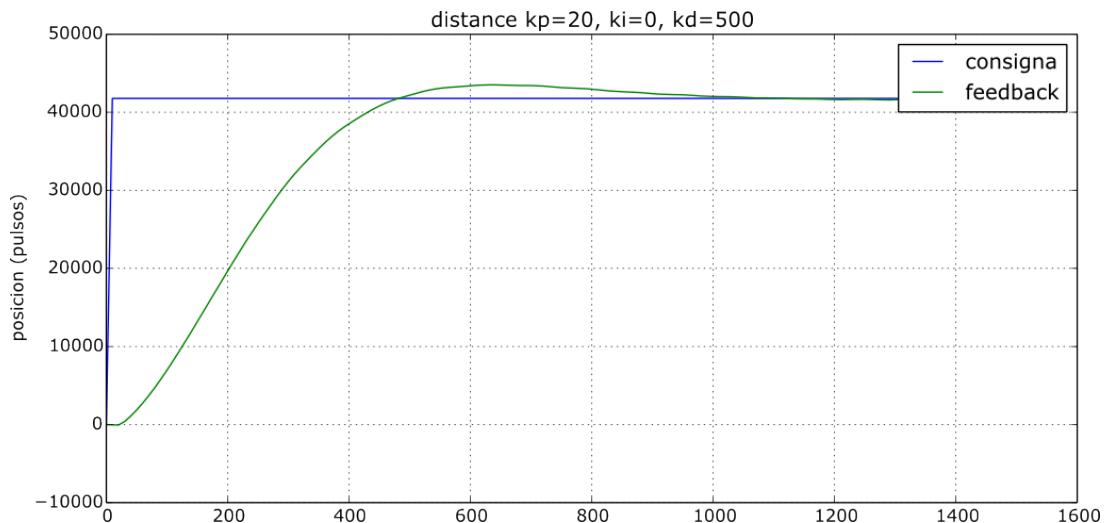


Figura 3.27: Resultado del ajuste del término proporcional  $K_d$  del controlador de distancia ante una consigna tipo escalón de 5cm

Por último, se realiza un último ajuste de los controladores a aplicando la consigna dada por el perfil de velocidad trapezoidal. En este último ajuste se busca obtener una fase de aceleración y frenado óptimo. Para ello se ha de prestar atención en que durante y al final de cada una de estas fase no existan variaciones bruscas de la posición del robot que puedan causar sobre impulsos de aceleración superiores a la aceleración máxima. Esto es, que la posición se ciña a la forma del perfil lo máximo posible. En las figuras 3.28 y 3.29 se muestra el resultado de ajustar el controlador de distancia con un perfil trapezoidal.

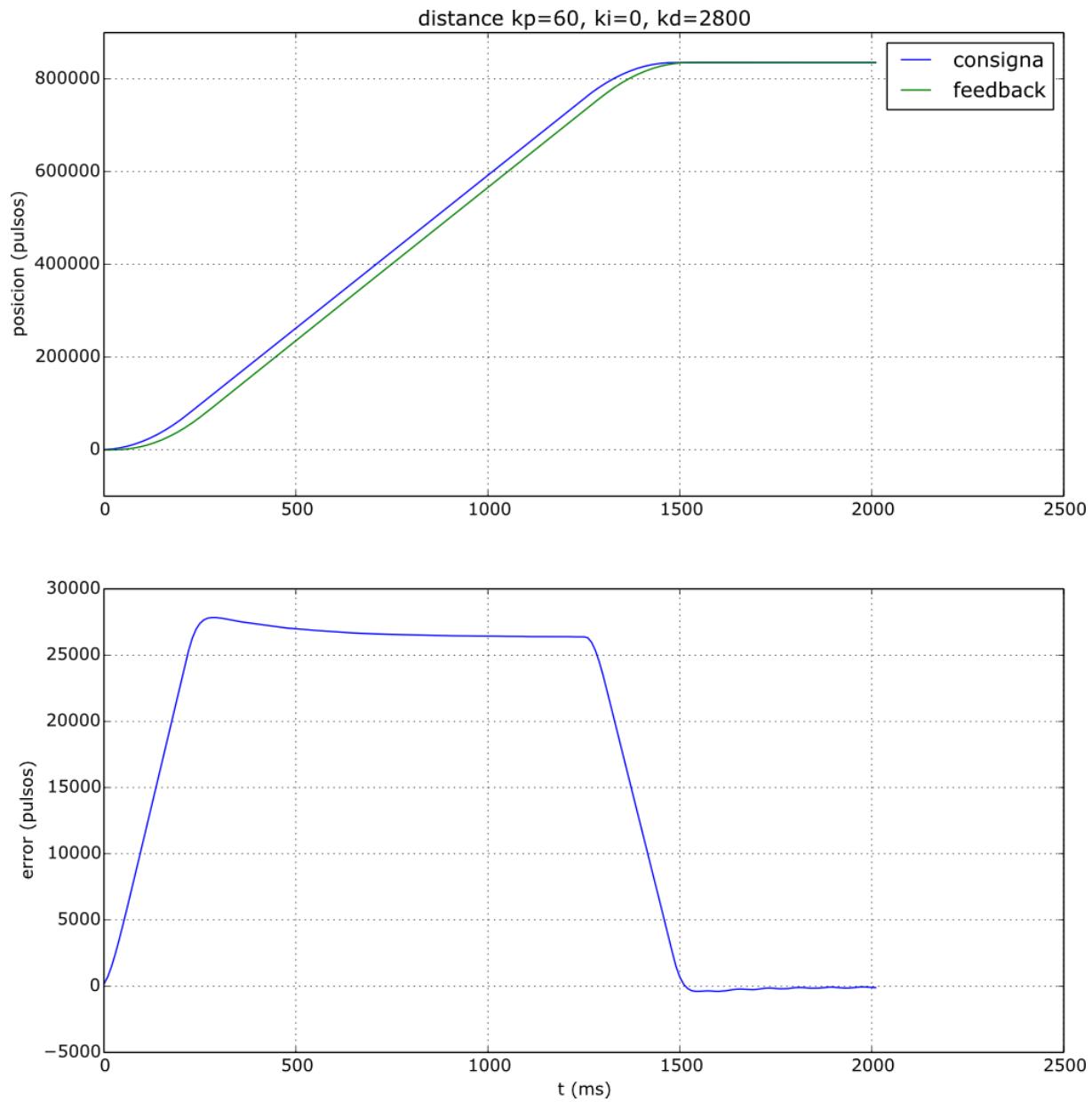


Figura 3.28: Resultado del ajuste del controlador de distancia aplicando un perfil de velocidad trapezoidal ( $a = 3,45 m/s^2$ ,  $v = 0,8 m/s$ ).

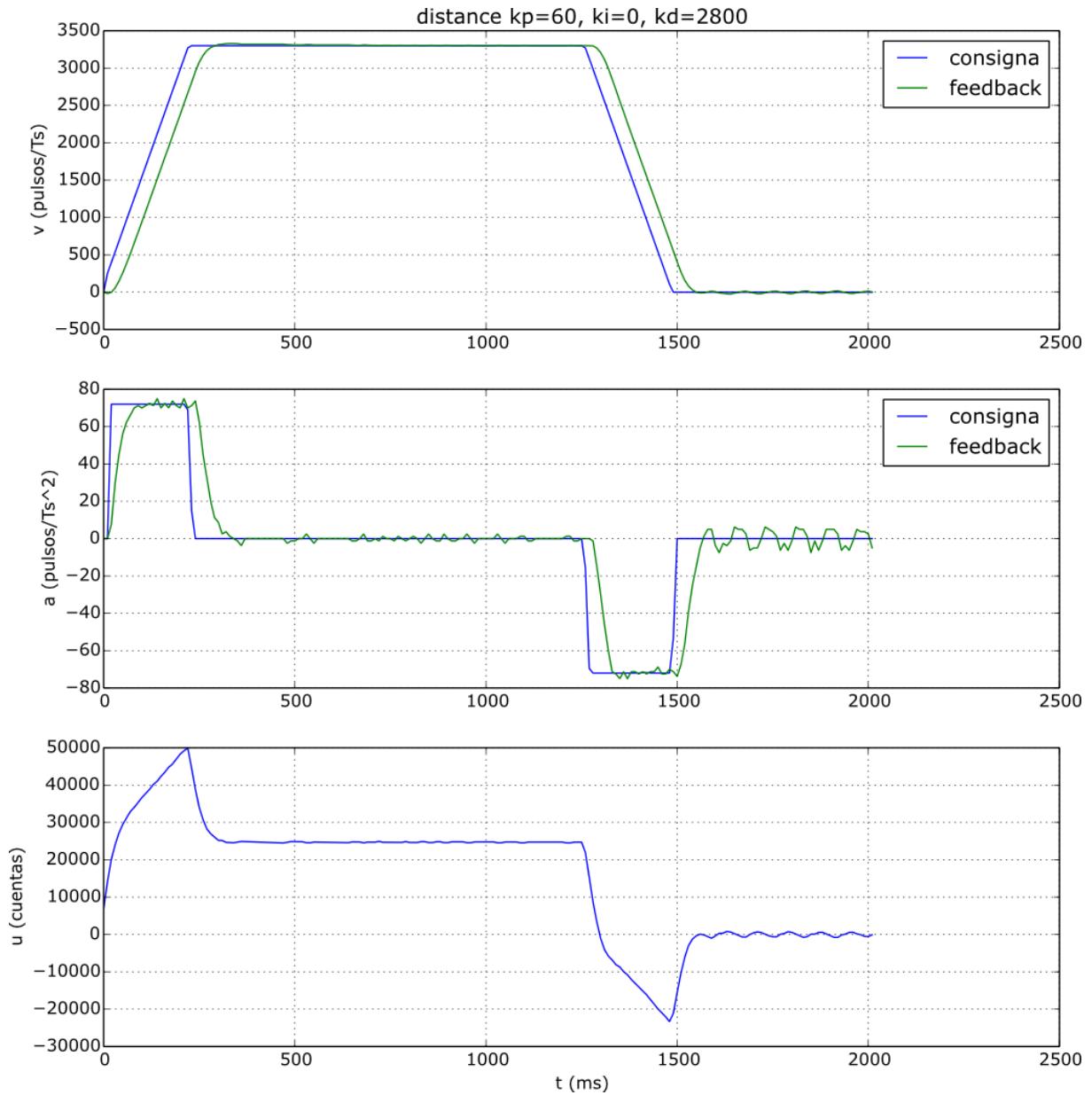


Figura 3.29: Resultado del ajuste del controlador de distancia aplicando un perfil de velocidad trapezoidal ( $a = 3,45 \text{m/s}^2$ ,  $v = 0,8\text{m/s}$ )

## 3.6 Posicionamiento mediante odometría

La odometría es el método más usado para determinar la posición momentánea de un robot. Dicha posición viene dada en un plano cartesiano por sus coordenadas absolutas  $(x, y, \theta)$  correspondientes a la distancia al origen de coordenadas y al ángulo respecto al eje de abscisas.

### 3.6.1 Calculo de la posición del robot

El sistema desarrollado para implementar este tipo de posicionamiento es el sistema de ruedas libres descrito en la sección 3.3. Los principales elementos de este sistema y las dimensiones relacionadas con la implementación de este método se muestra en la figura 3.30 donde  $D_n$  se corresponden con el diámetro nominal de las ruedas de los encoders y  $b$  con la distancia entre ejes de rueda de encoder.

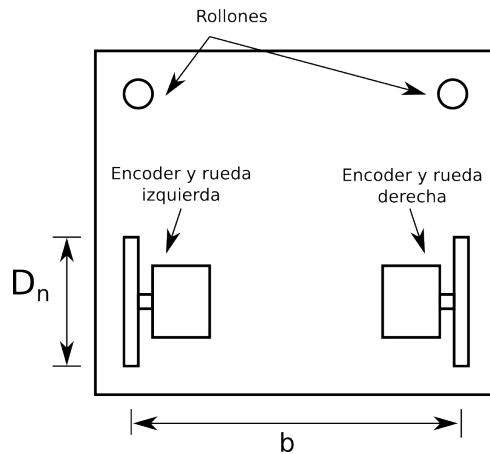


Figura 3.30: Representación en planta del robot. Componentes y dimensiones implicadas en la odometría

La determinación de la posición  $(x, y, \theta)$  del robot se obtiene a partir la medida del incremento de pulsos de cada encoder por periodo de muestreo  $T_s$ . A partir de la medida en un periodos de muestreo de los pulsos del encoder derecho  $N_D$  e izquierdo  $N_I$  se obtiene la medida de instantánea de distancia  $d$  y ángulo  $\theta$  en cada periodo de muestreo como:

$$d = \frac{N_D + N_I}{2} [\text{pulsos}] \quad (3.69)$$

$$\theta = (N_D - N_I) [\text{pulsos}] \quad (3.70)$$

La conversión de unidades del robot a unidad del SI se realiza multiplicando dichas medidas por los factores de conversión de distancia  $C_d$  y angulo  $C_\theta$ , calculados como:

$$C_d = \frac{N \times M}{\pi D_n} \left[ \frac{\text{pulsos}}{m} \right] \quad (3.71)$$

$$C_\theta = C_d \frac{b}{2} \left[ \frac{\text{pulsos}}{\text{rad}} \right] \quad (3.72)$$

Por un lado, la coordenada  $\theta$  del robot puede obtenerse directamente de la ecuación (3.70) como:

$$\theta_n = \frac{N_{Dn} - N_{In}}{C_\theta} [rad] \quad (3.73)$$

Por otro lado, las coordenadas  $(x, y)$  se calculan a partir del análisis de la trayectoria del robot muestreada para dos instantes consecutivos (ver figura 3.31). Las posiciones A y B se corresponden con los instantes  $n$  y  $n - 1$  a los que les corresponde una medida de ángulo  $\theta_n$  y  $\theta_{n-1}$  y una medida de distancia  $d_n$  y  $d_{n-1}$ . La distancia curvilinear entre los puntos A y B es  $L$  y su proyección sobre los ejes  $dx$  y  $dy$ .

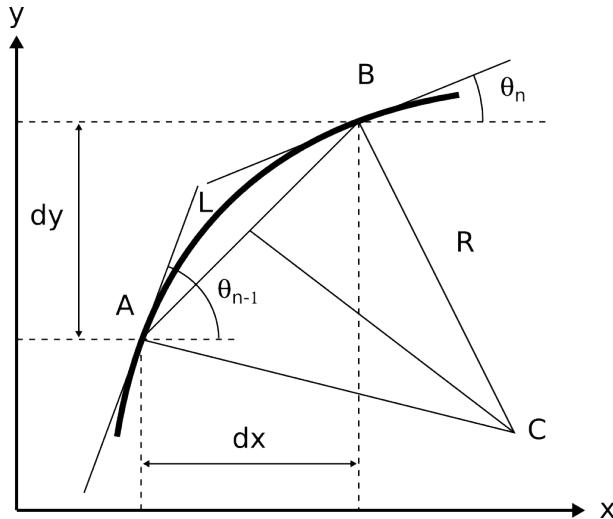


Figura 3.31: Representación geométrica de dos puntos de una trayectoria

Las coordenadas  $(x, y)$  en un instante dado se corresponden con:

$$x_n = x_{n-1} + dx \quad (3.74)$$

$$y_n = y_{n-1} + dy \quad (3.75)$$

$$(3.76)$$

En teoría la obtención de las proyecciones  $dx$  y  $dy$  resulta imposible dado que se desconoce la forma de la trayectoria  $L$  descrita entre los dos instantes de muestreo. En la práctica se aplica una aproximación lineal o circular, de forma que  $L$  se aproxima a una recta o a un arco de circunferencia de radio  $R$  y centro de giro en  $C$ .

Para el caso de la aproximación lineal el ángulo del segmento AB  $\theta_{avg}$  se calcula como el valor medio del ángulo en cada punto:

$$\theta_{avg} = \frac{\theta_n + \theta_{n-1}}{2} \quad (3.77)$$

Por la aproximación lineal,  $L$  se corresponde con la longitud del segmento AB:

$$L = d_n - d_{n-1} \quad (3.78)$$

El valor de  $dx$  y  $dy$  para la aproximación lineal se obtiene como:

$$dx = L \cos(\theta_{avg}) \quad (3.79)$$

$$dy = L \sin(\theta_{avg}) \quad (3.80)$$

El cálculo de odometría utilizando la aproximación circular se resuelve en [32]. Para dicha aproximación el valor de  $dx$  y  $dy$  se obtiene como:

$$dx = K L \cos(\theta_{avg}) \quad (3.81)$$

$$dy = K L \sin(\theta_{avg}) \quad (3.82)$$

Siendo

$$K = \frac{\sin(\nabla\theta_n/2)}{\nabla\theta_n/2} \quad (3.83)$$

$$\nabla\theta_n = \theta_n - \theta_{n-1} \quad (3.84)$$

Observar que las expresiones de ambas aproximaciones difieren únicamente en el factor  $K$  que constituye un factor de escala para  $L$ . El estudio comparativo entre ambas aproximaciones realizada en [32] demuestra que para las condiciones de un robot de Eurobot la aproximación circular no añade ninguna ventaja frente a la aproximación lineal. En la aproximación circular, además de necesitar mayor tiempo de cálculo, la corrección aportada por  $K$  para trayectorias muy exigentes se encuentra en  $1 \times 10^{-5}$  y  $2 \times 10^{-5}$ , algo despreciable.

### 3.6.2 Medida y corrección de errores sistemáticos

El principal inconveniente de la odometría es la acumulación continua de error a medida que el robot se desplaza. Estos errores pueden sistemáticos y no sistemáticos.

#### Errores sistemáticos

1. Diferente diámetro de las ruedas de encoder.
2. Valor medio del diámetro de las ruedas de encoder diferente del diámetro nominal.
3. Ruedas de encoder desalineadas.
4. Incertidumbre en la distancia efectiva entre ejes de las ruedas de encoder o de tracción (debido al punto de apoyo con el suelo).
5. Resolución del encoder insuficiente
6. Tiempo de muestreo limitado

#### Errores no sistemáticos

1. Trayectos sobre suelos irregulares
2. Trayectos sobre objetos inesperados en el suelo
3. Deslizamiento de las ruedas de encoder debido a: sobre-aceleraciones, suelos resbaladizos, fuerzas externas (colisión con robot oponente), fuerzas internas (bloqueo de los rollones) o falta de contacto de las ruedas con el suelo.

Los errores sistemáticos son particularmente graves debido a que se acumulan constantemente, incluso llegan a contribuir al error más que los errores no sistemáticos.

Los errores no sistemáticos debido a superficies irregulares podrían contribuir más que los sistemáticos, pero no es el caso de los robots Eurobot donde las superficies son lisas. Además, el sistema de amortiguación de las ruedas libres de la plataforma robótica permite evitar los errores no sistemáticos debido al deslizamiento de las ruedas.

Respecto a los errores sistemáticos, las ecuaciones de cálculo de la odometría pueden introducir errores debido a la aproximación del método utilizado en su cálculo. La precisión en dicho cálculo depende del tiempo de muestreo frente a la velocidad del robot. Este error puede considerarse despreciable si se trabaja con períodos de muestreo de  $T_s < 10\text{ ms}$  para velocidades de  $v < 1\text{ m/s}$  [33].

Los errores sistemáticos no suelen cambiar durante el recorrido del robot, aunque la distribución de pesos sobre las ruedas de encoders puede afectar si estas se encuentran solidarias a la estructura del robot. No es el caso del sistema de ruedas libres desarrollado en el que estas se encuentran desacopladas mediante un mecanismo lineal o pivotante con amortiguación.

Los errores sistemáticos son normalmente causados por imperfecciones en la implementación del diseño y la mecánica del robot. Las dos fuentes de errores sistemáticos más significativas son la diferencia de diámetros de rueda de encoder y la incertidumbre en la distancia efectiva entre ejes de las ruedas de encoder o de tracción.

El error por diferente diámetro de rueda de encoder se representa por  $E_d$  y se define como

$$E_d = \frac{D_R}{D_L} \quad (3.85)$$

donde  $D_R$  y  $D_L$  son los diámetros reales de las ruedas de los encoders, siendo la relación nominal entre diámetro de ruedas 1,0.

Como muestra la ecuación (3.72) la medida de ángulo del robot depende de la distancia entre ejes de ruedas de encoder  $b$ . Por lo tanto una incertidumbre en dicha distancia implica un error en el cálculo. Este error se denota como  $E_b$  y se define como

$$E_b = \frac{b_{real}}{b_{nominal}} \quad (3.86)$$

donde  $b$  es la distancia entre ejes de ruedas de encoder.

Según [33] si se quiere reducir los errores de odometría, los errores de ángulo son la principal preocupación ya que una vez se incurre en ellos crecen sin límite. Por ello, en [33] se analiza el efecto de diferentes diámetros de rueda de encoder durante los giros del robot. La figura 3.32 representa el análisis realizado. Debido a la diferencia de diámetro de rueda el centro de giro del robot real es el punto  $O$ . Los giros sobre dicho punto generan un error de desplazamiento representado por el punto  $C$ .

El estudio revela que *el diámetro medio real de la ruedas es al ángulo de giro real como el diámetro nominal de la rueda al ángulo de giro nominal* y se deducen tres importantes conclusiones:

1. Diámetros distintos de rueda no causan un error de orientación durante los giros.
2. El error de orientación depende del *diámetro medio de las ruedas*  $D_{avg}$  expresado como:

$$D_{avg} = \frac{D_L + D_R}{2}$$

(3.87)

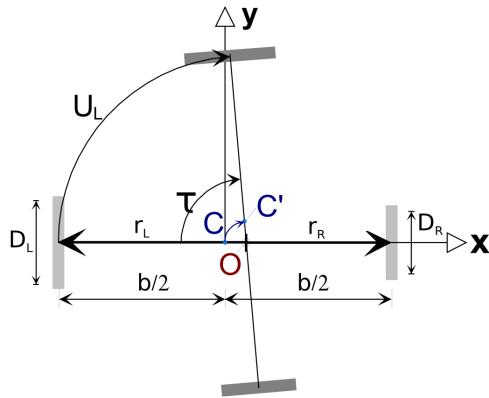


Figura 3.32: Análisis de la influencia de diferentes diámetros de rueda e incertidumbre de la distancia entre ejes durante el giro del robot. Imagen obtenida de [33].

Si  $D_{avg} > D_n$  el robot girará más de lo deseado. Si  $D_{avg} < D_n$  entonces el robot girará menos.

3.  $E_d$  tiene un efecto menor en la posición  $x$  e  $y$  del centro de giro produciendo un error de desplazamiento lateral.

### 3.6.2.1 Test del cuadrado bidireccional

Uno de los test que se suelen realizar con el fin de medir y corregir los errores de odometría es el test del cuadrado. En este test el robot realiza una trayectoria programada cuyo camino coincide con un cuadrado como muestra la figura 3.33. El robot comienza y termina en el mismo punto, al terminar el recorrido se mide la diferencia de la posición inicial y la final respecto a una pared de referencia. Debido al error  $E_d$  el robot realizará trayectorias curvas durante los trayectos rectos, y debido al error  $E_b$  el robot cometerá un error en los giros, como muestra la figura 3.33

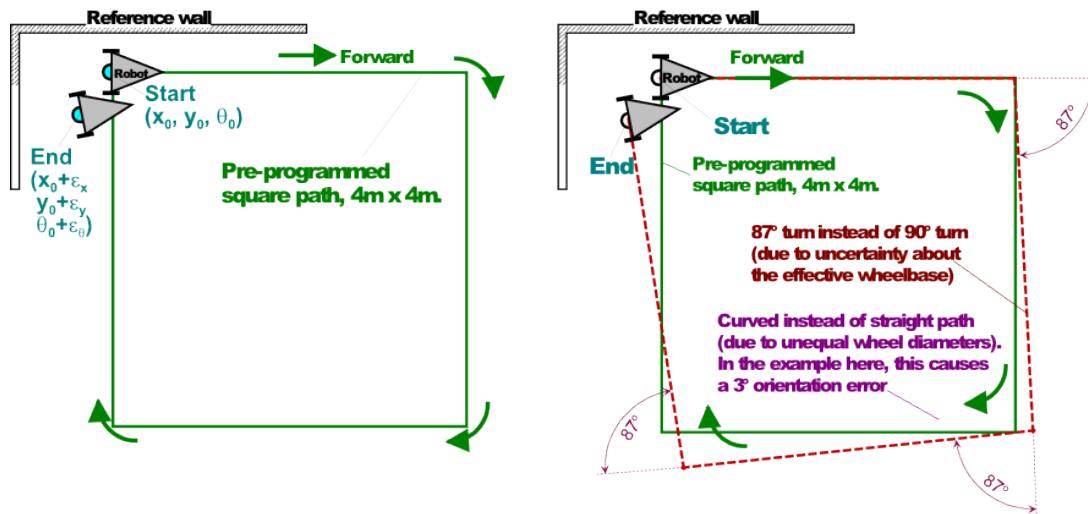


Figura 3.33: Test del cuadrado. Imagen obtenida de [33]

Sin embargo este test puede llevar a medidas erróneas debido a que los errores sistemáticos se pueden cancelar entre sí como muestra la figura 3.34. En [33] proponen un método que soluciona este inconveniente: el test del cuadrado bidireccional. En este test el robot realiza la trayectoria del cuadrado en dos direcciones: en sentido horario y antihorario de forma que aunque los errores se enmascaren en la

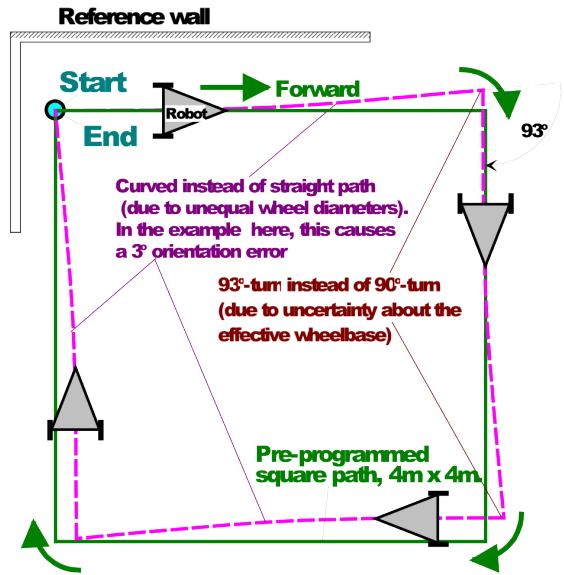


Figura 3.34: Test del cuadrado: cancelación de errores  $E_b$  y  $E_d$ . Imagen obtenida de [33]

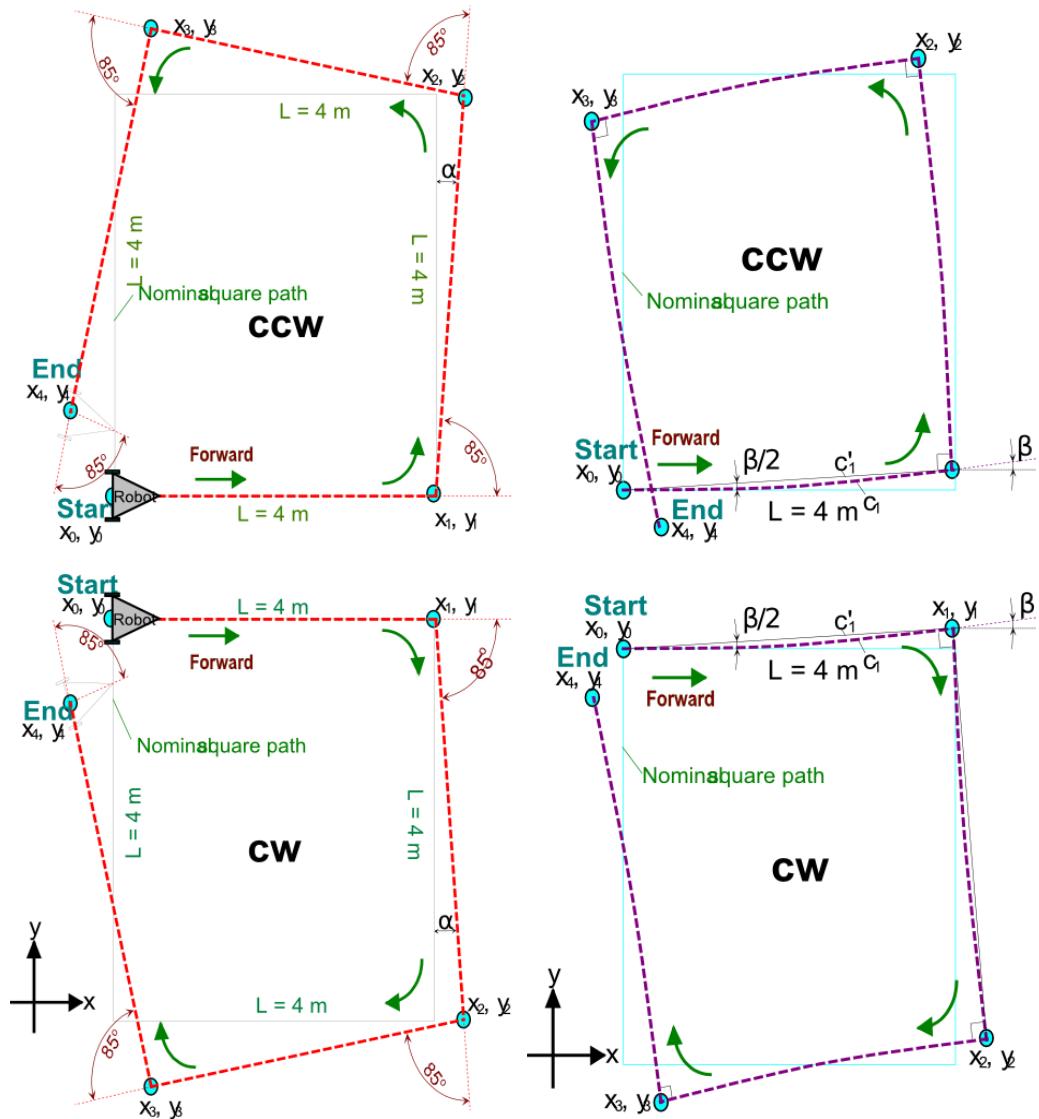


Figura 3.35: Test del cuadrado bidireccional. Efecto del error tipo A y B en la trayectoria y punto final del robot. Imagen obtenida de [33]

trayectoria en un sentido, darán la cara en el sentido contrario. El método es iterativo de forma que a cada iteración la corrección de los errores sistemáticos es mayor.

Así se definen dos tipos de errores: el error tipo A asociado a  $E_b$  debido a la distancia entre ejes de ruedas de encoder, el error tipo B asociado a  $E_d$  debido a diferente diámetro de rueda de encoder. La figura 3.35 representa el efecto de estos errores al realizar el test del cuadrado bidireccional.

Los resultados presentados en [33] utilizando el método del cuadrado bidireccional son satisfactorios. Este método ha sido utilizado para medir y corregir los errores de odometría del robot de Eurobot. Para la realización del test se realizó una trayectoria cuadrada de 1,5m de lado y se utilizó una cinta métrica para medir la posición del robot. Se realizaron 2 iteraciones obteniendo como resultado una mejora en los errores de odometría, pero no comparables con los resultados de [33]. Se cree que la diferencia de resultados fue debida a la longitud del camino recorrido (1,5m frente a 4m en el caso de [33]) y a la poca precisión de la herramienta de medida ya que en [33] realizaron medidas por ultrasonidos.

### 3.6.2.2 Método utilizado

En la práctica la medida y corrección de errores sistemáticos de odometría  $E_d$  y  $E_b$  se ha realizado utilizando un método propio. Este método realiza la corrección de cada error por separado teniendo en cuenta su dependencia. El método se divide en 2 partes:

1. Medida y corrección del error  $E_d$ .
2. Medida y corrección del error  $E_b$ .

El orden de realización es importante ya que  $E_d$  influye sobre  $E_b$ .

#### Medida y corrección del error $E_d$

Partiendo del sistema de ruedas libres de encoder diseñado (ver sección 3.3), una primera aproximación del valor de  $E_d$  se obtiene de la medida directa de la llanta de aluminio de las ruedas del encoder. Dicha medida se puede realizar con un calibre. Otra parte del error  $E_d$  viene dado por el echo de que las ruedas están formadas además por una junta tórica de goma que va a sufrir deformaciones debido a la amortiguación pivotante o lineal.

Para la medida y corrección de  $E_d$  se necesita una superficie lisa y una pared de referencia. En caso de disponer de un campo de juego, se puede utilizar el lado largo del mismo y su paredes como referencia. La figura 3.36 representa el test a realizar. Se recomienda utilizar una velocidad y aceleración baja con el fin de evitar deslizamientos.

El método consta de los siguientes pasos:

1. Situar el robot paralelo y cerca de las paredes de referencia. Fijar la posición del robot  $(x, y, \theta)$  a  $(0, 0, 0)$ . Medir la distancia del centro del eje motriz del robot, punto A, a las paredes de referencia.
2. Hacer recorrer al robot una distancia A-B lo mayor posible en linea recta ( $2,7m$  aproximadamente en caso de utilizar un campo de Eurobot).
3. Medir la distancia del centro del eje motriz del robot, punto C, a las paredes de referencia. Compensar la medida con la posición de odometría dada por el robot  $(x_r, y_r, \theta_r)$ .

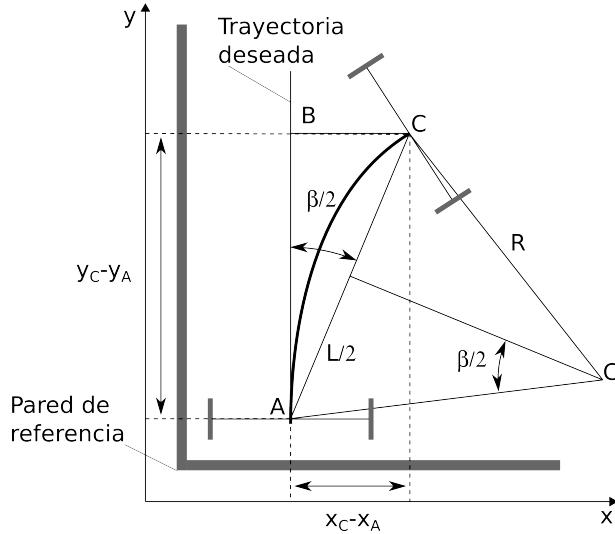


Figura 3.36: Método utilizado para corregir el error de odometría  $E_d$ . La trayectoria se ha exagerado.

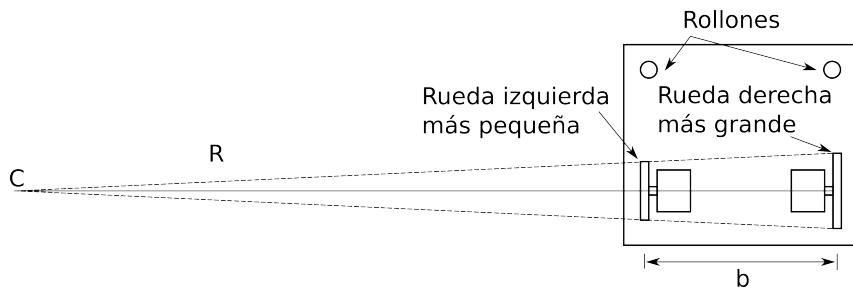


Figura 3.37: Radio de giro exagerado generado durante una trayectoria recta debido a diferentes diámetros de rueda de encoder.

Si el error  $E_d$  fuese nulo los puntos B y C deberían coincidir. El análisis utilizado para el cálculo de  $E_d$  es similar al realizado en [33] para el cálculo de error tipo B. Debido a la diferencia de diámetros el robot va a realizar una trayectoria curva de radio R y centro de giro C (ver figura 3.37).

Dicho radio se obtiene al partir del cálculo de la distancia L y del ángulo  $\beta$  de la siguiente manera:

$$L = \sqrt{(x_C - x_A)^2 + (y_C - y_A)^2} \quad (3.88)$$

$$\beta/2 = \arctan \frac{x_C - x_A}{y_C - y_A} \quad (3.89)$$

$$R = \frac{L/2}{\sin(\beta/2)} \quad (3.90)$$

El error  $E_d$  viene dado como

$$E_d = \frac{D_R}{D_L} = \frac{R + b/2}{R - b/2} \quad (3.91)$$

Donde el valor de  $b$  se obtiene de midiendo la distancia entre ejes de encoder utilizando un calibre.

Los factores de corrección de cada rueda se deducen de la ecuación (3.87) como

$$c_L = \frac{2}{E_d + 1} \quad (3.92)$$

$$c_R = \frac{2}{(1/E_d) + 1} \quad (3.93)$$

y se aplican al diámetro nominal de cada rueda como:

$$D_L = D_n \times c_L \quad (3.94)$$

$$D_R = D_n \times c_R \quad (3.95)$$

La versión experimental de este método consiste ir realizar los pasos anteriormente descritos e ir ajustando  $E_d$  partiendo de un valor  $E_d = 1,0$  utilizando aproximaciones sucesivas hasta obtener un error asumible. Esta aplicación experimental del método permite verificar la implementación de las correcciones de odometría y comprender mejor el fundamento de la corrección.

Aplicada la corrección del error  $E_d$  se vuelve a realizar el test con el fin de comprobar su efecto.

Por último, se vuelve a realizar la trayectoria AB con el fin de corregir el valor de diámetro nominal de ruedas de encoders  $D_n$ . El valor del diámetro nominal se encuentra en el coeficiente de conversión de distancia  $C_d$  (3.71). Debido a la deformación de la junta tórica de las ruedas de encoder el valor del diámetro nominal suele variar produciendo un error en la distancia recorrida por el robot. Notar que este error afecta por igual a ambas ruedas de encoder.

Para medir este error se vuelve a realizar la trayectoria AB. A partir de la medida de la distancia AB recorrida por el robot y de la medida de odometría del robot  $(x_r, y_r, \theta_r)$  se calcula un factor de corrección  $G_d$  para el coeficiente de distancia  $C_d$  como

$$G_d = \frac{d_r}{d_{AB}} \simeq \frac{y_r}{d_{AB}} \quad (3.96)$$

el cual se aplica al coeficiente de distancia de la siguiente forma

$$C'_d = C_d \times G_d. \quad (3.97)$$

### Medida y corrección del error $E_b$

El procedimiento para la medida del error  $E_b$  consiste en hacer girar al robot sobre su centro de giro. Dado que el cálculo del ángulo del robot (3.73)  $\theta$  es inversamente proporcional a la distancia entre ejes  $b$  el método utilizado busca amplificar el error producido por  $b$  girando un numero  $n$  de vueltas enteras.

Para la medida del error  $E_b$  se necesita una superficie lisa una regla larga, una escuadra y un cartabón. Al igual que en el método de medida de  $E_d$  puede utilizarse el campo de juego. Se recomienda utilizar una velocidad y aceleración baja con el fin de evitar deslizamientos.

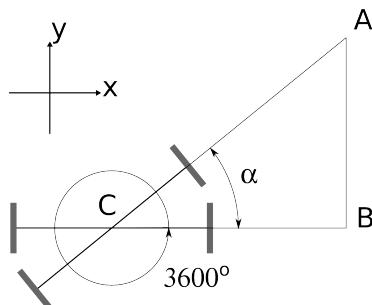


Figura 3.38: Método utilizado para medir el error de odometría  $E_b$ .

La figura 3.38 representa el el procedimiento a realizar, el cual consta de los siguientes pasos:

1. Posicionar el robot sobre el suelo y fijar la posición del robot  $(x, y, \theta)$  a  $(0, 0, 0)$ . Con ayuda de una regla y un marcador trazar una linea en el suelo paralela a la cara posterior del robot.
2. Llevar al robot a ángulo relativo  $\theta = 3600$  grados (10 vueltas).
3. Apuntar la medida absoluta de ángulo del robot  $\theta_r$  y volver a trazar una linea en el suelo paralela a la cara posterior del robot.
4. Construir el triángulo ABC y calcular el ángulo  $\alpha$ .

$$\alpha = \arctan\left(\frac{CA}{CB}\right) + \theta_r \quad (3.98)$$

El factor de corrección  $c_b$  para el error  $E_b$  viene dado por la expresión

$$c_b = \frac{\theta}{\theta \pm \alpha} = \frac{3600}{3600 \pm \alpha}$$

(3.99)

Donde  $\alpha$  suma o resta dependiendo de si el robot a girado más o menor de 3600 grados, respectivamente. Aplicado a la distancia nominal entre ejes  $b_n$  se obtiene el valor corregido

$$b = b_n \times c_b. \quad (3.100)$$

# Capítulo 4

## Sistema de balizas tipo faro

*I think work should be about making things work.*

*Better. Faster. Smaller. Smarter. So I build bridges between what's known and what's not. I tinker. I toil. I write poetically in an abundance of languages (including code). I hack. I dissect. I have an insatiable desire to un-complicate the complicated. I am easily inspired. I believe that just because it hasn't been thought of doesn't mean it won't be. Potential is my thrill ride. Imagination is my most-used tool. I am a maker, and I am what moves the world forward.*

Manifiesto *I am a maker*, Universidad Purdue

Los soportes destinados a sistemas de balizamiento dados por las reglas de Eurobot permiten implementar sistemas de posicionamiento absoluto o relativo con el fin de medir la posición de un robot o sus oponentes. Cada baliza puede ser activa o pasiva y las señales utilizadas para su detección pueden ser de luz (visible o infrarroja), de ultrasonidos, de radio o basada en marcas visuales [25].

Dado que la posición del robot se suele obtener mediante odometría, los soportes para balizas suelen ser utilizados principalmente para medir la posición de los oponentes. Aunque hay equipos que tienen desarrollado un sistema de posicionamiento absoluto a partir del cual, además de obtener la posición del oponente, realizan fusión sensorial con la odometría.

Este capítulo se centra en la medida de la posición relativa del oponente respecto al robot. Para ello lo más sencillo es utilizar el soporte para balizas de cada robot y el espacio destinado al sensores de balizas del robot. Se presenta un sistema basado en una baliza tipo faro, un sistema muy utilizado en Eurobot.

### 4.1 Principio de funcionamiento

El sistema de baliza tipo faro se compone de un sensor giratorio y una o varias balizas cilíndricas reflectantes. El sensor gira sobre el plano  $xy$  y emite una luz que es reflejada por balizas reflectantes cuando el sensor se encuentra enfrentado a una baliza. Como muestra la figura 4.1, el sensor puede estar montado en un cuerpo que gira y situado en la zona reservada a sensores de baliza del robot. Debido a la altura del soporte de las balizas de los robots oponentes el sensor ha de ser fijado con un ángulo  $\beta$  para así

detectar las balizas. Esto constituye un inconveniente que limita el rango de detección de la baliza entre una distancia mínima  $d_{min}$  y una máxima  $d_{max}$ .

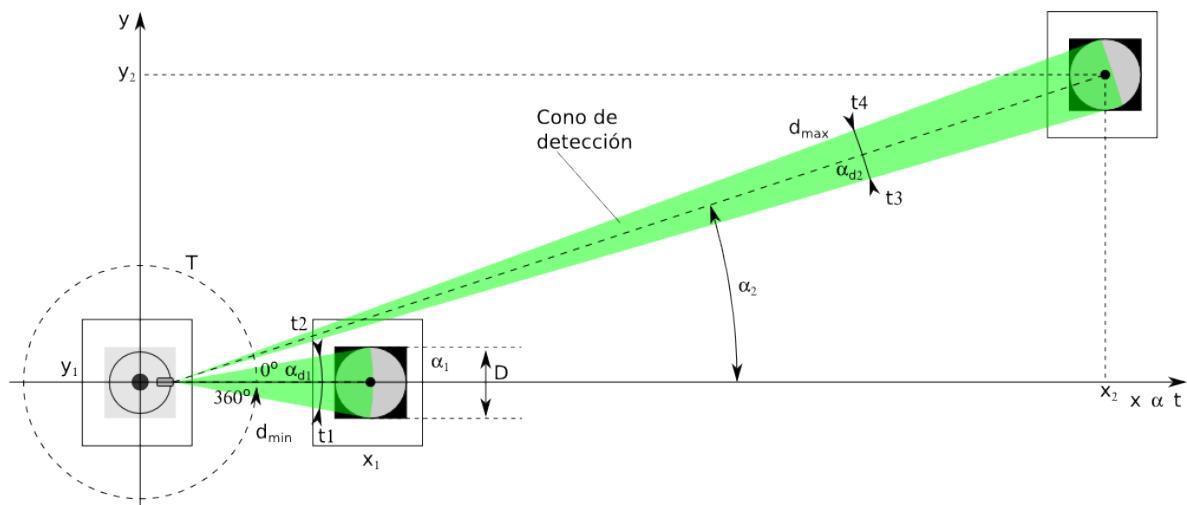
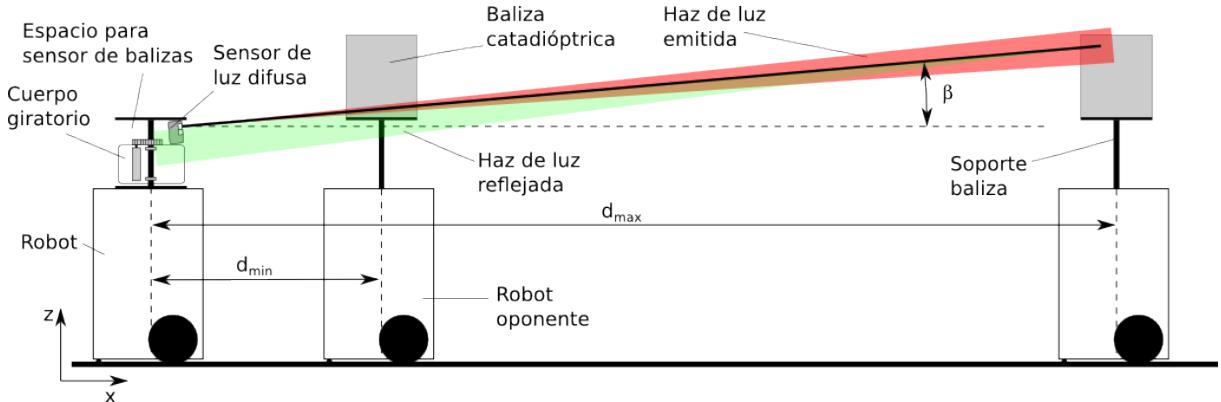


Figura 4.1: Principio de funcionamiento de una baliza tipo faro

La medida de distancia se basa en el principio por el cual un objeto cercano se ve más grande que uno lejano. En este caso (ver figura 4.1) a la baliza a la distancia  $d_{min}$  le corresponde un ángulo relativo de detección  $\alpha_d$  mayor que el de la baliza a la distancia  $d_{max}$ . Por otro lado, la medida de ángulo de la baliza se corresponde con la posición angular absoluta mitad de detección de la baliza  $\alpha$ .

La medida de dichos ángulos de detección puede ser realizada mediante un sensor de posición absoluta (un encoder) o a partir de la medida de los tiempos de detección de una baliza y el periodo de giro del sensor. Si se dispone de un encoder con la suficiente resolución la implementación por medida de posición resulta más sencilla y no necesita de un giro a velocidad constante. En caso contrario, puede realizarse una implementación basada en medida de tiempos que tiene el inconveniente de necesitar de un periodo de giro constante, pero añade la ventaja de que la resolución de medida es mucho mayor y por lo tanto el rango de detección también lo será.

#### 4.1.1 Medida de distancias y ángulos

La figura 4.2 muestra la señal obtenida del sensor al detectar las balizas. En una implementación basada en medida de tiempos, la medida de distancia a la baliza catadióptrica es proporcional al tiempo durante

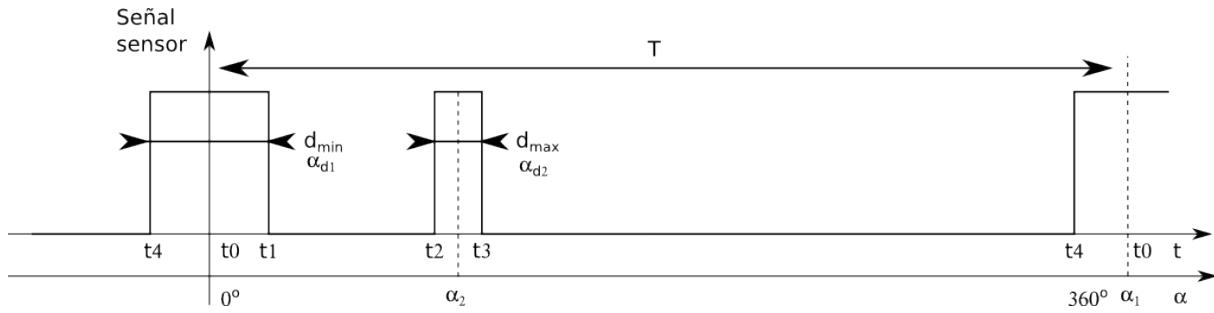


Figura 4.2: Señal dada por el sensor de luz al detectar balizas a diferentes distancias y ángulos

el cual el sensor detecta la baliza catadióptrica:

$$d \propto \frac{1}{(t_n - t_{n-1})} \vee t_n > t_{n-1} \quad (4.1)$$

El ángulo de detección relativo  $\alpha_d$  se puede obtener a partir del tiempo de detección relativo al periodo de vuelta  $T$  como

$$\alpha_d = \frac{(t_n - t_{n-1})}{T} \times 2\pi \text{ [rad]} \quad (4.2)$$

Aproximando la luz emitida por el sensor a una luz coherente (haz láser), teóricamente la distancia a la baliza catadióptrica se puede obtener a partir del ángulo de detección relativo  $\alpha_d$  y del diámetro de la baliza  $D$  como:

$$d = \frac{D/2}{\tan(\alpha_d/2)} \text{ [m]} \quad (4.3)$$

Esta aproximación permite estudiar el rango del ángulo de detección  $\alpha_d$  del sistema. Despejando  $\alpha_d/2$

$$\alpha_d/2 = \arctan\left(\frac{D/2}{d}\right) \text{ [rad]} \quad (4.4)$$

Así por ejemplo, para un rango de distancia  $d$  entre 30cm y 2,5m le corresponden ángulos de detección  $\alpha_d$  de 0,133rad y 0,016rad (7,6° y 0,92°). Para dichos ángulos la gráfica de la arcotangente se aproxima a una respuesta lineal. Por lo tanto en esta primera aproximación la medida del ángulo de detección es inversamente proporcional a la distancia de la baliza.

El modelo real de la medida de distancia es más complejo dado que la luz del sensor es una luz incoherente. La detección de la baliza va a depender del patrón de emisión y recepción del propio sensor. La luz reflejada por el catadióptrico depende del área proyectada sobre el mismo y su energía, y de las características del propio material catadióptrico. Además, no hay que olvidar que el sensor no está en el mismo plano que la baliza a detectar lo cual implica un comportamiento no lineal de la reflexión de la luz en los límites del rango de detección de la baliza.

Para una emisión incoherente, el área proyectada sobre la baliza viene dada por la definición de ángulo sólido  $\Omega$ , de forma que la superficie  $S$  se incrementa con el cuadrado de la distancia  $d$ :

$$\Omega = \frac{S}{d^2} \quad (4.5)$$

Teniendo en cuenta que la energía emitida por el sensor se reparte por la superficie  $S$ , se deduce que la energía proyectada sobre la baliza catadióptrica disminuye de forma inversamente proporcional con el cuadrado de la distancia. Lo cual completa la ecuación (4.4) de forma que la medida del ángulo detectado  $\alpha_d$ , al menos, disminuirá con el cuadrado de la distancia.

Por otro lado, la medida de ángulo de la baliza detectada es mucho más sencilla y se obtiene a partir de los tiempos de detección de la baliza y el periodo de vuelta  $T$  como:

$$\alpha = \frac{t_n - \frac{(t_n - t_{n-1})}{2}}{T} \times 360 [^{\circ}] \vee t_n > t_{n-1} \quad (4.6)$$

Una vez obtenida la distancia  $d$  y el ángulo  $\alpha$  a una baliza se puede calcular la posición relativa  $xy$  de la baliza a partir de las proyecciones con los ejes de la siguiente forma:

$$x = d \cos(\alpha) \quad (4.7)$$

$$y = d \sin(\alpha) \quad (4.8)$$

Y su posición absoluta se obtiene a partir de la posición absoluta del robot  $(x_r, y_r)$  como:

$$x_{abs} = x + x_r \quad (4.9)$$

$$y_{abs} = y + y_r \quad (4.10)$$

## 4.2 Implementación mecánica

La implementación mecánica de la baliza tipo faro puede implementarse de varias formas. En este caso se han realizado dos implementaciones que siguen dos líneas de diseño completamente diferentes:

1. Baliza de cuerpo giratorio
2. Baliza de espejo giratorio

En ambos diseños el efecto conseguido es el mismo: orientar la emisión y recepción del sensor en un rango de 360 grados. Los diseños implementados se muestran en la figura 4.3.

En ambos diseños el sensor utilizado es un sensor fotoeléctrico de reflexión difusa y de 60cm de rango. Este tipo de sensores permiten ajustar la detección de un objeto a una distancia fija variando la intensidad de la luz emitida. Esto permite ajustar dicha intensidad para que ningún objeto pueda ser detectado dentro del área del robot y que solo objetos con reflectante catadióptrico sean detectados más allá de las dimensiones del robot. No obstante, también se puede utilizar sensores retro-reflexivos preparados para detectar únicamente catadiópticos de un rango mayor.

Igualmente, ambos diseños utilizan un sensor fotoeléctrico de hendidura para medir el periodo de giro instantáneo y tomar la referencia de 0° de la medida de ángulo de la baliza. El sensor es activado por un final de carrera que coincide con el origen de 0°.

Los dos tipos de balizas implementadas utilizan las mismas balizas catadiópticas. La figura 4.4 muestra la baliza construida. Esta está realizada a partir de un cilindro de metacrilato de 8cm de diámetro exterior al que se le han añadido unas tapas en sus extremos y se ha forrado de catadióptico adhesivo.

### 4.2.1 Baliza de cuerpo giratorio

En el caso de la baliza de cuerpo giratorio (ver figura 4.3, imagen izquierda) el sensor junto con el resto de componentes de la baliza giran sobre una estructura fija anclada a la estructura del robot. Esta baliza constituye un sistema completamente autónomo. La obtención de las medidas es realizada a través de una comunicación inalámbrica Bluetooth con el robot.

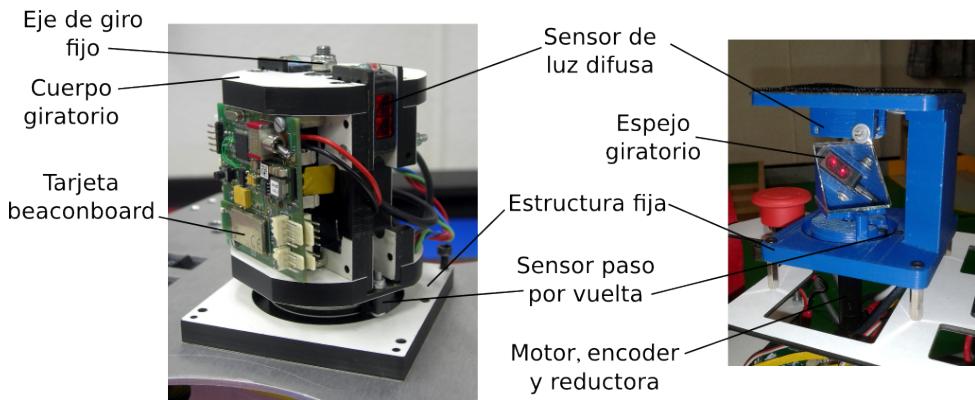


Figura 4.3: Implementaciones mecánicas de una baliza tipo faro. Baliza de cuerpo giratorio (izquierda). Baliza de espejo giratorio (derecha).



Figura 4.4: Baliza pasiva catadióptrica.

La estructura de esta baliza ha sido construida en Trespa (material similar a madera). Dispone de dos sensores de luz roja para la detección de balizas que permite duplicar la frecuencia de detección de una baliza. El giro se implementa a partir de un motor con reductora y encoder integrados que transmiten el movimiento a un engranaje solidario con el eje de giro fijo. Para disminuir la fricción el eje cuenta con rodamientos de bolas en sus extremos. En la parte superior del eje fijo de giro se coloca el soporte para la baliza del oponente. La baliza tiene además un sensor de herradura que permite medir el periodo de vuelta. Todos estos elementos se encuentran conectados una tarjeta electrónica alimentada a partir de una batería de 12V.

Tiene la ventaja de que los sensores tiene una visión directa con las balizas a detectar y aprovechan la propia óptica del sensor. Al estar situados en la parte superior de forma que están más cerca de la altura de los soportes de la baliza del oponente, lo cual permite disminuir el ángulo  $\beta$  y obtener un rango de detección muy amplio entre 30cm y 2,5m. El inconveniente de este diseño es que necesita tener un centro de masas simétrico de forma que al girar no se produzcan vibraciones, lo cual complica el diseño mecánico.

#### 4.2.2 Baliza de espejo giratorio

En el caso de la baliza de espejo giratorio (ver figura 4.3, imagen derecha) la luz emitida y reflejada del sensor detector de balizas es guiada mediante un espejo a 45 grados. De esta forma únicamente el espejo es el que gira, estando el resto de la estructura y componentes de la baliza fijos a la estructura del robot.

La estructura de la baliza está realizada en plástico mediante impresión 3D por aporte de material. Aunque el diseño es diferente la baliza dispone de los mismos elementos que la baliza de cuerpo giratorio, pero en este caso sólo se cuenta con un sensor de detección de balizas y la electrónica de control y batería son las utilizadas en el robot. El sensor utilizado en este caso es un sensor de luz infrarroja.

La ventaja de este diseño es principalmente su sencillez de mecánica, no obstante la alineación entre el espejo y el emisor del sensor constituye un punto muy importante. Si ambos no estuviesen correctamente alineados la medida de distancia y el rango de detección podrían variar en función del ángulo del espejo. El inconveniente de este diseño es debido a que la luz del sensor es guiada a través del espejo y debido a ello el rango de distancia obtenido es menor, entre  $30\text{cm}$  y  $80\text{cm}$ . Para reflejar la luz del sensor el espejo necesita estar más abajo que este, con lo cual la distancia con la altura a la que se encuentran las balizas a detectar es mayor, el ángulo  $\beta$  se incrementa y se obtiene un rango de detección menor. Además, al no tener el sensor visión directa con la baliza catadióptrica no se aprovechan las características de la óptica del sensor, existiendo perdidas por reflexión de la luz emitida y reflejada.

### 4.3 Implementación hardware y software

Como se ha visto en la sección anterior, aunque se han realizado dos implementaciones mecánicas diferentes ambas tienen los mismos elementos. Esto ha permitido reutilizar la misma arquitectura hardware y el mismo software para ambas balizas. Para la baliza de cuerpo giratorio se ha desarrollado la tarjeta *beaconboard* y para la baliza de espejo giratorio la tarjeta *mainboard v03*. Ambas implementan la misma arquitectura HW y SW representada por el diagrama de bloques de la figura 4.5, con la diferencia de que la tarjeta *mainboard v03* tiene además otros recursos destinados al control de la plataforma base del robot.

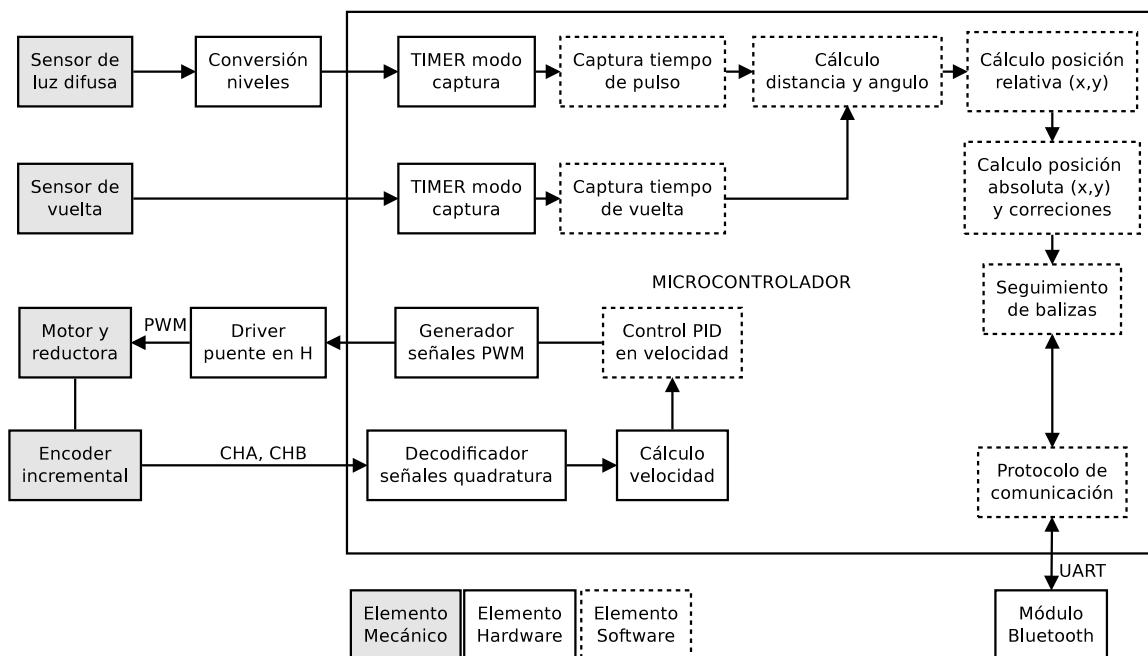


Figura 4.5: Diagrama de bloques HW y SW de una baliza tipo faro.

La arquitectura está basada en un microcontrolador de la familia dsPIC33 y se componen de 3 partes diferenciadas:

1. Control de velocidad de giro del la baliza o espejo.
2. Procesamiento de la medidas dadas por los sensores y calculo de la posición de una baliza.
3. Interfaz con la plataforma robótica base.

Las dos primeras partes son comunes en ambas balizas implementadas, la tercera únicamente es necesaria en el caso de la baliza de cuerpo giratorio ya que se trata de una tarjeta independiente que necesita comunicarse con la tarjeta de la plataforma robótica base.

### 4.3.1 Control de velocidad de giro

El motor que permite el giro de la baliza o del espejo es controlado mediante una señal PWM generada por el microcontrolador y amplificada por un driver en puente en H. La posición de dicho motor es medida a través un encoder conectado a su eje. Este encoder tiene como salida dos canales en cuadratura que son decodificados en el microcontrolador. El microcontrolador implementa un controlador PID en velocidad mediante el cual se garantiza un periodo de rotación constante.

El periodo de rotación de la baliza está fijado en  $6\text{ rev/s}$ . Dado que por cada vuelta se obtiene una medida de las balizas detectada, en el caso de la baliza de espejo giratorio el periodo de obtención de una medida es de  $167ms$ . Para el caso de la baliza de cuerpo giratorio, con dos sensores, dicho tiempo se reduce a la mitad,  $83ms$ .

### 4.3.2 Procesamiento de medidas y calculo de la posición

Las señales de los sensores de detección de balizas y de vuelta se encuentran conectados a entradas de interrupción asociadas a dos *timers* configurados en modo captura. En el caso del sensor de vuelta se detecta un único flanco de la señal y en el caso del sensor de balizas ambos flancos de la señal. A partir de la captura de un pulso válido (flanco de subida y uno de bajada) se obtiene la medida en cuentas del ancho de pulso *size* equivalente al ángulo relativo de  $\alpha_d$  de la detección de una baliza y su punto medio en cuentas *middle*, correspondiente con el ángulo de la baliza  $\alpha$ . Posteriormente estas medidas son utilizadas en el calculo de la distancia y el ángulo correspondiente a la baliza detectada.

La medida de ángulo de la baliza se obtiene mediante la función `get_angle` a partir de numero de cuentas del punto medio *middle* del ancho de pulso y del periodo instantáneo *period*:

```
static int32_t get_angle(int32_t middle, int32_t period, int32_t offset)
{
    int32_t ret_angle;

    ret_angle = (int32_t)(middle * 360.0 * MDEG / period);
    ret_angle = (ret_angle + offset*MDEG);

    /* XXX angle is -ret_angle because beacon turns clockwise */
    return (int32_t)(360-(ret_angle/MDEG));
}
```

La medida de distancia de la baliza se obtiene a partir de la función `get_dist_array`, para ello la medida de ancho de pulso en cuentas *size* es dividida por el número de cuentas del periodo *period*, obteniendo una medida de ancho de pulso *measure* independiente del periodo de giro de la baliza:

```
uint16_t get_dist_array(int32_t size, int32_t period)
{
    uint32_t measure;
    uint16_t index;
```

```

/* calculate measure */
measure = (uint32_t) ((size*100000)/period);

...
}

```

Notar que `size` es multiplicado por un factor  $1 \times 10^5$  para no perder precisión en la división.

Posteriormente, la distancia a la baliza  $d$  se obtiene direccionando el array `measure2distance` cuyas posiciones contienen la medida de distancia en cm. El índice `index` de dicho array se obtiene a partir de `measure` y los parámetros con los que se construyó el array durante la caracterización de la baliza.

```

uint16_t get_dist_array(uint8_t sensor, int32_t size, int32_t period)
{
    ...

    /* saturate to working range */
    if(measure > MEASURE_MAX)
        measure = MEASURE_MAX;
    if(measure < MEASURE_MIN) {
        measure = MEASURE_MIN;
        return DIST_ERROR;
    }

    /* calculate array index */
    index = (ARRAY_SIZE-1) - ((measure-MEASURE_MIN)/MEASURE_DELTA);

    /* return distance from array in mm */
    return (measure2distance[index]*10);
}

```

La obtención de la distancia mediante un *array* de datos permite obtener la medida sin penalizar en tiempo de cálculo.

Una vez obtenida la distancia y el ángulo relativos de la baliza detectada se obtiene su posición relativa y junto con la posición absoluta del robot, dada por la odometría, se obtiene la posición absoluta ( $x, y$ ) de la baliza. Normalmente la baliza se encuentra alineada con el centro de giro del robot y las medidas relativas de la baliza coinciden con las medidas relativas al robot. En caso de no ser así, tanto las medidas relativas como las medidas absolutas deben ser corregidas teniendo en cuenta el desplazamiento de la baliza respecto al centro de giro del robot.

Por último, dado que en un partido de Eurobot puede existir hasta dos robots oponentes, correspondientes a dos balizas reflectantes, se implementa un sistema de seguimiento de las balizas de dichos oponentes. Básicamente el seguimiento se realiza a partir de diferencia entre la posición de una baliza detectada frente a valores anteriores de las balizas de los oponentes. Se selecciona la diferencia menor y si se encuentra dentro de un radio máximo respecto a la posición anterior se actualiza la posición de la baliza correspondiente.

### 4.3.3 Interfaz con la plataforma base

En el caso de la baliza de cuerpo giratorio la baliza no tiene acceso directo a los datos de odometría del robot y ésta a su vez, tampoco tiene acceso a las medidas la baliza tipo faro. El intercambio de datos se produce mediante una interfaz serie inalámbrica implementada a partir de módulos de radio Bluetooth conectados a una UART de los microcontroladores de las tarjetas.

Mediante una comunicación por mensaje y respuesta el robot realiza peticiones periódicamente cada  $10ms$  a la baliza tipo faro de las balizas reflectantes detectadas. En dicho mensaje el robot al mismo tiempo envía su posición absoluta. La baliza tipo faro en el momento de recibir el mensaje utiliza la posición de odometría del robot para actualizar la posición absoluta de las últimas balizas detectadas y se la envía al robot en la respuesta. En el tiempo entre peticiones la baliza utiliza los últimos datos de odometría para sus cálculos.

## 4.4 Ajustes, modelado y calibración

A la hora de poner en marcha la baliza tipo faro es necesario realizar algunos ajustes con el fin de obtener el rango de medida de distancia máximo y ajustado a las necesidades. En el caso de la medida de ángulo, dependiendo de la implementación mecánica puede ser necesario ajustar el offset que define el ángulo  $0^\circ$ . Pero no es el caso de los dos tipos de balizas implementadas, ya que en ambas el sensor de medida de periodo de vuelta se encuentra sincronizado mecánicamente con la orientación  $0^\circ$ . Cabría un offset entre la interrupción de la señal del sensor y la captura del *timer* asociado, pero se puede considerar despreciable.

Una vez realizados los ajustes, la forma de obtener la medida de ángulo se corresponde con el modelo teórico (4.6). Respecto a la medida de distancias, se ha optado por obtener un modelado experimental.

El proceso de ajuste y modelado se aplica a cualquiera de los dos tipos de baliza desarrollados.

### 4.4.1 Ajustes del rango de medida

Los ajustes a realizar consisten en determinar la distancia mínima de detección de cualquier objeto y la distancia mínima y máxima de detección de una baliza.

La distancia mínima de detección de un objeto depende de la intensidad de emisión del sensor. Al ser un sensor de reflexión difusa cualquier objeto puede ser detectado por él. Es por ello que limitando la intensidad de emisión se puede fijar la distancia máxima a la que cualquier objeto no será detectado.

En el caso de el rango de distancia de detección de una baliza reflectante, este depende del ángulo  $\beta$  con el que el salga el haz de luz del sensor a las balizas reflectantes. En la distancia mínima el haz de luz se reflejará en la parte inferior de la baliza y en la distancia máxima en su parte superior. Fuera de dicho rango el haz de luz se escapará por la parte de abajo o de arriba del catadióptrico. Además, a la distancia máxima donde la duración del ancho de pulso detectado es el mínimo, entra en juego el tiempo de mínimo que necesita el sensor para detectar dicho pulso. Dicho requisito determina la velocidad de giro máxima y por tanto el periodo de medida de la posición de las balizas detectadas.

#### 4.4.1.1 Ajuste de la distancia mínima de detección

El criterio seguido en este ajuste es no detectar ningún objeto a una distancia mayor que los límites del robot. Por ejemplo, si el robot estuviese pegado a una pared, esta pared no debería detectarse.

El procedimiento seguido es el siguiente:

1. Con la baliza parada se sitúa perpendicular al sensor una hoja blanca (peor caso) de dimensiones suficientes para abarcar el haz de luz emitido.
2. Se ajusta la intensidad del sensor hasta obtener una detección nula y estable.

#### 4.4.1.2 Ajuste del rango de medida de distancia

El rango se determina a partir de la distancia mínima requerida. Dependiendo del caso dicha distancia puede variar. Para cubrir dicha distancia mínima de detección se utilizan sensores digitales de corto alcance situados en puntos estratégicos de las caras del robot. De esta forma la detección de oponentes y obstáculos queda cubierta. Si dichos sensores tienen un rango de  $30\text{cm}$ , la distancia mínima de detección de una baliza puede ser de  $20\text{cm}$ . De esta forma se solapan los rangos de detección, asegurando una detección segura.

Para este ajuste la baliza reflectante se sitúa alineada con la baliza tipo faro y a la altura del soporte de baliza del robot oponente. Es recomendable fabricar un robot de madera o cartón que simule un robot oponente. Este robot de madera puede ser utilizado posteriormente para las pruebas de estrategia además de para estos ajustes.

El procedimiento seguido es el siguiente:

1. Con la baliza parada se sitúa la baliza reflectante a la distancia mínima de detección.
2. Partiendo de un ángulo  $\beta = 0^\circ$  (sensor paralelo a la baliza reflectante) se empieza a aumentar el ángulo hasta que el haz de luz incida un 80 % en la baliza. En caso de ser un sensor de luz visible es fácilmente apreciable en condiciones de penumbra. Si se trata de un sensor de luz infrarroja, la luz puede ser observada utilizando una cámara digital. Se comprueba que el sensor detecta la baliza.
3. Se fija el ángulo del sensor mecánicamente.
4. Con la baliza parada se desplaza la baliza reflectante en linea recta hasta la distancia límite de detección. Se anota la distancia máxima de detección.
5. Con la baliza girando a la velocidad mínima y observando la medida de ancho de pulso de la baliza detectada.
6. Comprobar que se obtiene una medida de ancho de pulso en el rango de detección del sensor determinado en los pasos anteriores.
7. Situar la baliza reflectante a la medida máxima e ir incrementando la velocidad de giro de la baliza hasta el límite de detección de ancho de pulso. Fijar el periodo de giro de la baliza al valor máximo para el cual la medida de ancho de pulso sea estable.

Una vez realizados estos ajustes conviene realizar varias medidas de ancho de pulso a diferentes distancias observando la estabilidad y repetitividad de la medida.

#### 4.4.2 Modelado y calibración de la medida de distancia

Para realizar el modelado y calibración es necesario poder observar la medida de ancho y punto medio del pulso de la baliza detectada y el periodo instantáneo de giro de la baliza. Al igual que para los ajustes,

la baliza reflectante se sitúa alineada con la baliza tipo faro (ángulo de detección de  $0^\circ$ ) y a la altura del soporte de baliza del robot oponente.

Se realizan varias medidas en todo el rango de distancia de detección con una separación entre medidas de  $10\text{cm}$ . Se va a presentar como ejemplo el modelado realizado para la baliza de cuerpo giratorio. Estas son de las medidas obtenidas en un rango de  $20$  a  $220\text{ cm}$  para dicha baliza:

```
276.676: period = 02315 / size = 00126 / middle = 01192 (x0.1)
294.887: period = 02313 / size = 00123 / middle = 01173 (x0.1)
325.976: period = 02319 / size = 00118 / middle = 01171 (x0.1)
349.396: period = 02317 / size = 00116 / middle = 01170 (x0.1)
368.197: period = 02314 / size = 00113 / middle = 01167 (x0.1)
388.792: period = 02314 / size = 00110 / middle = 01165 (x0.1)
412.188: period = 02315 / size = 00105 / middle = 01166 (x0.1)
431.161: period = 02313 / size = 00099 / middle = 01165 (x0.1)
445.513: period = 02315 / size = 00092 / middle = 01165 (x0.1)
467.730: period = 02316 / size = 00085 / middle = 01165 (x0.1)
482.107: period = 02314 / size = 00079 / middle = 01163 (x0.1)
498.253: period = 02313 / size = 00073 / middle = 01162 (x0.1)
515.284: period = 02313 / size = 00067 / middle = 01160 (x0.1)
532.144: period = 02313 / size = 00062 / middle = 01161 (x0.1)
551.264: period = 02312 / size = 00058 / middle = 01161 (x0.1)
566.525: period = 02314 / size = 00052 / middle = 01161 (x0.1)
584.589: period = 02313 / size = 00048 / middle = 01161 (x0.1)
602.529: period = 02315 / size = 00044 / middle = 01161 (x0.1)
625.754: period = 02313 / size = 00038 / middle = 01160 (x0.1)
643.252: period = 02317 / size = 00031 / middle = 01162 (x0.1)
657.924: period = 02311 / size = 00022 / middle = 01159 (x0.1)
```

A partir de la medida de periodo `period` y punto medio `middle` se obtiene la medida de ángulo. Dicha medida de ángulo se utiliza para corregir la distancia efectiva de medida ya que la baliza reflectante, colocada a mano, pudo no estar perfectamente alineada con el ángulo cero grados. Observar que en las medidas obtenidas `middle` es la mitad de `period`, cuando debería ser un valor cercano a cero o al valor de `period`, correspondiente a un ángulo de  $0^\circ$ . Esto indica que las medidas se realizaron con la baliza girada un ángulo de  $180^\circ$ .

Por otro lado, la medida de ancho de pulso `size` se procesa de la forma que se ha descrito en la sección 4.3.2 para obtener una medida de ancho de pulso no dependiente del periodo y escalada:

```
measure = (uint32_t)((size*100000)/period);
```

La distancia corregida y la medida de ancho de pulso procesada se utiliza para modelar y calibrar la medida de distancias. Para ello se obtiene el polinomio de menor orden que se ajusta a estas medidas y que se corresponde con el modelo de medida de la baliza. La figura 4.6 muestra gráficamente las muestras obtenidas en la medida, el polinómico calculado y el error entre ambos.

Se observa que la respuesta se aproxima a una función lineal pero se pueden diferenciar 3 zonas levemente curvas. Aunque se podía haber aproximado a la ecuación de una recta se ha preferido tratar de preservar fehacientemente el modelo obtenido.

Dado el elevado orden del polinomio ( $N = 6$ ), se ha optado por implementar un método de obtención de la medida de distancia basado en una tabla o *array* de valores de distancias y cuyo índice es calculado

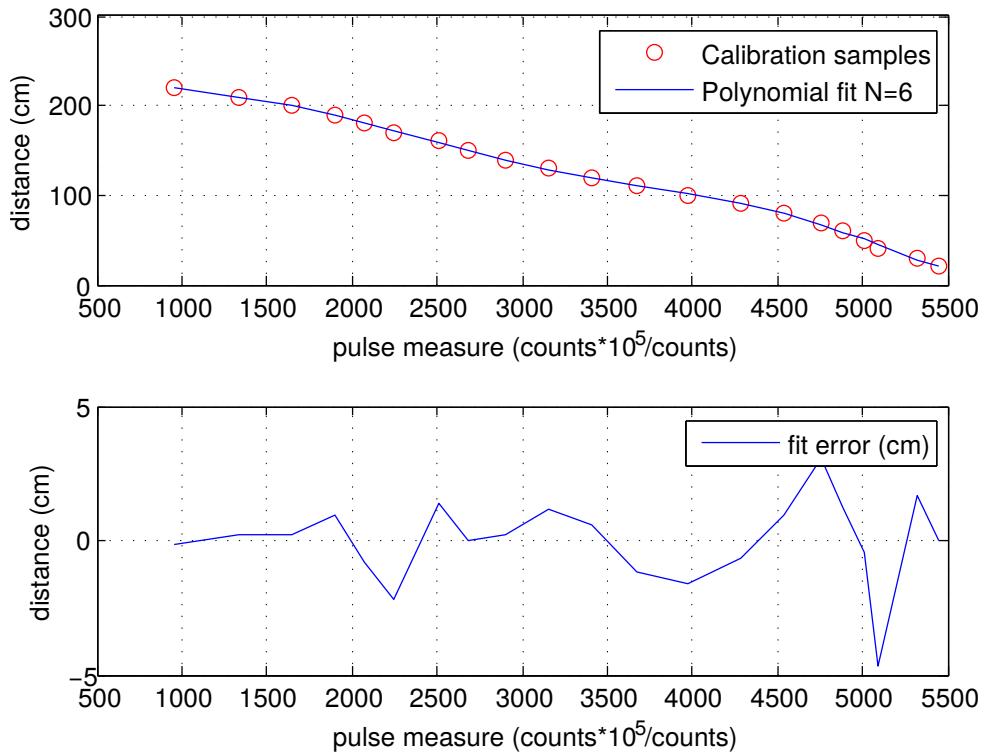


Figura 4.6: Modelo continuo de medida de distancia aproximado a un polinomio

a partir de la medida de ancho de pulso *measure*. Para ello utilizando el polinomio calculado se han obtenido un grupo de muestras suficientes para garantizar una resolución 1cm en la medida de distancias. Dicho grupo de muestras está definido por el valor mínimo y máximo de ancho de pulso (correspondientes a la distancia máxima y mínima, respectivamente) y un incremento de ancho de pulso (que define la resolución en la medida de distancia). El valor de incremento de ancho de pulso es el mínimo para garantizar la resolución de 1cm y genere con un numero datos almacenable en la memoria del microcontrolador.

El resultado se muestra en la figura 4.7, el conjunto de datos de la tabla de distancia obtenido es de 561 muestras de distancia y la diferencia entre muestras en ning n caso supera el cm.

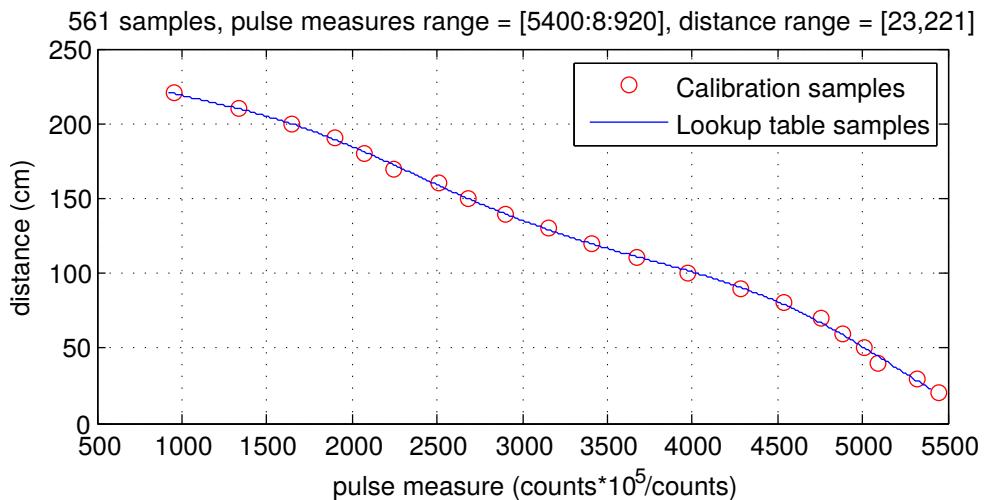


Figura 4.7: Discretizaci n del modelo continuo de medida de distancia para su mapeado en un *array* de datos

# Capítulo 5

## Desarrollo hardware

*No entiendes realmente algo a menos que seas capaz de explicárselo a tu abuela.*

Albert Einstein

El hardware a desarrollar para un robot de Eurobot no suele ser muy complicado, su arquitectura es la de un sistema embebido donde la mayoría de los elementos con los que interactuar, actuadores y sensores, suelen ser digitales. Dado que el robot ha de ser capaz de realizar varias cosas al mismo tiempo, como moverse y manipular elementos, es normal que el hardware tenga una arquitectura distribuida de varios microcontroladores o procesadores comunicados entre si.

En el año 2010 se definió una arquitectura hardware con el fin de ser reutilizada en el desarrollo de robots de Eurobot. Esta arquitectura vino dada a partir de los recursos necesarios para implementar las dos partes en las que se puede dividir un robot de Eurobot: la plataforma robótica base y los sistemas mecánicos de manipulación de elementos de juego. Los requisitos de la primera son conocidos, en cambio, los recursos de la segunda dependen de la temática de cada año de Eurobot.

### 5.1 Sobre sensores y actuadores

La experiencia de los integrantes del equipo de Eurobot y el estudio de los robots de otros equipos ayudó a determinar los tipos de sensores y actuadores que podrían llegar a ser utilizados en el desarrollo de robots de Eurobot. Hay que tener en cuenta que un robot de Eurobot es como una pequeña línea de montaje automatizada en miniatura, en continuo movimiento y cuyas condiciones exteriores son variables.

Respecto a los sensores se obtuvieron varias conclusiones:

- La utilización de sensores en aplicaciones fuera de su rango de aplicación no son fiables y antes o después constituyen un problema. En años anteriores al 2010 se utilizaron sensores fotoeléctricos como el GP2D12 (un sensor de distancia analógico), como sensores de detección de obstáculos. Estos sensores o similares están pensados para ser utilizados en entornos cerrados protegidos de focos de luz intensa, como por ejemplo el interior de una fotocopiadora. Durante un partido de Eurobot la propia iluminación del campo de juego puede influir en ellos.
- Los sensores a utilizar han de ser sensores diseñados específicamente para la aplicación y para el uso que se les va a dar. Han de ser robustos, reemplazables y fiables.

- El mercado de sensores industriales, destinados a líneas de producción automatizadas, constituye un mercado de referencia para los sensores de un robot de Eurobot.
- La principal razón por la que se utilizan sensores no específicos para la aplicación es el coste. La experiencia ha demostrado que merece la pena tratar de conseguir sensores específicos aunque tengan un coste mayor.
- Los sensores industriales son para toda la vida y pueden reutilizarse en cada robot.
- El mercado de segunda mano o chino (eBay, Aliexpress, Alibaba, Taobao) permite conseguir sensores industriales a menor coste. Es un compromiso entre la compra a demanda de las especificaciones de diseño, de coste elevado, y la fiabilidad de los sensores a utilizar. No obstante, sobre todo en el mercado chino, hay que tener cuidado con las falsificaciones.

Los actuadores eléctricos como motores y servomotores son los más comunes, aunque también se utilizan actuadores neumáticos o de vacío. Respecto a los actuadores, las conclusiones van en la misma linea de los sensores. Además, hay que tener en cuenta que los actuadores tienen un desgaste mecánico:

- Evitar la reutilización o utilización de actuadores de aplicación no específica o cuyas características no sean las adecuada para la aplicación. Esto puede derivar en unos requisitos del hardware mayores, complicar el control de los mecanismos u obtener un rendimiento mecánico insuficiente. En años anteriores se utilizaron motores de taladradoras o atornilladores eléctricos como motores de tracción del robot. Estos motores tienen un rendimiento muy pobre y un consumo muy elevado que implica una electrónica de control de alta potencia. Además, el uso de este tipo de motores para aplicaciones de control de posición o de velocidad, complican el ajuste de los controladores debido a respuestas no lineales, que en el caso de un atornillador no son importantes, pero para un robot de Eurobot si.
- Evitar utilizar actuadores cuyas especificaciones sean incompletas.
- Evitar utilizar actuadores de materiales de baja calidad. Especialmente el de las reductoras de los motores. Muchas veces constituye la parte más débil en un sistema mecánico, sobretodo en los servomotores.
- Al igual que con los sensores, los motores, si se cuidan adecuadamente, pueden ser reutilizados de robot en robot.
- El mercado de segunda mano constituye un punto intermedio entre el coste y las especificaciones, rendimiento y calidad del motor a utilizar.

Sin duda, el actuador más versátil y utilizado en la implementación de sistemas mecánicos es el servomotor o comúnmente llamado *servo*. Un motor de par elevado y controlado en posición que simplifica la implementación de sistemas mecánicos. Los más conocidos son los servos estándar utilizados ampliamente en aeromodelismo y radio-control, pero también existen servos industriales de un rango de características muy amplio. Para controlar estos servos sólo se necesita de una señal digital de 20Hz modulada en ancho de pulso (PWM).

En los últimos 6 años, han aparecido en el mercado de la robótica educativa servos que podrían denominarse inteligentes. Estos servos tienen una interfaz digital y son controlados mediante comandos. Además, mediante dicha interfaz es posible obtener información de su estado (velocidad, consumo, temperatura, par, entre otros). Es el caso de los actuadores AX12 de la marca Dynamixel. Estos actuadores pueden ser conectados en cadena simplificando así el cableado de los mismos. Además, tienen una interfaz serie half-duplex de hasta 1 Mbps que permite un control prácticamente simultáneo de hasta 254 actuadores.

## 5.2 Arquitectura hardware

Como se ha comentado en la sección anterior, la arquitectura HW se ha definido a partir del tipo de actuadores y sensores, y de los sistemas a implementar con ellos: la plataforma robótica base y los sistemas mecánicos de manipulación de elementos del juego.

El elemento principal de la arquitectura HW es un microcontrolador. Dicho microcontrolador tiene parte de sus recursos dedicados a la implementación de la plataforma robótica base y el resto a los sistemas mecánicos. Dado que el número de recursos de estos es variable y la carga de procesamiento no es conocida, una de las especificaciones de la arquitectura es la de una interfaz de expansión de recursos de entrada/salida (E/S) y de procesamiento. Es decir, la implementación de un sistema de procesamiento distribuido.

La figura 5.1 muestra el diagrama de bloques de la arquitectura HW. Dos microcontroladores se encuentran conectados mediante un bus I2C del que cuelga además un expansor E/S. La utilización de un bus I2C permite una implementación distribuida maestro-esclavo o multi-maestro. Ambas implementaciones permiten compartir los recursos de procesamiento y de E/S entre los dispositivos del bus.

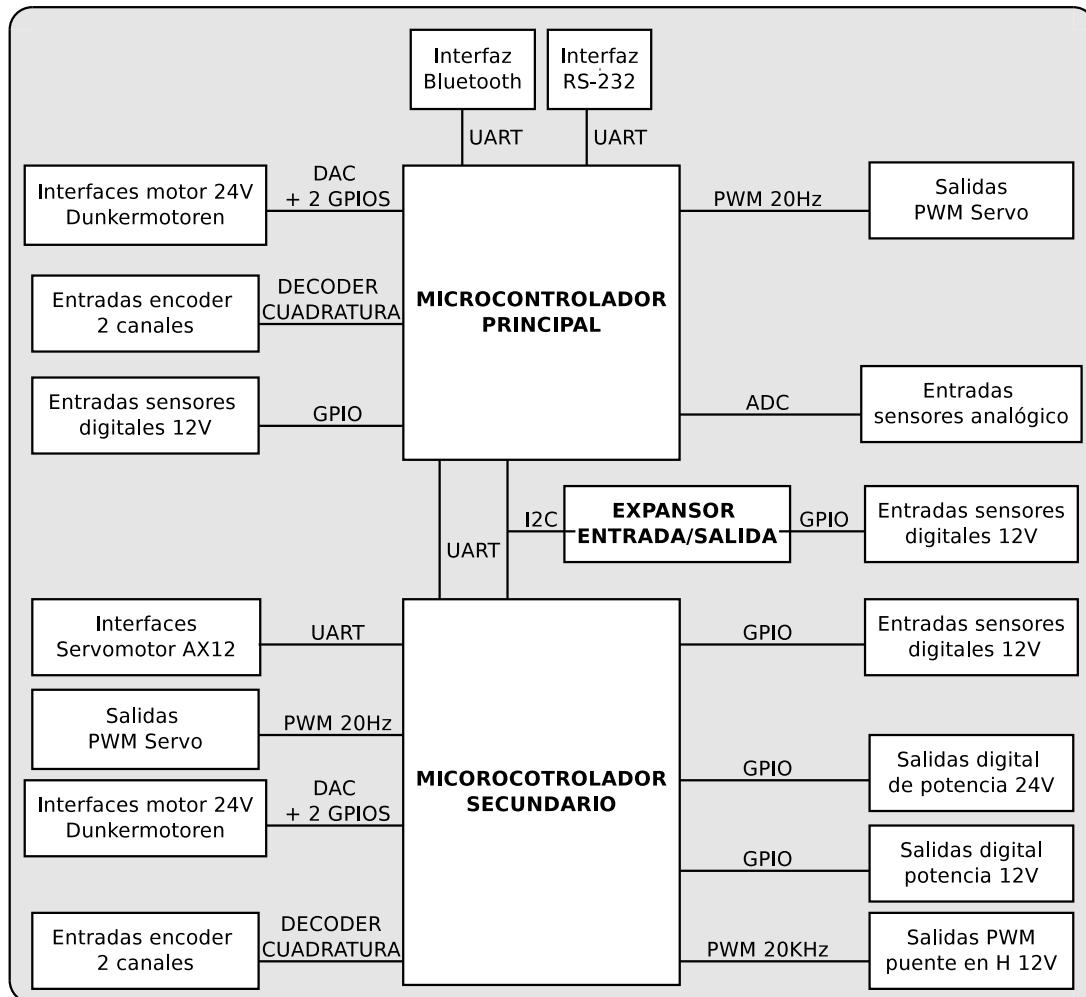


Figura 5.1: Arquitectura hardware principal

Una característica muy importante de la arquitectura HW es una interfaz de depuración independiente a través de la cual poder controlar, monitorizar y probar el hardware y el software. Para ello la arquitectura incluye una interfaz serie RS-232 conectada a una UART del microcontrolador principal. Este a su vez

se encuentra conectado en cadena por una segunda UART con el microcontrolador secundario y así sucesivamente con otros microcontroladores. Esto permite acceder a cualquier microcontrolador mediante un *bypass* de los datos.

Otro aspecto que se tuvo en cuenta fue la comunicación con la baliza tipo faro y con otro posible robot o sistema. Para ello la arquitectura cuenta con una interfaz inalámbrica Bluetooth conectada a una UART del microcontrolador principal. Mediante esta interfaz es posible implementar hasta 16 canales serie independientes en una configuración tipo estrella.

Por otro lado, a partir del estudio de los posibles sensores y actuadores industriales del mercado, y de algunos de los que ya se disponía, se determinó el tipo de recursos de entrada y salida necesarios, así como las interfaces necesarias en el microcontrolador. El resultado se muestra en las tablas 5.1 y 5.2.

Tabla 5.1: Tipos de actuadores y recursos E/S del microcontrolador

Tipo de actuador	Recurso E/S del microcontrolador
Motores DC	Salida PWM >20KHz
Motores Dunkermotoren	Salida analógica, DAC
Servomotores estándar	Salida PWM 20Hz
Servomotores AX12	Intefaz UART half-duplex 1Mbps
Actuadores ON/OFF	Salida digital (GPIO)

Tabla 5.2: Tipos de sensores y recursos E/S del microcontrolador

Tipo de sensor	Recurso E/S del microcontrolador
Encoders digitales con salida en cuadratura	Decodificador de modulo y signo y contador)
Sensores digitales ON/OFF	Entrada digital (GPIO)
Sensores analógicos	Entrada analógica (DAC)
Sensores inteligentes	Bus de comunicación (I2C, UART)

La cantidad de los recursos de cada tipo y su distribución entre los dos microcontroladores se define a partir de los requisitos de la plataforma robótica base y los sistemas mecánicos de manipulación.

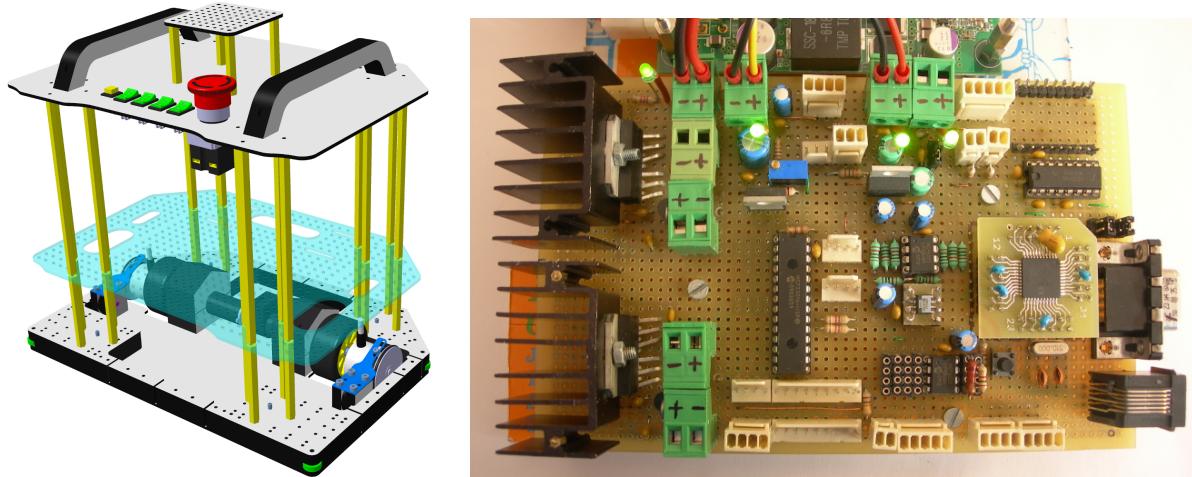


Figura 5.2: *Dummy robot* y tarjeta *dspic33\_protoboard*

Esta arquitectura HW fue implementada inicialmente en tarjeta prototipo *dspic33\_protoboard*, mediante la cual se desarrolló un prototipo de plataforma robótica base, el *Dummy robot* (ver figura 5.2). Esta tarjeta implementaba la arquitectura basada en un único microcontrolador e incluía todas las interfaces y recursos especificados anteriormente.

Una vez validada la arquitectura, ésta ha sido implementada de diferentes formas dependiendo de las necesidades del robot a desarrollar. En el caso del robot principal ha sido necesario utilizar dos microcontroladores, mientras que en el caso del robot secundario con un microcontrolador ha sido suficiente. En el caso del robot principal la implementación se dividió en 3 tarjetas electrónicas, mientras que en el robot secundario se desarrolló una única tarjeta que además controlaba la baliza tipo faro. Independientemente de la implementación, la arquitectura se ha mantenido en todos los robots desarrollados desde el año 2010, lo que ha posibilitado reutilizar el desarrollo HW y SW, reducir el tiempo de desarrollo y aumentar la fiabilidad de los robots.

### 5.2.1 Plataforma robótica base

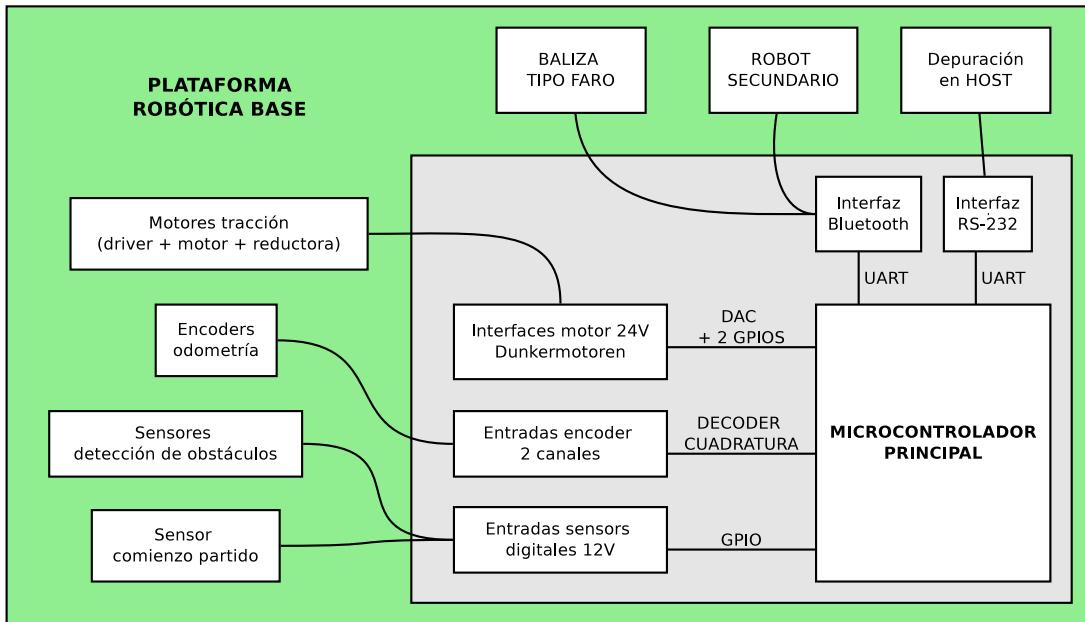


Figura 5.3: Recursos de hardware destinados a la plataforma robótica base

La figura 5.3 muestra los recursos de la arquitectura HW destinados a la plataforma robótica base. La elección del microcontrolador vino dada por las necesidades de la plataforma robótica, concretamente por los motores de tracción y los encoders.

Con anterioridad al diseño del HW se disponían de motores Dunkermotoren, destinados a la tracción del robot o a sistemas mecánicos de alto par. Estos motores incluyen el driver de control y una reductora mecánica. La velocidad se controla mediante una señal analógica de rango ajustable entre 4 y 10V, y el sentido de giro mediante una entrada digital. Además tienen una señal de freno.

Por otro lado, se disponía de encoders HENGSTLER de referencia RI41-3600, destinados a implementar la odometría y el control en posición del robot, o cualquier otro sistema mecánico realimentado. Estos encoders son de 3600 pulsos/vuelta y de dos canales de salida en cuadratura (desfasados 90 grados). La decodificación de dichas señales permite determinar el sentido de giro del encoder y obtener una medida de la posición de una resolución 4 veces mayor a la de las señales. La decodificación de las señales de los encoders constituyen un factor muy importante, sobre todo en la implementación de la odometría. Esta decodificación se puede realizar por SW por HW. El primer caso no es recomendable dada la carga de procesamiento. Lo ideal es realizar una decodificación por HW. Aunque existen dispositivos específicos para la decodificación HW de las señales en cuadratura, no son fáciles de conseguir. En este caso la

decodificación de los encoders se realizó mediante un microcontrolador con módulos decodificadores HW integrados.

El microcontrolador utilizado es el dsPIC33FJ128MC804 de la familia dsPIC33 de Microchip y pensado para aplicaciones de control de motores (familia MC). Este microcontrolador incluye 2 decodificadores de encoders en cuadratura y un conversor digital analógico (DAC) que cumplen con los requisitos de los motores Dunkermotoren y los encoders planteados anteriormente.

Los microcontroladores dsPIC33 tienen una arquitectura Harvard de 16 bits con una capacidad de procesamiento de 40 MIPS. Como se puede observar en la tabla 5.3, la familia MC dispone los recursos necesarios para implementar la arquitectura HW especificada. Dado que la salida analógica es un requisito indispensable, entre las posibles opciones se utilizó la de mayor memoria de programa y memoria RAM.

Tabla 5.3: Recursos de la familia de microcontroladores dsPIC33FJXMC. Tabla obtenida del datasheet del fabricante [34]

Device	Pins	Remappable Peripheral														I/O Pins	Packages					
		Program Flash Memory (Kbyte) <sup>(1)</sup>	RAM (Kbyte) <sup>(1)</sup>	Remappable Pins	16-bit Timer <sup>(2)</sup>	Input Capture	Output Compare Standard PWM	Motor Control PWM (Channels) <sup>(3)</sup>	Quadrature Encoder Interface	UART	SPI	ECAN™	External Interrupts <sup>(4)</sup>	RTCC	I <sup>2</sup> C™	CRC Generator	10-bit/12-bit ADC (Channels)	6-pin 16-bit DAC	Analog Comparator (2 Channels/Voltage Regulator)			
dsPIC33FJ128MC804	44	128	16	26	5	4	4	6, 2	2	2	2	1	3	1	1	1	9	1	1/1	11	35	QFN TQFP
dsPIC33FJ128MC802	28	128	16	16	5	4	4	6, 2	2	2	2	1	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ128MC204	44	128	8	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ128MC202	28	128	8	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ64MC804	44	64	16	26	5	4	4	6, 2	2	2	2	1	3	1	1	1	9	1	1/1	11	35	QFN TQFP
dsPIC33FJ64MC802	28	64	16	16	5	4	4	6, 2	2	2	2	1	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ64MC204	44	64	8	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ64MC202	28	64	8	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ32MC204	44	64	8	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ32MC304	44	32	4	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ32MC302	28	32	4	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S

**Note** 1: RAM size is inclusive of 2 Kbytes of DMA RAM for all devices except dsPIC33FJ32MC302/304, which include 1 Kbyte of DMA RAM.

2: Only four out of five timers are remappable.

3: Only PWM fault pins are remappable.

4: Only two out of three interrupts are remappable.

## 5.2.2 Sistemas mecánicos

Los recursos del microcontrolador principal no asignados a la plataforma robótica son utilizados en la implementación de los sistemas mecánicos. En el caso de no ser suficientes, serán necesarios los recursos de un segundo microcontrolador. El tipo y número de sensores y/o actuadores a utilizar a priori es

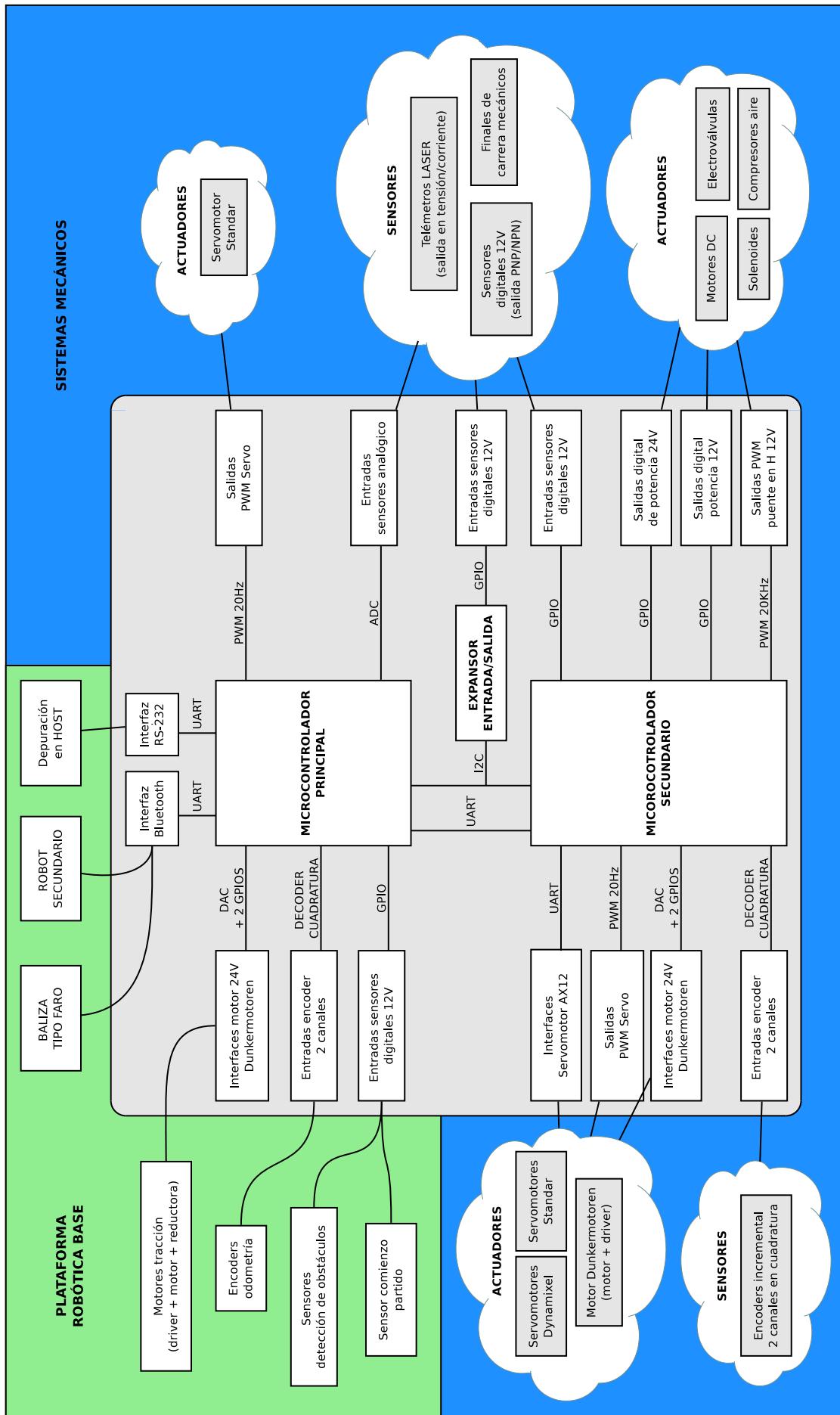


Figura 5.4: Recursos hardware destinados a la plataforma robótica y a los sistemas mecánicos

desconocido. Dependerán en última medida del diseño específico de los sistemas mecánicos destinados a manipular los elementos de juego de la temática de Eurobot.

La figura 5.4 da una idea del tipo de sensores y actuadores posibles. De cara a la implementación de los sistemas mecánicos es preferible utilizar sensores digitales a los analógicos. Es por ello que la E/S digital se encuentra reforzada mediante un expansor I2C de E/S. No obstante, en el mismo bus I2C también tiene cabida la expansión de E/S analógica.

Al igual que el microcontrolador principal, el secundario es también un dsPIC33FJ128MC804, lo que permite la implementación de sistemas mecánicos basados en el motor Dunkermotoren y encoders de salida en cuadratura. Además, el dsPIC33 dispone de salidas PWM específicas para el control de motores DC.

Por otro lado, los servomotores estándar de los sistemas mecánicos son controlados a partir de la señal PWM generada mediante *timers* en modo salida por comparación. Respecto a los servos AX12, estos se manejan utilizando uno de los módulos UART del microcontrolador. Cabe destacar, que la implementación half-duplex se ha realizado internamente al microcontrolador utilizando la característica de mapeado dinámico de pines del mismo. Un mismo pin físico del microcontrolador hace las veces de pin de transmisión y de pin de recepción. El ahorro de recursos E/S es notable, una implementación externa necesita de 3 pines y dos *bufferes* externos, mientras que la implementación interna utiliza únicamente un pin.

### 5.2.3 Esquema de alimentación

La alimentación de hardware, motores y sensores del robots se obtiene a partir de una batería de 24V. A partir de dicha tensión se obtienen las tensiones necesarias para cada elemento. La tabla 5.4 muestra las tensiones y potencia características asociada a los elementos más representativos del robot que constituyen una carga a alimentar.

Tabla 5.4: Elementos de carga representativos relacionados con el esquema de alimentación un robot de Eurobot

Elemento	Tensión (V)	Potencia máx./u (w)
Microcontrolador	3,3	0,8
Lógica E/S	5	0,2
Servomotores estandar	5	3
Sensores	12	0,3
Motores DC	12	10
Servomotores AX12	12	8,6
Motores tracción	24	30

En definitiva se necesitan tensiones de 24V, 12V, 5V y 3,3V. Todas ellas son obtenidas a partir de un esquema de alimentación con una distribución en árbol. Como se puede ver en la figura 5.5, la tensión de 24V se obtiene directamente de la tensión de batería, la cual es regulada mediante sendas fuentes de alimentación para obtener 12V y 5V. Y ésta tensión de 5V a su vez es regulada por una tercera fuente de alimentación para obtener 3,3V.

Notar que todo el árbol de alimentación cuelga del bloque interruptor de emergencia y del bloque fusible. Por normativa, el robot ha de llevar un interruptor de emergencia visible y rojo (la seta de emergencia). En caso de accionarse toda la alimentación del robot quedaría deshabilitada. El fusible tiene también una función de seguridad y se dimensiona para proteger la batería en caso de cortocircuito de la tensión de 24V.

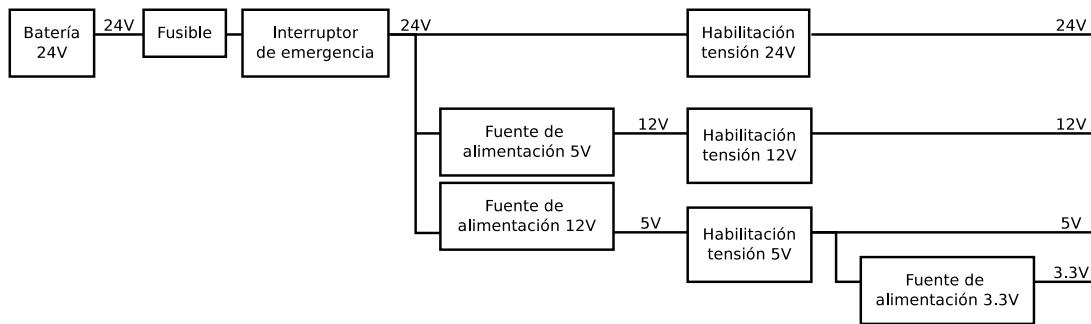


Figura 5.5: Arquitectura hardware del árbol de alimentación del robot de Eurobot

Por otro lado, cada una de las tensiones de alimentación tiene asociado un bloque de habilitación independiente. Esto permite utilizar sólo las tensiones de alimentación necesarias durante las diferentes fases de implementación del robot o ayudar a depurar algún problema relacionado con la alimentación, deshabilitando los dispositivos asociados a una o varias tensiones.

Respecto a los requisitos de potencia, como se puede observar en la tabla 5.4 estos vienen dados principalmente por los actuadores (motores y servomotores) y sensores. La potencia de la electrónica apenas tiene peso.

Por ejemplo, una estimación de la potencia máxima del robot P.Tinto (Eurobot 2015) se muestra en la tabla 5.5.

Tabla 5.5: Estimación de la potencia máxima del robot P.Tinto (Eurobot 2015)

Elemento	Tensión (V)	P. máx./u (w)	Cantidad	P. max. total (w)
Microcontrolador	3,3	0,8	2	1,6
Lógica E/S	5	0,2	1	0,2
Servomotores estandar	5	3	5	15
Sensores	12	0,3	13	3,9
Motores DC	12	10	1	10
Servomotores AX12	12	8.6	11	94.6
Motores tracción	24	30	2	60

Sumando las potencias de cada tensión de alimentación y de cada rama, se obtiene la potencia necesaria de cada bloqueo o etapa de alimentación presentado en la tabla 5.6.

Tabla 5.6: Estimación de la potencia máxima del robot P.Tinto (Eurobot 2015)

Bloque de alimentación	P. máx (w)	I máx. (A)
F. alimentación 3,3V	1,6	0,48
F. alimentación 5V	16,8	3,36
F. alimentación 12V	108,5	9
Batería 24V	185,3	7,72

Comentar que este robot es de los robots con más sensores y servomotores de los desarrollados para la competición Eurobot. Hay que tener en cuenta que estos valores son para el caso de que todos los elementos estén consumiendo su potencia máxima al mismo tiempo. En la práctica, no todos los servomotores AX12 estarán demandando su máxima potencia. Aun así, cada bloque de alimentación ha de elegirse teniendo en cuenta dicha potencia máxima.

Por último, a la hora de implementar el esquema de alimentación tipo árbol, para cada tensión de alimentación (3,3V, 5V, 12V y 24V) se ha de implementar una distribución en estrella de forma que se minimice la impedancia común entre dispositivos de diferente potencia y misma tensión de alimentación.

Por ejemplo, en el caso de la tensión de 5V, deben de implantarse dos caminos de alimentación diferenciados para la lógica E/S y los servomotores estándar, y cuyo punto de unión se sitúe lo más cerca de la fuente. En este caso de la fuente de alimentación de 5V. De esta forma se minimiza el ruido producido por la corriente de los servomotores en las líneas de alimentación de la lógica E/S.

### 5.3 Implementación HW del robot principal

El robot principal de Eurobot suele tener asociado un mayor número de motores y sensores que el robot secundario debido principalmente a que el perímetro máximo permitido es mayor. Esto hace que el robot principal implemente un mayor numero de sistemas mecánicos y más complejos que los que pueda tener el robot secundario.

La implementación de la arquitectura HW descrita anteriormente, para el caso de un robot principal, se ha dividido en 5 tarjetas electrónicas:

#### Tarjeta principal

Implementa la mayor parte de los bloques de la arquitectura HW. Contiene dos microcontroladores, uno principal destinado al control de la plataforma robótica y estrategia de juego, y uno secundario destinado al control de los sistemas mecánicos.

#### Tarjeta sensores

Implementa la expansión de entradas digitales para sensores.

#### Tarjeta sistemas de vacío

Implementa la expansión de salidas de potencia. Específicamente de salidas de control para actuadores de sistemas de vacío (bombas de vacío y electroválvulas).

#### Tarjeta de alimentación

Implementa las fuentes de alimentación de 12V y 5V del esquema de alimentación.

#### Tarjeta de control de alimentación

Implementa la habilitación de las diferentes tensiones de alimentación.

Estas tarjetas junto con el interruptor de emergencia y la batería de 24V forman el diagrama de bloques HW del robot principal. La conexión entre estos elementos se muestra en la figura 5.6. Los 24V de la batería pasan por el interruptor de emergencia antes conectarse con la tarjeta de alimentación de 12V y 5V. Las tensiones de alimentación de 24V, 12V y 5V son conectadas a la tarjeta principal pasando por la tarjeta de habilitación de alimentación. Internamente a la tarjeta principal, la tensión de 3,3V es generada a partir de la tensión de 5V.

La tarjeta de sensores se encuentra conectada a la tarjeta principal y se alimenta mediante las tensiones de 12V y 5V desde la misma. Además ambas se encuentran conectadas mediante una interfaz I2C. Por último, la tarjeta de sistemas de vacío se alimenta directamente de las tensiones de 24V y 12V, pasando igualmente por la tarjeta de habilitación de alimentación. El control de esta tarjeta se realiza mediante el cableado de algunas de las salidas digitales o PWM de la tarjeta principal.

#### 5.3.1 Tarjeta principal

La tarjeta principal *mainboard\_v02* se muestra en la figura 5.7 y su diagrama de bloques es el representado en la figura 5.6. Está compuesta por dos microcontroladores dsPIC33 comunicados por I2C, un maestro y

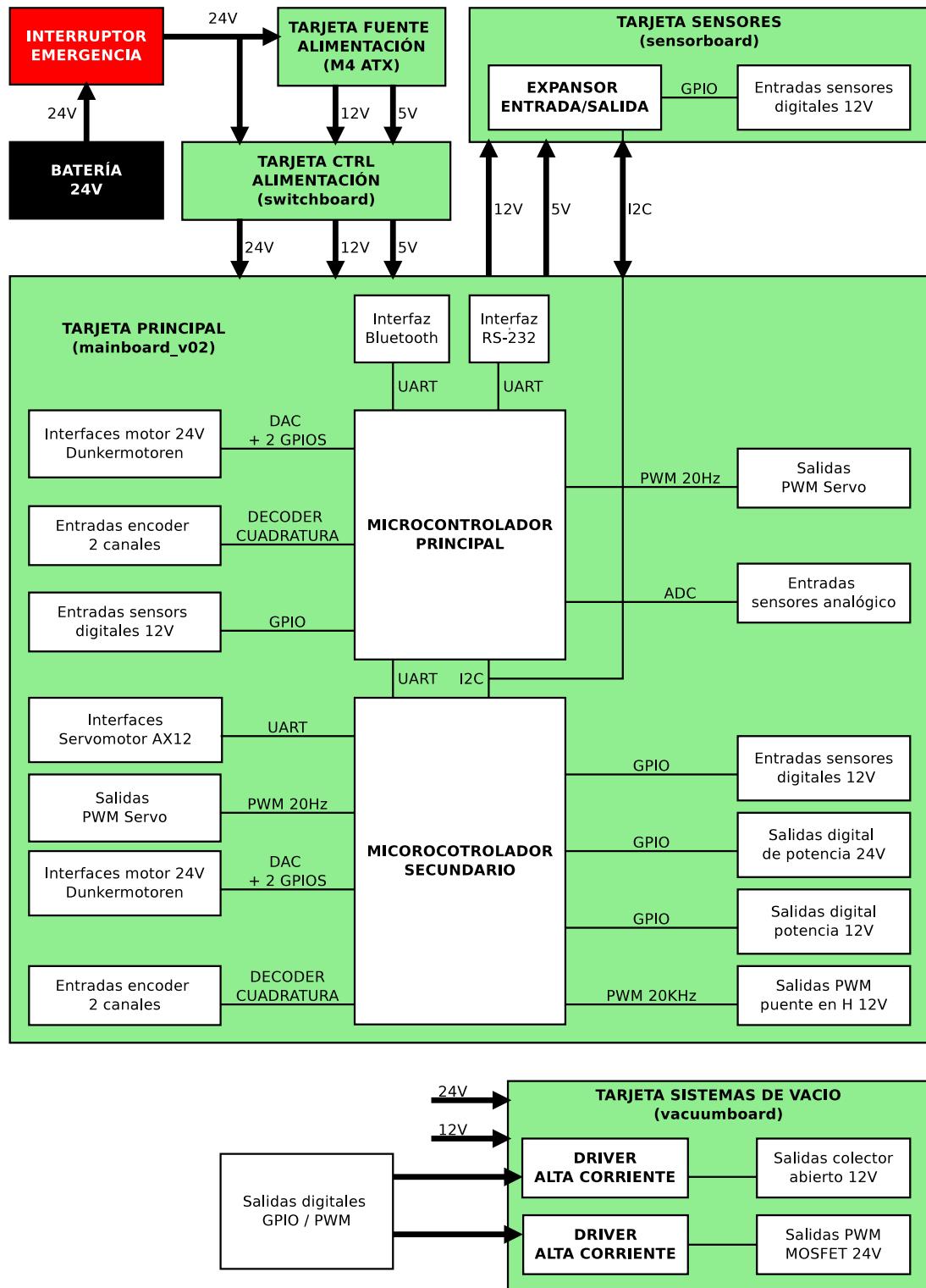


Figura 5.6: Diagrama de bloques del HW del robot principal

un esclavo. El maestro está pensado para controlar la plataforma robótica base y la estrategia del robot, y el esclavo para controlar los sistemas mecánicos. La versión v01 de esta tarjeta tarjeta fue desarrollada para el robot Trompetero (Eurobot 2010). La versión v02 utilizada a partir del robot Zamorano (Eurobot 2011) soluciona varios errores de la primera versión.

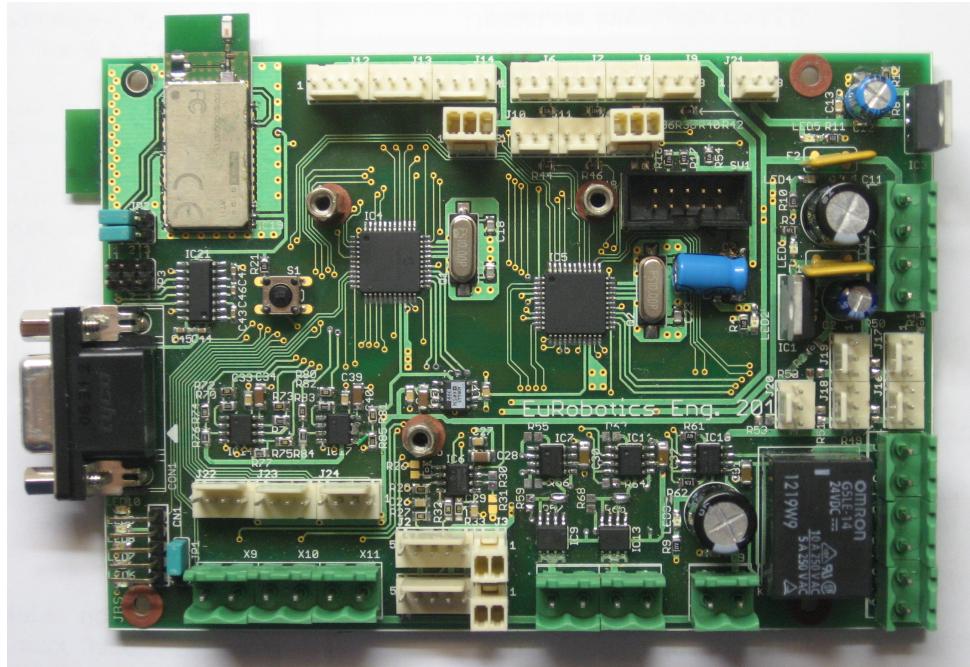


Figura 5.7: Tarjeta *mainboard\_v02*

Las características de esta tarjeta son las siguientes:

- 3 motores Dunkermotoren
- 2 motores DC de 12V
- 1 solenoide de 12V
- 1 relé
- 2 telémetros láseres
- 5 servos estándar
- 1 interfaz servos AX12
- 3 encoders con salida en cuadratura
- 7 sensores 12V
- 1 interfaz RS-232
- 2 interfaces Bluetooth (SPP)
- 1 conector para tarjeta de sensores *sensorboard*.

### 5.3.2 Tarjeta de sensores digitales

La tarjeta de sensores digitales *sensorboard* se conecta directamente sobre la tarjeta principal *mainboard* como una tarjeta hija (ver figura 5.8). Esta tarjeta incluye dos dispositivos expansores de E/S con interfaz I2C y la electrónica de acondicionamiento necesaria para la conexión de sensores digitales alimentados a 12V y de salida en colector o drenador abierto (salida PNP o NPN). Cada dispositivo expander de puertos cuenta con 2 puertos de 8 líneas. En total la tarjeta soporta hasta 32 E/S.

Esta tarjeta fue desarrollada junto con la tarjeta *mainboard* en el año 2010 y ha sido reutilizada desde entonces.



Figura 5.8: Tarjeta *sensorboard*

### 5.3.3 Tarjeta de sistemas de vacío

La tarjeta *vacuumboard* (ver figura 5.9) fue desarrollada para el robot Automático (Eurobot 2014), el cual utilizaba un sistema de vacío para manipular los elementos del juego. Esta tarjeta permite el control de 4 electroválvulas y 4 bombas de vacío. Para ello utiliza drivers de potencia alimentados a 12V y 24V. Esta tarjeta es controlada por la tarjeta principal mediante las salidas digitales y las salidas PWM para servomotores.

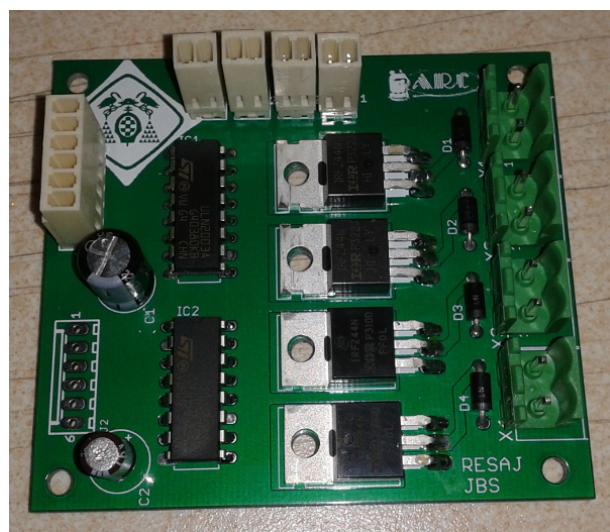


Figura 5.9: Tarjeta *vacuumboard*

### 5.3.4 Tarjeta de control de alimentación

La tarjeta de control de alimentación *switchboard* permite habilitar de forma controlada las diferentes alimentaciones del robot (24V, 12V y 5V). Para ello cuenta con 3 interruptores, uno por cada alimentación, como muestra la figura 5.10. La tarjeta está diseñada para asegurar un correcto encendido del robot. Las tensiones de 12V y 24V dependen de la tensión de 5V, si esta no es habilitada ninguna otra puede ser encendida. De la misma forma, la tensión de 24V depende de la tensión de 12V.

Esta tarjeta fue diseñada en el año 2010 y se ha reutilizado desde entonces.

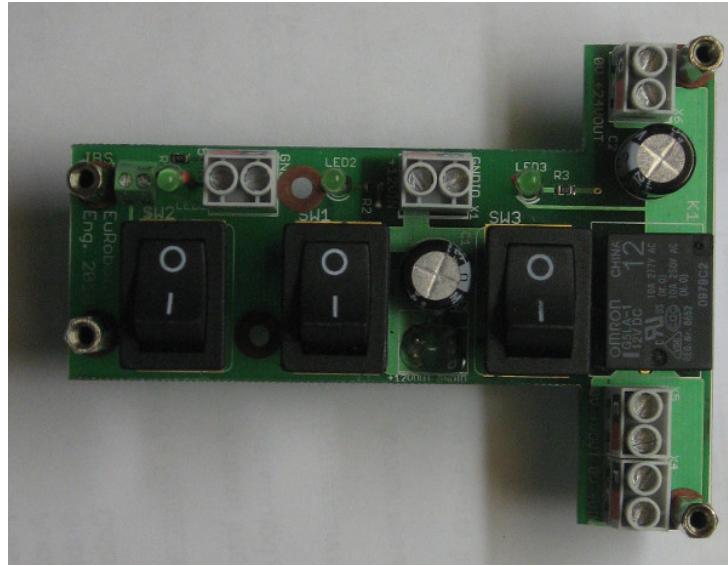


Figura 5.10: Tarjeta *switchboard*

### 5.3.5 Tarjeta de alimentación

La tarjeta de alimentación M4-ATX es una fuente de alimentación comercial mini ATX. Permite un rango de alimentación variable entre 6V y 30V y da salidas de +/-12V, 5V y 3.3V. A partir de ella se obtienen las tensiones de 5V y 12V necesarias para el robot. Su amplio margen de entrada permite además utilizar una batería de 24V o de 12V, según las necesidades del robot.

Por otro lado tiene una potencia total de 250w, de los cuales 75w son de la salida de 5V (15A) y 120w (10A) de la salida de 12V, llegando a picos de corriente de 20A y 16A, respectivamente. La potencia suministrada por esta tarjeta deja un amplio margen al diseño de los sistemas mecánicos. Comparando estas potencias con estimación de consumo del robot P.Tinto (ver tabla 5.5) se puede dar por válido el uso de esta tarjeta, y así se ha comprobado.

Este modelo de tarjeta se ha utilizado desde Eurobot 2010 para alimentar la electrónica tanto del robot principal como del robot secundario.

## 5.4 Implementación HW del robot secundario

La implementación de la arquitectura HW en el robot secundario está compuesta únicamente por dos tarjetas: una tarjeta principal y una tarjeta de alimentación M4-ATX, como la utilizada en el robot principal. Además incluye el interruptor de emergencia y dos interruptores que habilitan la alimentación de 24V y, de 12 y 5V simultáneamente.

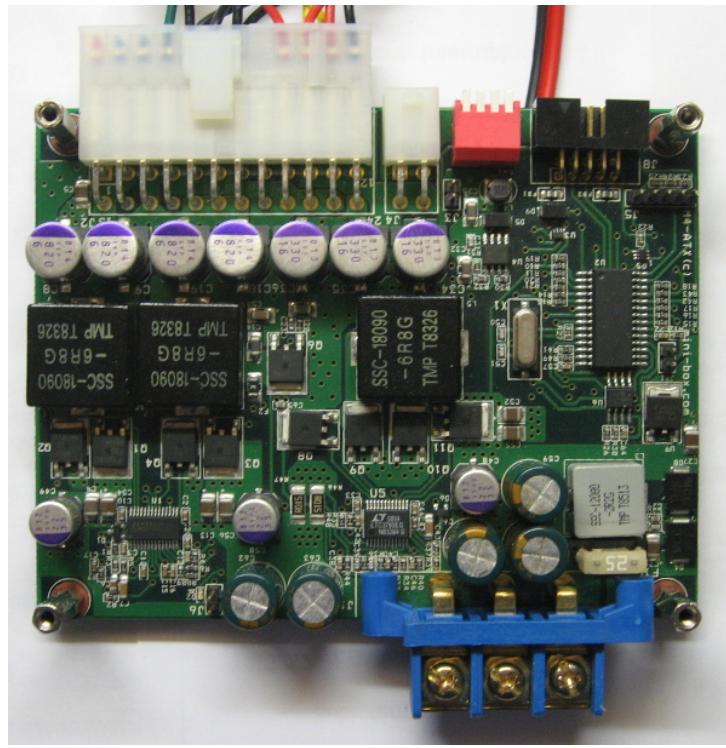


Figura 5.11: Tarjeta *M4-ATX*

El diagrama de conexión entre estos elementos se muestra en la figura 5.12. Al igual que en el robot principal, la batería conecta por medio del interruptor de emergencia con un sistema de habilitación de alimentación y con la fuente de alimentación de 12V y 5V. En este caso la tensión de 24V es habilitada mediante un interruptor y las tensiones de 12V y 5V mediante un segundo interruptor que habilita la fuente de alimentación. Por último, todas estas tensiones alimentan a la tarjeta principal.

Respecto a la tarjeta principal, a partir de año 2012 las reglas de Eurobot permitieron la participación de dos robots por equipo: un robot principal y uno secundario. El robot secundario con un perímetro máximo mucho menor. Dicho año se desarrollaron dos robots: Automático (principal) y Crispín (secundario). Crispín fue implementado a partir de la tarjeta *dspic33\_protoboard*, tarjeta prototipo con la que se validó la arquitectura HW (ver figura 5.2). Al año siguiente se desarrolló la tarjeta *mainboard\_v03* (ver figura 5.13).

La tarjeta *mainboard\_v03* está diseñada para controlar la plataforma robótica, la baliza tipo faro y los sistemas mecánicos del robot secundario, e integra todo lo necesario para ello. El diagrama de bloques de la tarjeta se encuentra representado en la figura 5.12. Un único microcontrolador dsPIC33 es un elemento principal de la tarjeta. Sus recursos se distribuyen entre los elementos de la plataforma robótica y la baliza tipo faro de espejo giratorio. El resto de recursos E/S están destinados al control de los sistemas mecánicos. Aunque éstos suelen ser mucho menos complejos y simples que los del robot principal, la tarjeta además incluye un expansor de E/S digital por I2C de 16 líneas.

El resto de características de E/S de la tarjeta *mainboard\_v03* son las siguientes:

- 3 motores DC de 12V
- 4 servomotores estándar
- 1 interfaz servos AX12
- 2 encoders con salida en cuadratura

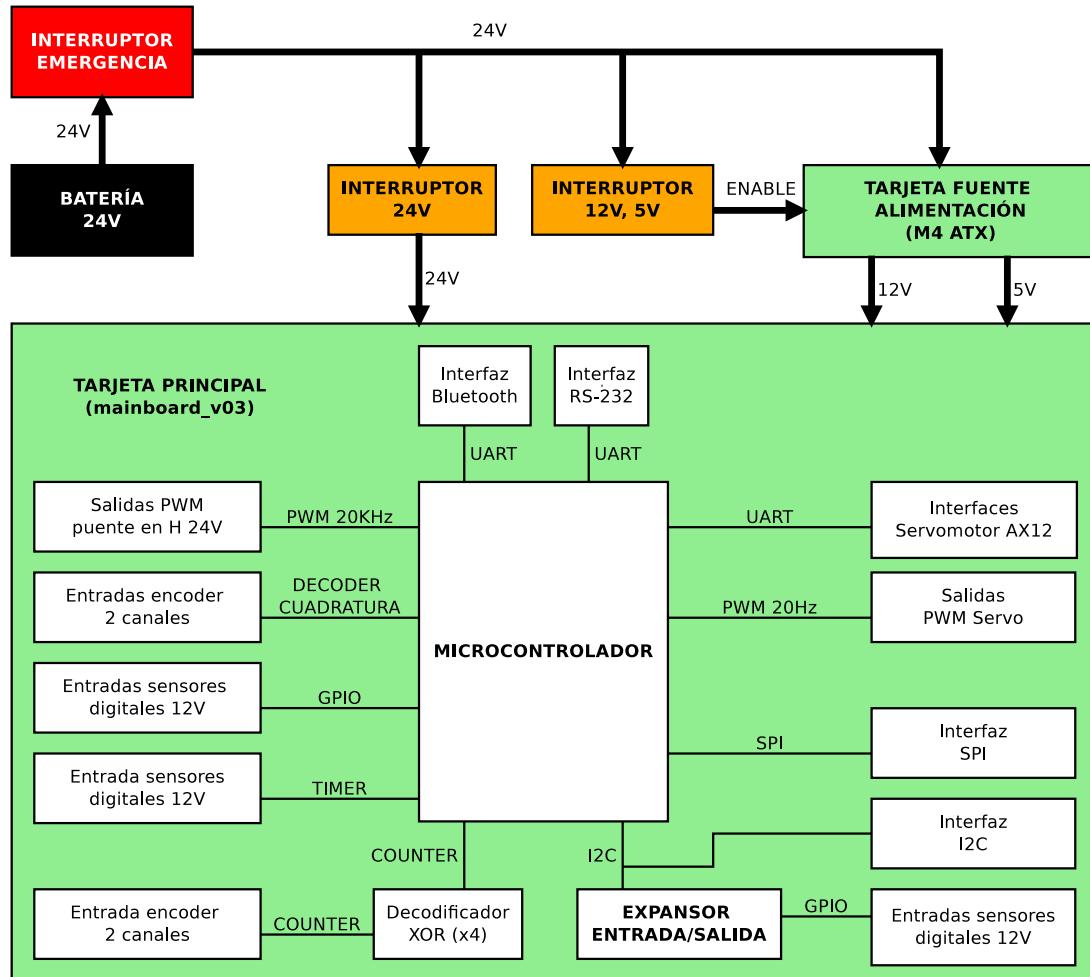


Figura 5.12: Diagrama de bloques del HW del robot secundario

- 1 encoder con salida en cuadratura (módulo x4, sin decodificación de signo).
- 16 E/S de 12V/5V
- 2 entradas de captura de tiempos
- 1 interfaz RS-232/Bluetooth
- 1 interfaz I2C
- 1 interfaz SPI

Respecto a la plataforma robótica, ésta utiliza 2 de las 3 salidas PWM para motores y las entradas en cuadratura decodificadas por el microcontrolador. Además, los sensores de detección de obstáculos se conectan a algunas de las entradas digitales. La comunicación con el robot principal así como la depuración del robot se realiza mediante una de las UART del microcontrolador, que se encuentra cableada a la interfaz Bluetooth y a la interfaz RS-232 mediante el mapeado dinámico de pines del microcontrolador.

Respecto a la implementación de la baliza tipo faro, ésta utiliza las 2 entradas de captura de tiempo, la tercera entrada de encoder y la tercera de las salidas para motor DC. El resto de recursos (salidas PWM de servomotores estándar, la interfaz de servomotores AX12 y la E/S digital) quedan a disposición de los sistemas mecánicos. Por último, la tarjeta cuenta con una interfaz I2C y SPI accesible para posibles ampliaciones de recursos E/S.

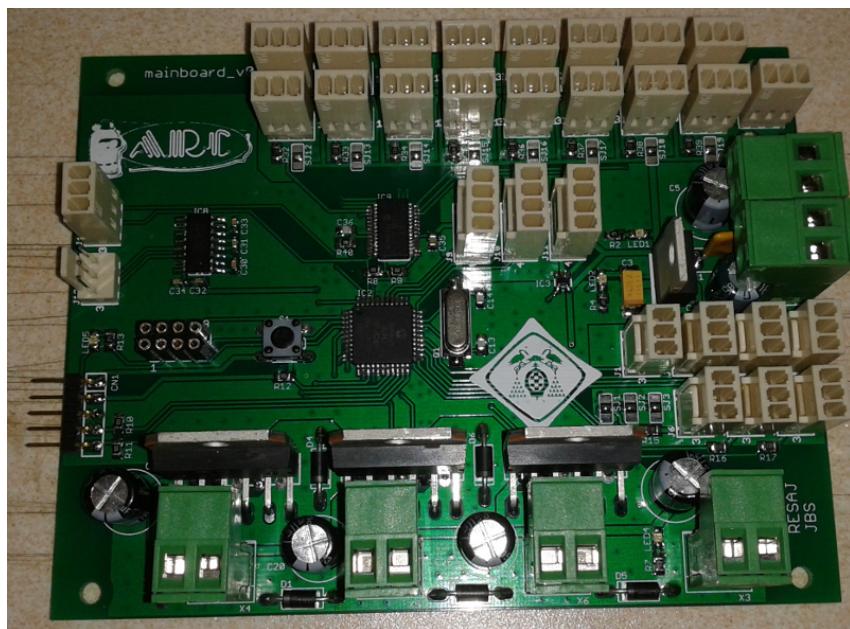


Figura 5.13: Tarjeta *mainboard\_v03*



# Capítulo 6

## Desarrollo software

*Desde hace muchas generaciones un P.Tinto se ha distinguido por tres cualidades que lo hacen inconfundible en cualquier lugar del mundo: un P.Tinto siempre mira hacia arriba con optimismo, a un P.Tinto la elegancia se le reconoce dónde quiera que vaya,...informal sí, pero elegante. Y por sobre todo, un P.Tinto siempre lleva su propia energía, sin olvidar que a un P.Tinto le gusta echarse azúcar en el café hasta que haga isla.*

Padre de P.Tinto, película *El milagro de P.Tinto*

El desarrollo software del robot de Eurobot se organiza en capas o módulos funcionales, de forma que aquellos de nivel superior implementan funciones más abstractas que los inferiores. De mayor a menor nivel de abstracción, son las siguientes:

1. Estrategia de juego
2. Temática de juego
3. Sistemas mecánicos
4. Plataforma robótica base

El módulo *plataforma robótica* incluye las funcionalidades de desplazamiento, localización y evitación de obstáculos. Al mismo nivel, se encuentra el módulo *sistemas mecánicos* encargado de controlar los mecanismos utilizados por el robot para manipular los elementos de juego. Ambos módulos se integran y se sincronizan en la capa de *temática de juego* para implementar funcionalidades propias de la temática de Eurobot. Por último, la capa *estrategia de juego* decide cuándo y qué acciones del juego realizar en cada momento para jugar un partido de Eurobot.

Así por ejemplo, si la capa de estrategia decide recolectar el tomate 1, llama a funciones del tipo `harvest_tomato(TOMATO_1)` de la capa de temática de juego, la cual a su vez utilizará funciones de la plataforma robótica del tipo `goto_avoid_xy(TOMATO_1_X, TOMATO_1_Y)` para moverse evitando los obstáculos hacia la posición del tomate 1 y, funciones de los sistemas mecánicos del tipo `system_tomatoes_set_mode(MODE_HARVEST)` para recolectar el tomate mediante el sistema mecánico de recolección de tomates.

## 6.1 Librerías Aversive y Aversive4dspic

La implementación SW de los robots se ha realizado a partir de las librerías Aversive [4], desarrolladas por el equipo de Eurobot Microb Technology [5]. Las librerías Aversive constituyen un marco de trabajo para el desarrollo de sistemas basados en microcontroladores AVR de Atmel.

Las librerías Aversive4dspic son una migración de las librerías Aversive a microcontroladores dsPIC de Microchip, utilizados por el HW desarrollado para los robots. Para realizar la migración de las librerías Aversive ha sido necesario desarrollar la capa de *drivers de dispositivos HW* dependiente directamente del microcontrolador dsPIC, manteniendo las interfaces de los drivers equivalentes de microcontroladores AVR. Además, ha sido necesario migrar dependencias específicas del compilador utilizado por las plataformas AVR al equivalente en el compilador para dsPICs.

Las librerías Aversive4dspic mantienen la compatibilidad hacia atrás y permiten trabajar también con plataformas AVR. El código nuevo desarrollado o las modificaciones de código existente han sido realizadas siguiendo la misma convención de código que las librerías originales.

Las librerías Aversive utilizan un *toolchain* basado en la herramienta make. Así, a partir de un fichero `makefile`, es posible configurar el proyecto para utilizar los módulos necesarios de la librería, especificar los ficheros de código fuente, compilar el código fuente y programar el microcontrolador. Además de soportar la compilación cruzada para microcontroladores AVR, estas librerías soportan la compilación y ejecución en *HOST* para plataformas GNU/Linux.

Respecto a las librerías Aversive4dspic, soportan la creación de un proyecto y configuración de los módulos a utilizar mediante la herramienta make y la compilación en *HOST*. La compilación cruzada para microcontroladores dsPIC y su programación se realiza mediante el entorno de desarrollo de Microchip MPLab y MPLabX.

Las estructura de las librerías se divide en 5 carpetas principales:

- **config**: ficheros relacionados con la configuración de las librerías.
- **include**: ficheros de cabecera generales, no relacionados con los módulos de la librería.
- **mk**: ficheros `makefiles` de las librerías (proyecto, módulos y plantillas).
- **modules**: módulos de la librería. Un módulo es una pequeña librería con una funcionalidad específica.
- **projects**: ejemplos de proyecto.

Los módulos se encuentran organizados en las siguientes carpetas:

- **modules/base**: módulos comunes y frecuentemente utilizados.
- **modules/comm**: módulos de comunicación (UART, SPI, I2C, ...).
- **modules/crypto**: módulos para operaciones de cifrado de datos.
- **modules/debug**: módulos de ayuda a la depuración.
- **modules/devices**: módulos de dispositivos específicos no dependientes del HW del microcontrolador (LCDs, motores, ...).
- **modules/encoding**: módulos de codificación (base 64, *hamming*, ...).

- **modules/hardware**: módulos de interfaces del microcontrolador (*timers*, ADC, ...).
- **modules/ihm**: módulos de interfaz hombre-máquina (menues, interfaz de comandos, ...)

Algunos módulos incluyen una configuración previa a la compilación. En estos casos, los módulos tienen asociado un fichero de cabecera de configuración con el formato `nombre_modulo_config.h`. Estos ficheros se encuentran en la carpeta `config` de cada módulo. Durante la configuración de un proyecto Aversive, estos ficheros son copiados a la carpeta raíz del proyecto, donde pueden ser modificados para ajustar la configuración a las necesidades del proyecto.

Por otro lado, cada uno de los módulos de la librerías incluye un proyecto de prueba. Este proyecto sirve para probar el correcto funcionamiento del módulo durante su desarrollo y como ejemplo de uso para aprender a utilizarlo. Los proyectos de prueba se encuentran en la carpeta `test` de cada módulo.

De cara a la implementación SW del robot de Eurobot (o cualquier sistema embebido), cada capa funcional utiliza ciertos módulos de la librería de funcionalidad muy específica. Sin embargo, hay algunos módulos que se utilizan transversalmente a todas las capas y que implementan funcionalidades comunes y útiles a todas ellas. Las siguientes secciones describen estas funcionalidades y los módulos utilizados en su implementación.

### 6.1.1 Herramientas matemáticas

La librería incluye varios módulos matemáticos que permiten realizar operaciones matemáticas en punto fijo o con objetos como polígonos y vectores:

- **modules/base/math/fixed\_point**: operaciones matemáticas en punto fijo.
- **modules/base/math/geometry**: operaciones con puntos, líneas, polígonos y círculos.
- **modules/base/math/vect2**: operaciones con vectores.

Estos módulos son utilizados por ejemplo para la implementación de sistemas de control en punto fijo, o en la implementación de evitación de obstáculos del robot, donde cada obstáculo es representado por un polígono.

### 6.1.2 Gestión de *buffers* de datos

Es muy común que para la implementación de interfaces de comunicación (UART, I2C, SPI, ...) se necesiten *buffers* donde almacenar los datos que van a ser transmitidos o que se han recibido. Para ello puede utilizarse el módulo **modules/comm/cirbuf** que permite crear y gestionar buffers de datos circulares.

### 6.1.3 Planificador de eventos

El módulo **modules/base/scheduler** implementa un planificador de eventos con prioridad. Los eventos pueden ser periódicos o de ejecución única. Lo primeros llevan asociado un periodo y los segundos un instante de ejecución a partir del momento de creación.

El módulo *scheduler* está pensado para ejecutarse en la interrupción de un *timer*, el periodo de dicho *timer* constituye el periodo base del *scheduler* (por ejemplo 1ms). En cada periodo se evalúa que evento

ejecutar. Dado que los eventos se ejecutan en un contexto de interrupción, las funciones asociadas a estos han de ser de corta duración y no bloqueantes.

Mediante eventos del *scheduler*, el robot implementa por ejemplo: la lectura de sensores cada 10ms, los sistemas de control cada 5ms o los protocolos de comunicación con la baliza o el robot secundario cada 10ms.

#### 6.1.4 Medida de tiempo

El módulo **modules/base/time** permite medir el tiempo del sistema. Este módulo se ejecuta como un evento del *scheduler* y su periodo de ejecución se corresponde con la resolución de medida del tiempo. Permite obtener el tiempo transcurrido desde su inicialización en horas, minutos, segundos y microsegundos.

Este módulo permite implementar esperas bloqueantes o no bloqueantes. Por ejemplo, a los 90 segundos los robots han de dejar de jugar y desactivar todos los sistemas. Esta espera se puede implementar de forma precisa a partir de este módulo evaluando periódicamente el número de segundos transcurridos desde el inicio de partido.

#### 6.1.5 Mensajes de depuración

El módulo **modules/debug/error** implementa mensajes de depuración o *logs* que permiten trazar el funcionamiento del software implementado con el fin de verificar su correcta ejecución. Cada mensaje tiene asociado un número de error y un nivel de gravedad: EMERG, ERROR, WARNING, NOTICE y DEBUG.

El módulo permite definir la función a partir de la cual los mensajes son generados para cada nivel de gravedad. En dicha función se define el formato de los mensajes, la información de estos y la interfaz por la cual son transmitidos (UART, SPI, I2C, etc.).

El software desarrollado para los robots y los módulos de las librerías utilizan el módulo *error*. En este caso, la función que genera los mensajes utiliza la salida estándar (conectada a una UART) para transmitir los mensajes. Esta función, además, filtra los mensajes por número de error y nivel de gravedad, de forma que sólo aquellos mensajes con números de error habilitados y de nivel de gravedad superior al establecido serán transmitidos, con la excepción de los mensajes de nivel ERROR que serán siempre transmitidos.

#### 6.1.6 Interfaz de línea comandos

Las librerías permiten implementar una interfaz de línea comandos similar a la de sistemas GNU/Linux con las siguientes características:

- Interfaz de comunicación serie configurable (UART, I2C, SPI, ...).
- Caracteres imprimibles y no imprimibles VT100.
- Ayuda/información de cada comando.
- Completado automático de argumentos y sugerencia del tipo de argumento a introducir.
- Tipo de argumentos: texto (*strings*), números enteros (8 y 16 bits) y números decimales (*float*).

- Comprobación de número argumentos y tipo de los argumentos.
- Histórico de comandos ejecutados.

La interfaz de línea de comandos se implementa a partir de los siguientes módulos:

- **modules/ihm/vt100**: módulo que permite interpretar los caracteres no imprimibles VT100.
- **modules/ihm/rdline**: módulo de lectura de una línea, terminada en retorno de carro, a través de una interfaz serie de entrada, por ejemplo una UART.
- **modules/ihm/parse**: módulo que realiza la correlación de una línea con los posibles comandos a ejecutar y sus argumentos, y en caso de coincidencia, ejecuta el comando correspondiente.

Los módulos `rdline` y `parse` utilizan el módulo `vt100` para poder interpretar los caracteres no imprimibles como el tabulador o los cursores. Por ejemplo, el tabulador es utilizado para completar un argumento tipo texto o para indicar el tipo del argumento siguiente. Por otro lado, los cursores permiten moverse por el listado de comandos ejecutados o por una línea escrita para modificarla.

El módulo `parse` tiene asociado una lista de comandos y cada comando tiene asociado unos parámetros de entrada y una función a ejecutar. A medida que `rdline` recibe una línea utiliza información de la lista de comandos de `parse` para implementar el completado de argumentos de texto o indicar el tipo del siguiente argumento. Si el comando y sus argumentos coinciden con alguno de los comandos de la lista de comandos, el módulo `parse` ejecuta la función asociada al comando, propagando los argumentos introducidos por la línea de comandos a dicha función.

En el caso del desarrollo software del robot de Eurobot, la interfaz de línea de comandos constituye la capa de mayor nivel, más abstracta y la interfaz entre el robot y sus desarrolladores. La línea de comandos está presente durante todo el desarrollo del robot e implementa comandos mediante los cuales controlar o probar el funcionamiento de las diferentes capas o niveles del software. Así, hay comandos que permiten controlar directamente el HW del robot, como la generación de señales PWM o la lectura del valor de sensores, o comandos muchos más abstractos que permiten mover el robot hasta una posición (x,y) del campo o iniciar un partido de Eurobot.

## 6.2 Arquitectura software

La arquitectura SW desarrollada se definió en el año 2010 durante el desarrollo del robot Trompetero. Por aquél entonces sólo se competía con un robot por equipo, más tarde, en el año 2014, la arquitectura evolucionó hacia la competición con dos robots por equipo.

La figura 6.1 muestra el diagrama de bloques SW de un equipo de dos robots: un robot principal y uno secundario. Ambos robots implementan la misma arquitectura SW diferenciándose únicamente en el número de elementos implicados en la implementación. Debido a la complejidad de los sistemas mecánicos del robot principal, éste implementa la arquitectura mediante dos microcontroladores, mientras que el robot secundario utiliza un único microcontrolador. Además, la baliza tipo faro del robot principal constituye un sistema independiente, mientras que la baliza del robot secundario se encuentra integrada en la arquitectura HW y SW del robot.

Cada robot tiene una interfaz serie que permite acceder a la línea de comandos de cada uno, utilizando un terminal serie de un ordenador. En el caso del robot principal, utiliza una interfaz serie RS-232, mientras que el robot secundario utiliza una interfaz serie Bluetooth (perfil SPP, *Serial Port Profile*).

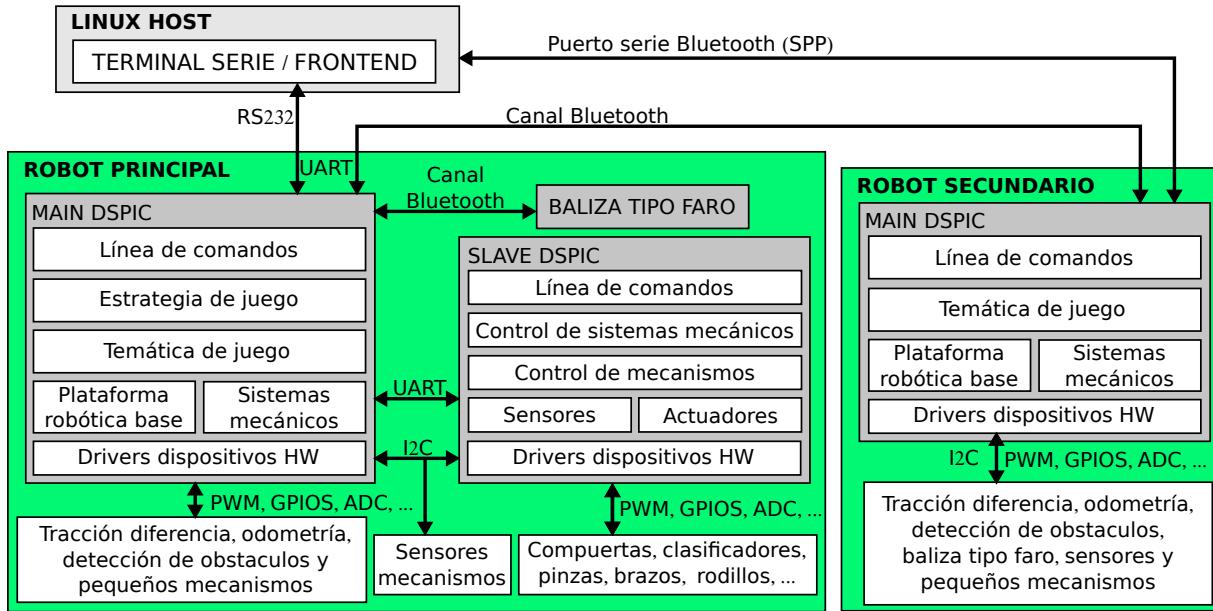


Figura 6.1: Diagrama de bloques software del robot principal y secundario

Ambos robots y la baliza del robot principal se encuentran conectados mediante sendos canales serie Bluetooth. Estos canales son utilizados para implementar el protocolo de comunicación entre el robot principal y la baliza, y entre el robot principal y el robot secundario. También es posible utilizar estos canales Bluetooth para acceder a la línea de comandos de la baliza y del robot secundario desde el terminal serie del ordenador, a partir de un *bypass* de los datos en el robot principal.

Igualmente, ambos robots utilizan una interfaz I2C maestro-esclavo para acceder a dispositivos de expansión de E/S, a los que se encuentran conectados sensores. Además, en el caso del robot principal, esta interfaz es utilizada por el microcontrolador principal para comunicarse con el microcontrolador esclavo (SLAVE DSPIC), el cual implementa el control de sistemas mecánicos. Mediante la interfaz I2C se implementa un protocolo de mensajes que permite compartir la información de los dispositivos conectados al bus (sensores) con ambos microcontroladores y, una gestión de los sistemas mecánicos por parte del microcontrolador principal.

Por otro lado, como se puede observar en la figura 6.1, la arquitectura SW de cada robot se organiza en las siguientes capas y módulos funcionales:

### Drivers de dispositivos HW

Esta capa permite abstraer los módulos o dispositivos HW del robot, como módulos de comunicación del microcontrolador (UART, I2C, SPI, ...) o dispositivos externos a éste como servomotores, o drivers en puente en H para el control de motores. Éste nivel trabaja, por ejemplo, con funciones del tipo `uart_send(UART0, dato)` para transmitir un dato por la UART numero 0, o `pwm_mc_set(OUT1, -1000)` para fijar el valor y sentido del motor conectado al la salida OUT1 a -1000.

### Plataforma robótica base

Módulo que implementa las funcionalidades de la plataforma robótica base: control de posición, odometría, gestión de trayectorias y evitación de obstáculos. Implementa funciones del tipo `robot_goto_xy(1000, 500)` para ir al punto (x=1000, y=500), o `robot_get_xy()` para obtener la coordenada (x,y) del robot.

### Sistemas mecánicos

Módulo que abstrae el control de los diferentes sistemas mecánicos del robot mediante los cuales se manipulan los elementos del juego. Éste módulo trabaja con funciones del estilo `system_tomatoes_set_mode(HARVEST)` (configura el sistema de tomates en modo recolección), o como `system_tomatoes_get_num()` (obtener el número de tomates recolectados).

### Temática de juego

Esta capa permite abstraer completamente las funcionalidades de la plataforma robótica y de los sistemas mecánicos e integra la información de la baliza. De esta forma, a este nivel se trabaja con funciones que sincronizan los movimientos del robot con los sistemas mecánicos, como `harvest_tomato(TOMATO_1)` (ir hasta la posición del tomate 1 y recolectarlo).

Esta capa también tiene funciones que integran la información de varios módulos, como la información de odometría, del tiempo transcurrido, de la baliza y de los sensores de obstáculos. Es el caso de la función `wait_traj_end(END_TRAJ | END_TIMER | END_OBSTACLE)`, que espera el fin de una trayectoria debido a diferentes razones: *punto destino alcanzado, el tiempo de partido se acaba, o por un obstáculo en el camino encontrado.*

### Estrategia de juego

Esta capa implementa la estrategia de juego de un partido, así como las diferentes fases de las que se compone un partido. Las funciones esperadas a este nivel son del tipo `is_the_opponent_in_zone(ZONE_TOMATOES_1)` (¿está el oponente en la zona de los tomates?), o `goto_zone(ZONE_TOMATOES_1)` (ir a la zona de los tomates), o `work_on_zone(ZONE_TOMATOES_1)` (trabajar en la zona de los tomates).

Como se puede observar en la 6.1, la capa *estrategia de juego* no se encuentra implementada en el robot secundario. Esto es debido a que la estrategia se encuentra centralizada en el robot principal y, es éste, el que decide en todo momento lo que le corresponde hacer al robot secundario.

### Línea de comandos

La línea de comandos permite ejecutar comandos de cualquier módulo o nivel. De esta forma durante el desarrollo de las diferentes capas o módulos, las funcionalidades de éstas pueden ser probadas por partes y verificar su correcto funcionamiento. Por ejemplo, a nivel estrategia, el comando `init yellow`, inicializa los robots para jugar en el lado amarillo del campo de juego. Y mediante el comando `start qualification match`, se ejecuta un partido con la estrategia diseñada para la fase de clasificación de una competición de Eurobot.

Respecto al módulo *sistemas mecánicos*, a su vez se encuentra dividido en varias capas o módulos, los cuales se encuentran representados en el microcontrolador esclavo del robot principal, el cual, implementa únicamente el control de los sistemas mecánicos (ver figura 6.1). Aclarar, que en el caso del robot secundario estas capas se encuentran integradas en un único microcontrolador.

## 6.3 Implementación funcional

A continuación se describe la implementación funcional de los módulos y capas principales de la arquitectura SW del robot de Eurobot. El orden seguido es el mismo en el que se suelen desarrollar, de menor a mayor nivel de abstracción.

### 6.3.1 Plataforma robótica base

Como se ha visto en la sección de arquitectura SW, la plataforma robótica base se sitúa entre la capa *drivers de dispositivos HW* y la capa *temática de juego*. Este módulo implementa los siguientes bloques funcionalidades del robot:

- Gestión de trayectorias
- Control de posición
- Posicionamiento por odometría
- Detección de bloqueos

La figura 6.2 representa el diagrama de bloques funcional de cada uno de estos bloques y su relación con otros módulos o capas de la arquitectura SW. Su implementación se ha realizado utilizando las librerías Aversive4dspic. Principalmente a partir de módulos de implementación de sistemas de control (`modules/devices/control_system`) y módulos específicos de implementación de plataformas robóticas (`modules/devices/robot`). Todos ellos se ejecutan periódicamente como eventos del módulo `scheduler`.

#### 6.3.1.1 Gestión de trayectorias

La gestión de trayectorias está compuesta por los módulos siguientes:

`modules/devices/robot/trajectory_manager`

Módulo de gestión de trayectorias que calcula la trayectoria para alcanzar un punto destino, a partir de las coordenadas de dicho punto y de la posición actual del robot. El estado de la ejecución de la trayectoria puede consultarse en todo momento. Dicho estado informa de si la trayectoria está en proceso o si ha finalizado satisfactoriamente.

Para alcanzar las coordenadas de destino, pueden ser necesarios varios movimientos del robot (lineales o de orientación). Estos movimientos se corresponden a consignas de posición de distancia y ángulo. Las consignas generadas por el módulo de gestión de trayectorias son procesadas por el control de posición.

`modules/devices/robot/obstacle_avoidance`

Módulo de evitación de obstáculos que permite, a partir de un punto de destino y una lista de obstáculos, representados por polígonos, obtener el camino más corto para llegar a dicho punto. Como resultado, el módulo devuelve la lista de puntos que representa el camino a seguir.

Estos dos módulo permiten, por ejemplo, implementar a nivel de temática de juego una función que permita al robot ir hasta un punto determinado del campo de juego, independientemente de si hay obstáculos en medio. El nivel de temática de juego conoce las dimensiones del campo de juego, la posición de sus elementos, y tiene información de la posición de los oponentes (obtenida de la baliza). Así por ejemplo, la función `goto_avoid(x,y)` de la figura 6.2, utiliza el módulo de evitación de obstáculos para obtener los puntos del camino a recorrer para alcanzar una coordenada (x,y), y mediante el módulo de gestión de trayectorias, hace que el robot vaya pasando por cada uno de ellos hasta llegar al destino.

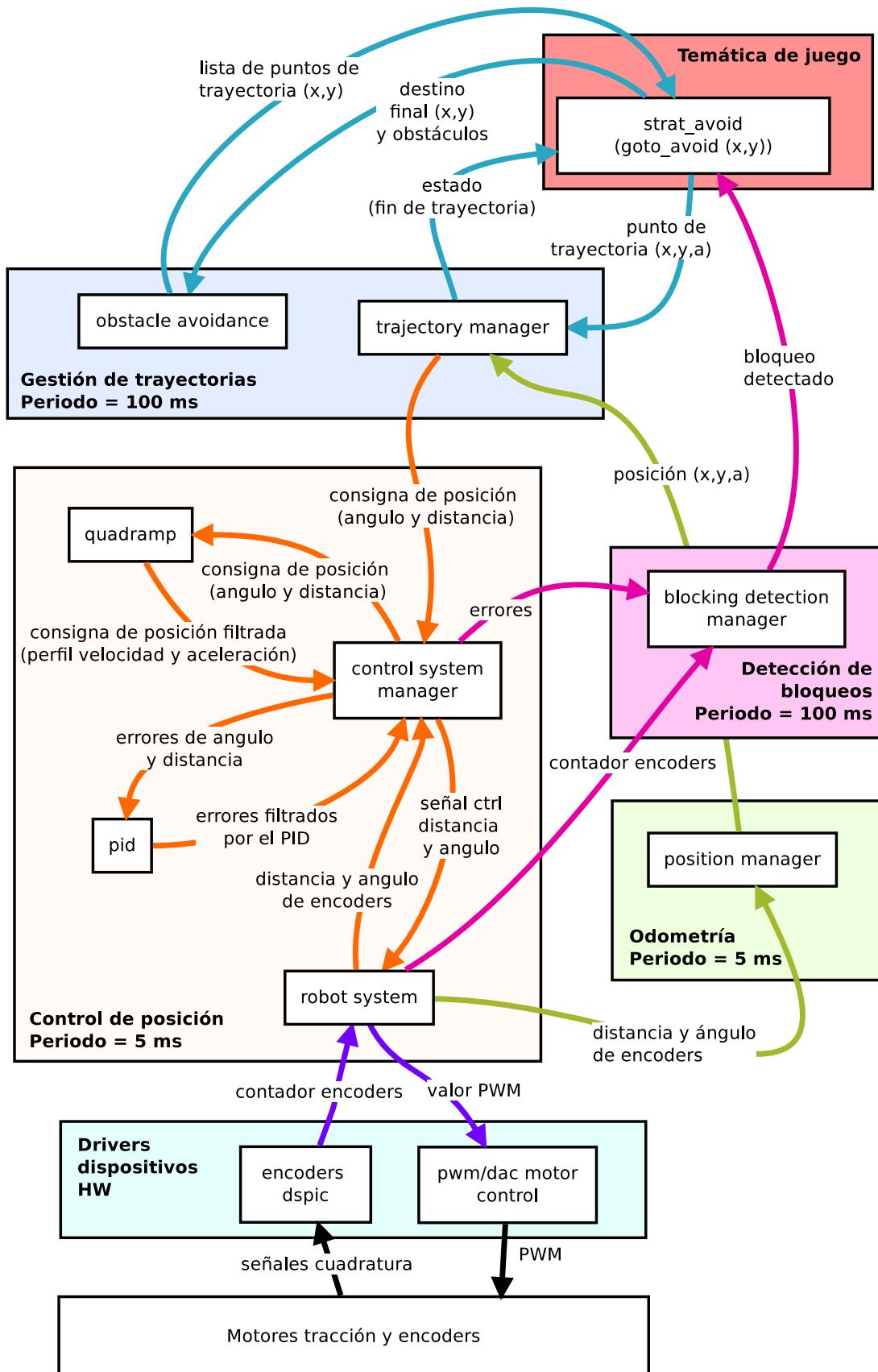


Figura 6.2: Diagrama de organización y funcionamiento SW de la plataforma robótica base

### 6.3.1.2 Control de posición

El control de posición se implementa a partir de dos módulos principalmente:

`modules/devices/robot/robot_system`

Módulo que hace de interfaz entre los drivers de dispositivos HW, y el control de posición y odometría. Por un lado, convierte los pulsos de los encoders, a medidas de posición y ángulo del robot. Y por otro, convierte las señales de control de distancia y ángulo, a valores de PWM para cada motor.

`modules/devices/control_system/control_system_manager`

Módulo a partir del cual se implementan los controladores de posición de distancia y ángulo del robot. El módulo gestiona y hace de interfaz entre los diferentes bloques que forma un sistemas de control clásico.

El control de posición polar del robot se implementa de la siguiente forma. Por un lado, el módulo de gestión de sistemas de control recoge las consignas de posición dadas por el gestor de trayectorias. Éstas consignas de posición son filtradas por un módulo `quadramp` (`modules/devices/control_system/filters`) para implementar un perfil de velocidad trapezoidal. A partir de la posición en distancia y ángulo del robot, dada por el módulo `robot_system`, y las consignas filtradas, se calculan los errores de posición en ángulo y distancia. Dichos errores, son procesados por un filtro `pid` que calcula la señal de control de cada controlador. Por último, estas señales de control, son traducidas por el módulo `robot_system` a valores de señal PWM y aplicadas a los motores por medio del driver de motores.

### 6.3.1.3 Odometría

La odometría se implementa a partir del módulo `position_manager`, pertenece al grupo de módulos de implementación de robots (`modules/devices/robot/`). Este módulo recibe la medida de posición en distancia y ángulo del módulo `robot_system` y calcula la posición del robot a partir de algoritmos de odometría. La posición del robot es utilizada por el módulo de gestión de trayectorias, pero además, podría ser utilizada por capas SW de mayor nivel.

### 6.3.1.4 Detección de bloqueos

La detección de bloqueos se implementa a partir del módulo `blocking_detection_manager`, al igual que el módulo de odometría, también pertenece al grupo de módulos de implementación de robots (`modules/devices/robot/`). Este módulo estima la corriente de los motores de tracción a partir de la medida de velocidad de los encoders y de la señal de control de distancia y ángulo de los motores. En caso de que la corriente estimada supere un umbral máximo durante un tiempo determinado, la detección es notificada, de forma que la trayectoria en curso se pueda abortar.

Observar que la función `goto_avoid(x, y)`, descrita anteriormente, además de comprobar el estado de la trayectoria en ejecución, puede comprobar si durante la misma se detecta un bloqueo mecánico, e interrumpir la ejecución si lo considera oportuno.

### 6.3.2 Sistemas mecánicos

Los sistemas mecánicos se encuentran al mismo nivel que la plataforma robótica base, por lo tanto se sitúan igualmente entre la capa *drivers de dispositivos HW* y la capa *temática de juego*.

El módulo de sistemas mecánicos tiene como objetivo abstraer su control y gestión a capas superiores, como la temática de juego o la estrategia. La implementación SW se divide en varias capas y módulos:

#### Sensores

Módulo encargado de la adquisición y procesamiento de los sensores de los sistemas mecánicos (finales de carrera, medidores de distancia, detectores de color, etc.). Este módulo implementa funciones del tipo `sensor_get (SENSOR1)` para, por ejemplo, leer el estado del sensor número 1.

#### Actuadores

Módulo de control de los actuadores de los mecanismos (servomotores, motores, electroválvulas, etc). Este módulo implementa funciones del tipo `ax12_set_pos (AX12_ID_1, 512)` (mover el servomotor AX12 a la posición 512).

#### Control de mecanismos

Esta capa abstrae e integra los actuadores y sensores que forman mecanismos funcionales. Por ejemplo, un mecanismo puede ser un brazo robótico formado por varios servomotores y una pinza, o un clasificador de tomates compuesto por un palo movido por un servomotor. De esta forma se utilizan funciones del tipo `arm_goto_xy()` para llevar al brazo a una coordenada, o funciones como `tomatoes_clasifier_set_pos (SIDE_LEFT)` para clasificar un tomate hacia el lado izquierdo.

#### Control de sistemas mecánicos

Esta capa gestiona los diferentes mecanismos de los que se compone un sistema mecánico, e implementa los diferentes modos de funcionamiento de éste. Por ejemplo, un sistema mecánico puede ser el sistema de recolección de tomates. Éste sistema puede estar compuesto por un mecanismo de clasificación de tomates, un sensor detector de presencia de tomates, un mecanismo de almacenamiento y un mecanismo de liberación de los tomates recolectados. El sistema, por ejemplo, puede trabajar en modo *recolectar* o *liberar tomates* y, permite *clasificar* los tomates rojos y verdes. Dependiendo del modo de funcionamiento se han de utilizar unos mecanismos u otros.

La forma más común de implementar esta capa es mediante máquinas de estado. Por ejemplo, la máquina de estados del ejemplo anterior puede corresponderse con la función `do_system_tomatoes(uint8_t mode, uint8_t tomato_type)`, la cual permite hacer funcionar el sistema de recolección de tomates en sus diferentes modos de funcionamiento (parámetro `(mode)`), y elegir el tipo de tomate (parámetro `tomato_type`).

Por ejemplo, en el juego *Robomovies* de Eurobot 2015, los robots tenían que realizar las siguientes acciones:

- Construir focos de cine (*spotlights*) a partir de las piezas (*stands* y *bulblight*) repartidas por todo el campo de juego.
- Cerrar unas claquetas de cine (*claperboards*)
- Llenar vasos con palomitas de maíz (*popcorns*)

- Poner unas alfombras rojas es una escalinata.
- Subir la escalinata.

Entre todas estas acciones el robot P.Tinto se encarga de las tres primeras. Para cada una de ellas, P.Tinto tiene un sistema mecánico dedicado. De forma que a cada sistema mecánico le corresponde una implementación HW y SW.

La figura 6.3 representa la organización y funcionamiento de la implementación SW de los sistemas mecánicos de P.Tinto:

- Sistema de focos de cine (*stands\_system*)
- Sistema de palomitas (*popcorns\_system*)
- Sistema de claquetas (*claperboards\_system*)

Los módulos correspondientes a estos sistemas forman el nivel de *control de sistemas mecánicos*. Por encima, el nivel de *temática de juego* gestiona los modos de funcionamiento de éstos y obtiene información de su estado.

En la figura 6.3 se representa concretamente el funcionamiento del sistema de focos de cine (*stands\_system*). Este sistema maneja las piezas (*stands*) con las que se construyen los focos de cine. Permite apilar los *stands* en dos torres de hasta 4 elementos, y posteriormente construir un foco de cine de hasta 8 elementos mediante un mecanismo que transporta los *stands* de una torre a otra.

El sistema de *stands* se compone de varios mecanismos. Cada uno de ellos tiene asociado un módulo SW controlador:

- 2 elevadores de *stands* (*stand\_elevator*)
- 2 pinzas de sujeción de torres de *stands* ()
- 2 clasificadores de *stands* (*stand\_blades*)
- 1 intercambiador de *stands* (*stand\_exchanger*)
- 2 pinzas de *stands* (*stands\_claps*)

Los controladores de mecanismos hacen de interfaz entre la funcionalidad del mecanismo y los actuadores con los que está construidos. Un ejemplo sencillo es la pinza que coge los *stands*, esta tiene dos consignas posibles: abierta o cerrada; y a cada una le corresponde una consigna de posición de un servomotor. Un ejemplo más complejo sería el de un brazo robótico, formado por varios servomotores, cuya consigna sea una posición del espacio (x,y,z). En este caso, para alcanzar dicha consigna es necesario realizar el control sincronizado de todos los servomotores.

Por debajo del nivel de *control de mecanismos* se encuentra la *lectura y procesamiento de sensores* y el *control de actuadores*. Los sistemas mecánicos utilizan sensores para sincronizar los movimientos de los mecanismos. En el caso de P.Tinto, un sensor a la entrada del sistema de focos de cine permite detectar cuando un *stand* se encuentra en la entrada y puede ser recolectado.

En el caso de la figura 6.3 los mecanismos están formados por los siguientes tipos de actuadores, y sus módulos controladores SW correspondientes:

- Motores (cs)

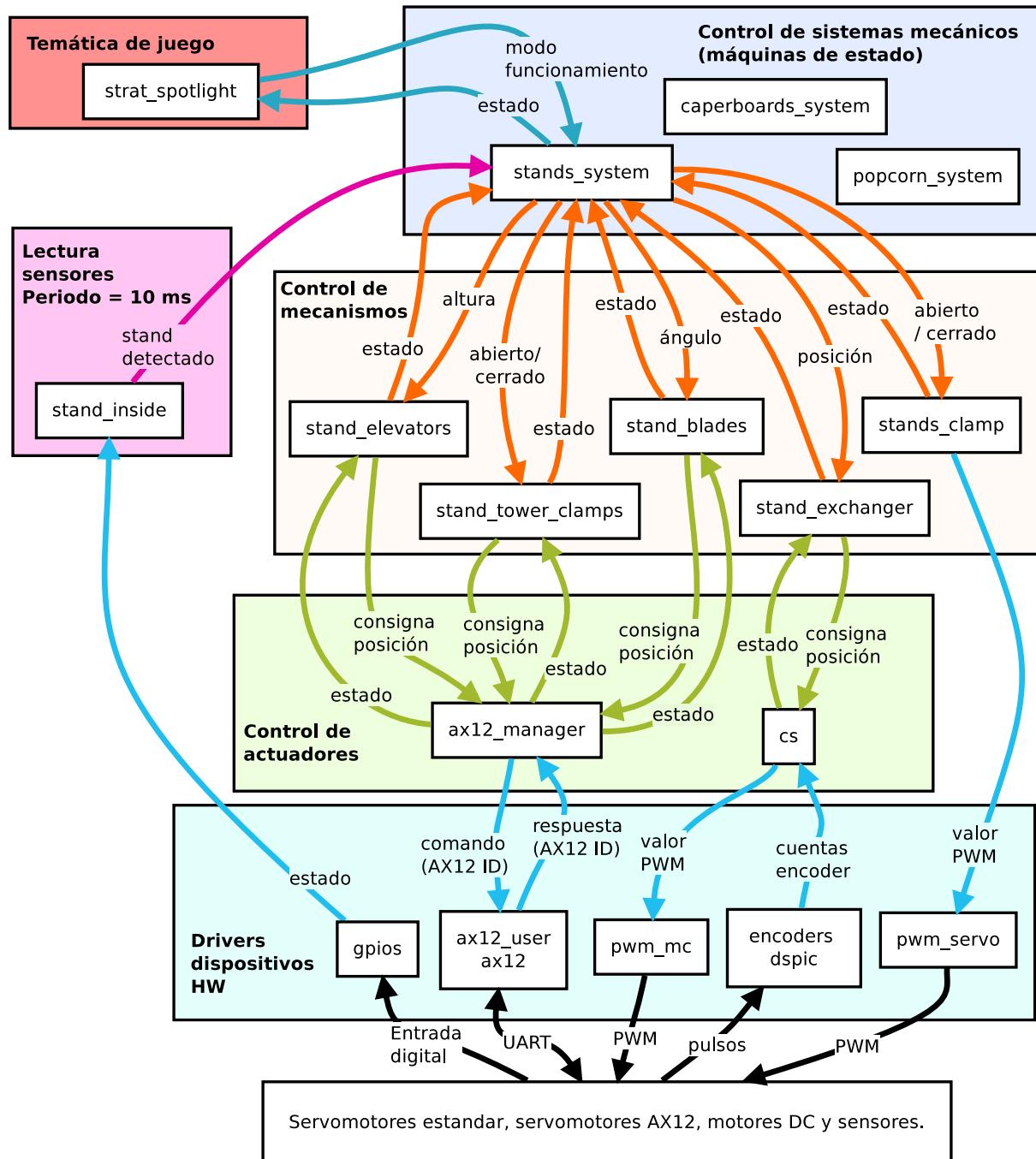


Figura 6.3: Diagrama de organización y funcionamiento SW de los sistemas mecánicos

- Servomotores AX12 (`ax12_manager`)
- Servomotores estándar

El módulo `cs` implementa un sistema de control utilizando las librerías Aversive4dspic, de igual forma que la plataforma robótica implementa el control de posición del robot. Tanto el controlador de motores como el de servomotores AX12 permiten conocer la posición real del actuador. Esta información es utilizada para sincronizar las operaciones sin necesitar de sensores adicionales. Sin embargo, en el caso de los servomotores estándar no es posible obtener directamente esta información, y la sincronización ha de realizarse por tiempo o mediante sensores adicionales. Notar que en el caso del mecanismo `stands_clamps` no existe controlador de actuadores, y utiliza directamente el driver generador de PWM `pwm_servo`.

### 6.3.3 Temática de juego

El nivel de *temática de juego* integra toda la funcionalidad de los niveles inferiores y sistemas externos, como la baliza y el robot secundario, para implementar las funcionalidades del robot relativas a la prueba de Eurobot y a la navegación del robot por el campo de juego. En el supuesto de que el robot de Eurobot fuese un producto comercial (un juguete) el desarrollo del mismo llegaría hasta este nivel. A partir de aquí, el usuario del robot podría programar el nivel de estrategia para competir con otros robots en un partido de Eurobot.

Por otro lado, el nivel de *estrategia* básicamente utiliza los recursos que le brinda la *temática de juego* para así realizar las acciones del juego en un orden determinado.

La figura 6.4 representa la funcionalidad y organización de estos dos niveles de implementación SW, así como la relación entre los módulos que los componen.

La implementación de la **temática de juego** se divide principalmente en dos tipos de módulos:

- Módulos de navegación del robot
- Módulos de acciones del juego
- Módulos de ayuda a la estrategia

Los módulos de navegación son los siguientes:

#### **strat\_base**

Módulo principal de navegación. Permite iniciar una trayectoria hacia un punto dado y conocer el fin de la misma. Para ejecutar la trayectoria utiliza el módulo de gestión de trayectorias de la plataforma robótica. El final de una trayectoria se determina a partir de la información de varios módulos:

- Punto destino alcanzado: obtenido por el módulo de gestión de trayectorias.
- Punto destino cercano: el robot se encuentra cerca del punto destino (a una distancia máxima). Esta información es dada también por el módulo de gestión de trayectorias.
- Obstáculo en la trayectoria: por detección de los sensores de obstáculos de corto alcance.
- Oponente en la trayectoria: por la información de la baliza.
- Tiempo de partido: por la cuenta de tiempo de partido (módulo Aversive4dspic `time`).

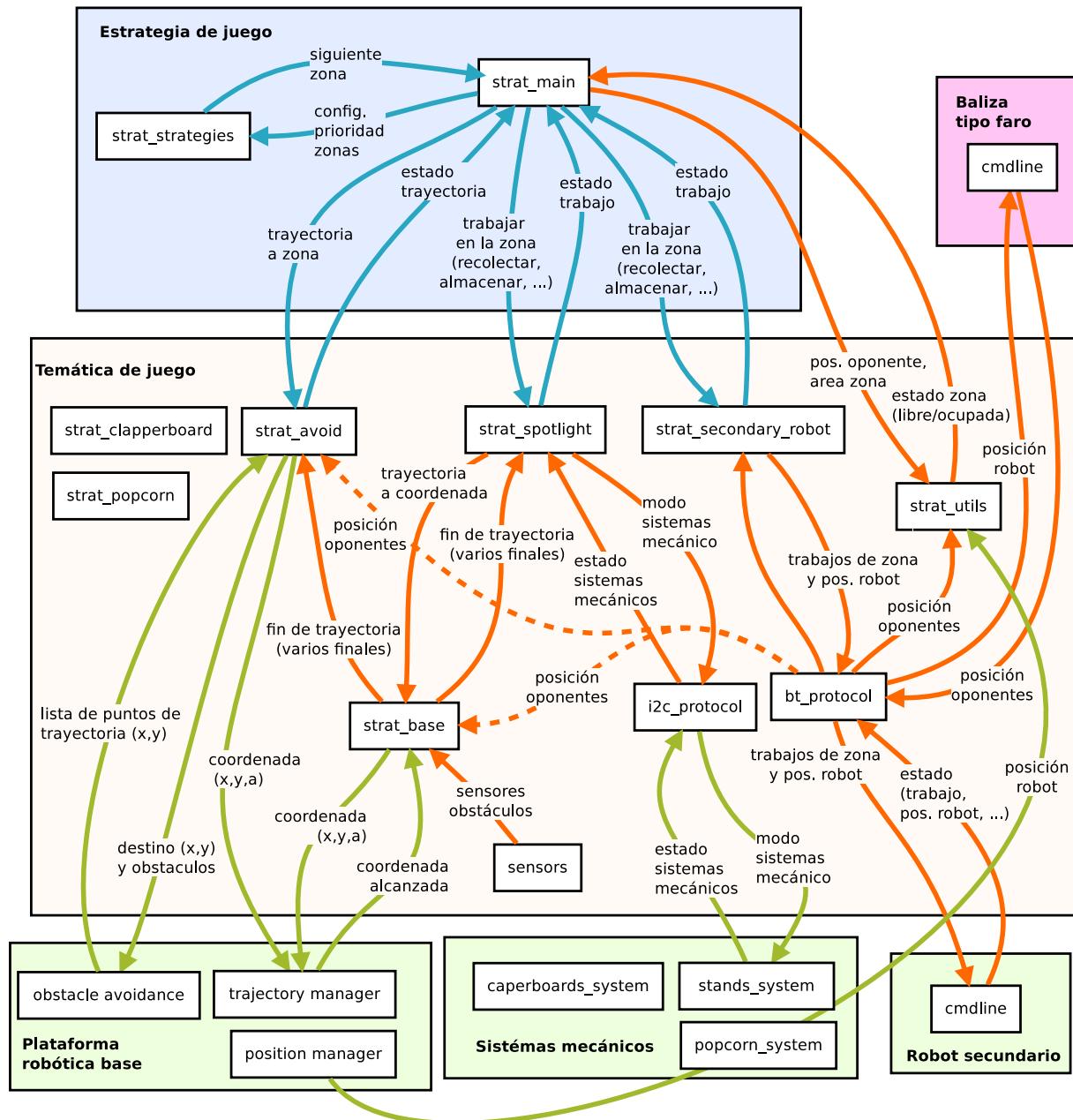


Figura 6.4: Diagrama de organización y funcionamiento SW de los niveles de estrategia y temática de juego

Además, este módulo implementa la gestión de la velocidad durante las trayectoria dependiendo de la posición y distancia a los oponentes. Por ejemplo, al acercarse o pasar por el lado de un oponente, la velocidad se decrementa para prevenir una posible colisión o maniobra de evitación.

#### **strat\_avoid**

Módulo que implementa la navegación con obstáculos. Para ello utiliza el módulo de evitación de obstáculos de la plataforma robótica. A partir de la posición de los elementos del campo de juego y de los oponentes, obtiene la lista de puntos que define el camino a seguir para alcanzar el destino. Cada tramo del camino es ejecutado como una trayectoria por la plataforma robótica.

El fin de cada tramo se obtiene del modulo **strat\_base**. La utilización de la condición de *punto cercano* para los tramos intermedios permite realizar caminos más suaves, ya que el robot no llega a pararse, enlazando los puntos como una única trayectoria.

Respecto a los módulos relativos a acciones del juego, estos tienen una fuerte dependencia con el mismo. Continuando con el ejemplo de los sistemas mecánicos, la figura 6.4 muestra los correspondientes al robot P.Tinto:

- **strat\_clapperboard**: acciones de claquetas de cine.
- **strat\_popcorn**: acciones de vasos de palomitas.
- **strat\_spotlight**: acciones de focos de cine.
- **strat\_secondary\_robot**: acciones delegadas en el robot secundario. Incluyen acciones de claquetas, vasos de palomitas y de las alfombras rojas.

Los módulos funcionan siguiendo un mismo patrón que implementa la sincronización de movimientos del robot con la manipulación de elementos de juego. La figura 6.4 muestra el caso del módulo **strat\_spotlight**. Por un lado, las acciones a realizar están pensadas para ser ejecutadas en las cercanías de los elementos de juego donde los desplazamientos son pequeños. Por ello, se realizan trayectorias sin evitación de obstáculos utilizando el módulo. Por otro lado, la manipulación de los elementos se realiza a partir de los modos de funcionamiento del sistema mecánico correspondiente. La sincronización se consigue a partir del estado de la trayectoria y de los mecanismos.

La forma de llegar hasta el control de los sistemas mecánicos depende de dónde estén implementados. En el caso del módulo **strat\_spotlight** se realiza por medio del protocolo I2C, ya que los sistemas mecánicos se encuentran implementados en el microcontrolador esclavo. Igualmente, las acciones delegadas en el robot secundario, son gestionadas por medio del enlace Bluetooth y del protocolo de comunicaciones asociado.

Por último, el módulo **strat\_utils** implementa funciones de ayuda a la estrategia a partir de la información de posición del robot, obtenida de la odometría de la plataforma base, e información de la posición de los oponentes, obtenida de la baliza.

### 6.3.4 Estrategia de juego

Los robots desarrollados han implementado alguna de estas estrategias o combinación de ellas:

#### **Estrategia secuencial bloqueante**

Estrategia en la que el robot sigue una secuencia de movimientos y acciones. En caso de encontrar un obstáculo espera hasta que éste desaparezca para continuar.

### Estrategia reactiva

Estrategia en la que el robot reacciona ante los obstáculos, evitándolos o realizando acciones alternativas. Además, decide qué acciones realizar en función de la posición actual del oponente y zonas que el oponente ha visitado.

### Estrategia basada en prioridades

Estrategia que decide las acciones a realizar basándose en la prioridad de las mismas. En caso de no poder realizar la acción más prioritaria, salta a la siguiente en prioridad.

### Estrategia adaptativa

Estrategia que en función de la evolución del partido, a partir de las acciones realizadas por el oponente y el tiempo, cambia dinámicamente las prioridades de las acciones.

### Estrategia sincronizada

Estrategia en la que las acciones se realizan de forma sincronizada entre dos robots.

La implementación de las estrategias se ha realizado utilizando un patrón basado en zonas de trabajo y prioridades. En este patrón el campo se divide en zonas de trabajo. Cada una de ellas tiene asociado varios elementos:

- Tipo de elemento de juego: en el caso del robot P.Tinto, claquetas, *stands* o palomitas.
- Área de trabajo: área necesaria para trabajar en la zona.
- Posición de inicio: posición inicial desde la que se trabaja en la zona.
- Posición del elemento: posición del elemento de juego a manipular.
- Prioridad: prioridad de la zona de trabajo.
- *Flags* de control: señales de control para la gestión de estrategia. Por ejemplo, para marcar si la zona ya ha sido trabajada o si ha sido visitada por el oponente.
- Robot: robot que puede trabajar en la zona, que tiene la funcionalidad.

Una vez definidas todas las zonas, la estrategia interpreta la información de cada zona para decidir a qué zona ir a trabajar. El algoritmo a seguir consta de 3 fases que se repiten continuamente:

1. Obtener la zona de mayor prioridad y válida
2. Ir a la zona
3. Trabajar en la zona

Este patrón se encuentra representado en el bloque de estrategia de juego de la figura 6.4. La estrategia al comienzo de partido, o de forma dinámica durante el partido, configura las prioridades de cada zona mediante el módulo `strat_strategies`.

La obtención de la siguiente zona a la que ir a trabajar, se realiza a partir de la información de este módulo y en base a los criterios de zona válida establecidos. Por ejemplo, un criterio fundamental es que el oponente no se encuentre trabajando en la zona. Esta comprobación se realiza mediante el módulo `strat_utils` de la temática de juego.

El desplazamiento a la zona elegida se realiza utilizando el modulo `strat_avoid`, que permite evitar los posibles obstáculos en el camino hacia el punto de inicio de la zona. Si la zona no pudiese ser alcanzada se volvería a obtener una nueva zona.

Una vez alcanzada la zona, en función del tipo de elemento de juego, se utilizaría el módulo de la temática correspondiente. Por ejemplo, si la zona es tipo *stand* se llamaría a la función de recolección de *stands* del módulo `strat_spotlight`.

## 6.4 Simulador de robots y campo de juego

La utilización de un simulador vino motivada porque en el año 2014 dos de los integrantes del equipo, encargados de desarrollar la estrategia, estaban viviendo en Alemania. El desarrollo del simulador permitió trabajar conjuntamente con el desarrollo del resto del robot en España.

El poder simular el comportamiento del robot y los elementos del campo de juego, como los robots oponentes, en una plataforma PC tiene grandes ventajas:

- Desarrollo de la estrategia independiente y en paralelo con el desarrollo de la mecánica y de la electrónica.
- Necesita pocos recursos: un ordenador. Con uno portátil se puede desarrollar software en cualquier lugar.
- Reduce el tiempo de puesta en marcha de la estratégica y permite simular todo tipo de situaciones.
- Permite probar cambios de estrategia de último momento sin necesidad de realizar pruebas en el campo de juego.

Las desventajas son pocas, pero a tener en cuenta:

- La implementación de los elementos simulados: dispositivos HW, sistemas mecánicos, elementos externos, como la baliza, o interfaces de comunicación; han de tener un comportamiento idéntico al real. En caso contrario las pruebas realizadas en simulación no se corresponderán con pruebas reales en el campo de juego.
- Tendencia a confiar demasiado en la simulación. Aunque la simulación dé como resultado el comportamiento esperado, es necesario validarlos en pruebas reales, siempre que sea posible.

El simulador desarrollado permite simular el funcionamiento del robot y el campo de juego. Está basado en el simulador desarrollado por el equipo Microb Technology [5] y utiliza la capacidad de las librerías Aversive4dspic para ser compiladas para un *host GNU/Linux*.

La figura 6.5 muestra el diagrama de bloques del simulador de dos robots. Si lo comparamos con el diagrama de bloques de la figura 6.1 se observa que los bloques dependientes del HW y la mecánica de cada robot han desaparecido. En su lugar se encuentra el simulador de cada robot, los cuales se encuentran conectados con el simulador de campo de juego.

La implementación del simulador consta de tres partes:

- Simulación de los módulos de las librerías Aversive4dspic.
- Simulación del HW y mecánica del robot.

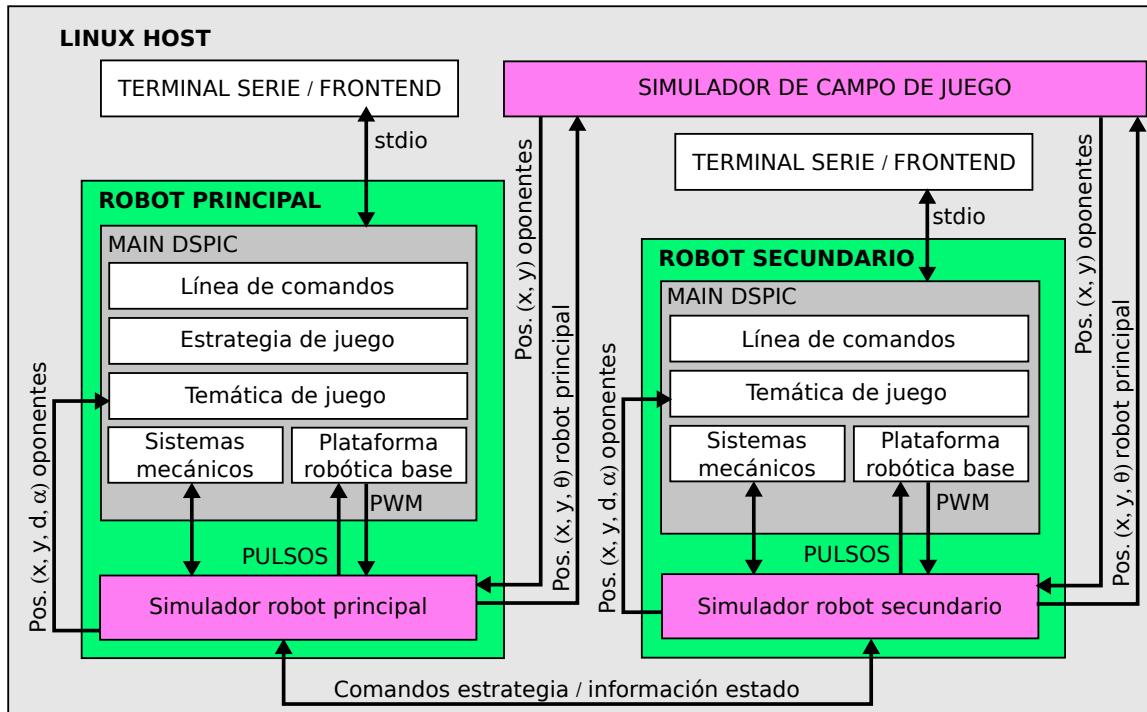


Figura 6.5: Diagrama de bloques software del simulador de robots y campo de juego

- Visualizador de campo de juego y robots.

La simulación de algunas funcionalidades o módulos de las librerías Aversive4dspicse se realiza mediante el módulo `hostsim` (de `modules/base`). Este módulo simula la salida estándar del módulo de comunicaciones `uart`, la medida de tiempo y la ejecución periódica del gestor de eventos, mediante una implementación mediante recursos propios del sistema GNU/Linux.

Respecto a la simulación del robot esta se realiza mediante el módulo `robotsim` que implementa la simulación de los elementos mecánicos y dependientes del HW. Este módulo es compilado con el proyecto del robot. El ejecutable obtenido da como resultado un robot virtual que corre sobre GNU/Linux.

El módulo `robotsim` implementa la simulación de los siguientes elementos:

### Plataforma robótica

El simulador a partir del valor de control de los motores emula el comportamiento de los motores de tracción y devuelve el valor de pulsos de encoders equivalentes a la velocidad del robot.

### Sistemas mecánicos

La emulación de sistemas mecánicos se realiza mediante un retardo que equivale al tiempo que los mecanismos tardan en realizar una función.

### Comunicación entre robots

En la arquitectura real del sistema de dos robots, ambos están comunicados por medio de un canal serie Bluetooth conectado a un módulo UART. En el simulador la UART es emulada mediante la escritura y lectura de ficheros. Cada simulador de robot tiene un fichero de escritura y otro de lectura que emulan el canal Bluetooth.

### Baliza

Simulación del comportamiento y medidas reales de la baliza del robot en función a su rango de medida y precisión.

### Colisión con obstáculos

Simulación de bloqueos mecánicos y de sensores de obstáculos. Por ejemplo, simulación de la colisión con las paredes del campo de juego, de forma que el robot detecte un obstáculo por bloqueo en los límites del campo de juego.

De esta forma, al compilar el proyecto de ambos robots se generan sendos programas, que al ser ejecutados en un terminal, dan como resultado dos robots virtuales comunicados entre sí y controlados mediante la línea de comandos de cada uno (ver figura 6.6).

Respecto al tercer elemento en juego, el simulador de campo de juego está realizado en Python (`display.py`) y permite la visualización de los robots en un campo de juego virtual y la simulación de robots oponentes. La visualización se implementa mediante la librería `visual` de Python que utiliza el API de renderización OpenGL. La simulación de los robots oponentes se consigue mediante eventos de un dispositivo tipo ratón. El desarrollador puede simular la posición y movimiento de los robots oponentes interactuando con el ratón sobre el campo de juego virtual.

Como representa la figura 6.6, el simulador de campo de juego se encuentra conectado con cada simulador de robot por un fichero de lectura y uno de escritura. De esta forma cada simulador de robot envía su posición al visualizador de campo de juego y, éste a su vez, envía la posición de los oponentes a los simuladores de los robots para que las procesen como medidas de la baliza.

La ejecución de los programas de cada robot y del visualizador del campo de juego conforman el entorno de simulación completo que se muestra en la figura 6.6.

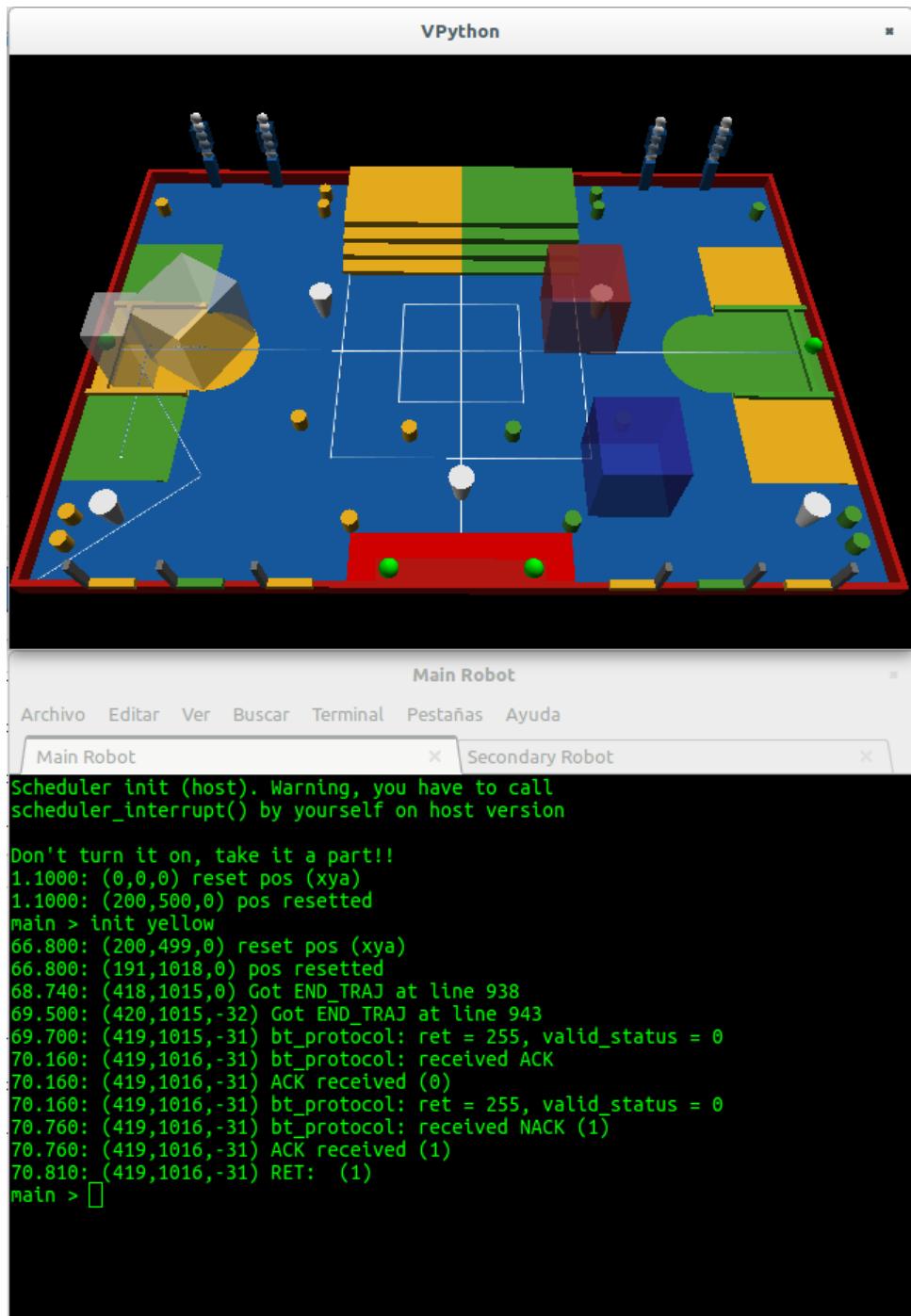


Figura 6.6: Entorno de simulación de los robots P.Tinto y Tirantes, y del campo de juego de la prueba *Robomovies* (Eurobot 2015)



# Bibliografía

- [1] “Pagina oficial de Eurobot,” <http://www.eurobot.org> [Último acceso 28/noviembre/2015].
- [2] “Información sobre la Copa de Francia de la robotica,” [https://fr.wikipedia.org/wiki/Coupe\\_de\\_France\\_de\\_robotique](https://fr.wikipedia.org/wiki/Coupe_de_France_de_robotique) [Último acceso 25/abril/2016].
- [3] “Pagina web del equipo Eurobotics Engineering,” [Último acceso 6/mayo/2016].
- [4] “Código fuente de las librerías Aversive,” <http://git.droids-corp.org/?p=aversive.git;a=summary> [Último acceso 9/junio/2016].
- [5] “Pagina web del equipo Microb Technology ,” [http://wiki.droids-corp.org/articles/m/i/c/Microb\\_Technology.html](http://wiki.droids-corp.org/articles/m/i/c/Microb_Technology.html) [Último acceso 9/junio/2016].
- [6] “Foros de Eurobot,” <http://www.planete-sciences.org/forums/viewforum.php?f=55&sid=c04f70114013ab7853f2f3738c57c505> [Último acceso 26/abril/2016].
- [7] “Foros de Eurobot Open,” <http://www.planete-sciences.org/forums/viewforum.php?f=2> [Último acceso 26/abril/2016].
- [8] “Foro de la Copa de Francia de la robótica,” <http://www.planete-sciences.org/forums/viewforum.php?f=1> [Último acceso 26/abril/2016].
- [9] “Pagina de la asociación *Planet Sciencies*,” <http://www.planete-sciences.org/> [Último acceso 25/abril/2016].
- [10] “Pagina web del Dpto. de electrónica de la Universidad de Alcalá,” <http://depeca.uah.es> [Último acceso 6/mayo/2016].
- [11] “Página web de la Universidad de Alcalá,” <http://www.uah.es> [Último acceso 6/mayo/2016].
- [12] “Pagina web de la empresa Ayudas hidráulicas,” <http://ayudashidraulicas.com> [Último acceso 6/mayo/2016].
- [13] “Pagina web de la tienda Ro-botica,” <http://ro-botica.com> [Último acceso 6/mayo/2016].
- [14] “Pagina web del Consejo de estudiantes de la Universidad de Alcalá,” <http://consejoestudiantes.uah.es> [Último acceso 6/mayo/2016].
- [15] “Pagina web de la empresa Mobile Minds,” <http://www.cologne-intelligence.de/ci-mobile-minds/> [Último acceso 6/mayo/2016].
- [16] “Educación STEM,” [https://es.wikipedia.org/wiki/Educaci%C3%B3n\\_STEM](https://es.wikipedia.org/wiki/Educaci%C3%B3n_STEM) [Último acceso 26/abril/2016].

- [17] “Putting the power of digital making into the hands of people all over the world,” Blog de Philip Colligan, <https://philipcolligan.com/2016/02/20/strategy-1-0/> [Último acceso 26/abril/2016].
- [18] “Página web del Proyecto Robocompetences,” <http://asimov.depeca.uah.es/robocompetences/#Top> [Último acceso 26/abril/2016].
- [19] E. White, Ed., *Making Embedded Systems. Design Patterns for Great Software.* O'Reilly Media, 2011.
- [20] “Pagina web de la plataforma de desarrollo Arduino,” <https://www.arduino.cc/> [Último acceso 6/junio/2016].
- [21] “Código fuente de las librerías Arduino,” <https://github.com/arduino/Arduino> [Último acceso 6/junio/2016].
- [22] “Pagina web del equipo RCVA,” <http://www.rcva.fr/> [Último acceso 9/junio/2016].
- [23] “Pagina web del equipo RCA Aachen,” <http://www.roboterclub.rwth-aachen.de/> [Último acceso 9/junio/2016].
- [24] “Pagina web del equipo Turak,” <https://www.turag.de/> [Último acceso 9/junio/2016].
- [25] O. MATZ, “Faire des balises laser en buvant des bières,” Microb Technology, Tech. Rep., 2010, <http://perso.crans.org/~ghaessig/Microb%20Balises.pdf> [Último acceso 11/julio/2016].
- [26] “Video del robot de Eurobt 2010 del equipo Microb vs. RCVA. Trayectorias continuas.” <https://www.youtube.com/watch?v=PXHjuGzP9fo> [Último acceso 1/junio/2016].
- [27] H. E. J. Borenstein and L.Feng, *Where am I?. Sensors and Methods for Mobile Robot Positioning.* The University of Michigan, 1996, ch. 5. Odometry and Other Dead-Reckoning Methods.
- [28] J. Coulon, “Réflexions sur un robot eurobot,” RCVA, Tech. Rep., 2007, [http://www.rcva.fr/images/stories/site/cours/10ansdexperience/devoir\\_de\\_vacances.pdf](http://www.rcva.fr/images/stories/site/cours/10ansdexperience/devoir_de_vacances.pdf) Traducción Diego Salazar: [https://github.com/eurobotics/documentation/blob/master/curso\\_rcva\\_10xp/curso\\_rcva\\_eurobot.pdf](https://github.com/eurobotics/documentation/blob/master/curso_rcva_10xp/curso_rcva_eurobot.pdf) [Último acceso 16/julio/2016].
- [29] J. J.-O. y. D. B.-A. José A.Bercerra-Vargas, Francisco E.Moreno-García, “Estimación de parámetros y modelo de caja negra de un motor cd sin escobillas,” *Tecno Lógicas*, vol. 17, no. 33, pp. 55–64, Julio-Diciembre 2014.
- [30] J. Coulon, “Asservissement du robot à une trajectoire,” RCVA, Tech. Rep., 2007, <http://www.rcva.fr/images/stories/site/cours/asservissement/asservissement.pdf> [Último acceso 16/julio/2016].
- [31] K. Ogata, Ed., *Ingeniería de control moderna.* PEARSON EDUCACIÓN, 2010.
- [32] J. Coulon, “Odometrie,” RCVA, Tech. Rep., 2010, [http://www.rcva.fr/images/stories/site/cours/Odometrie2010/ODOMETRIE\\_2010.pdf](http://www.rcva.fr/images/stories/site/cours/Odometrie2010/ODOMETRIE_2010.pdf) [Último acceso 11/julio/2016].
- [33] H. E. J. Borenstein and L.Feng, *Measurement and Correction of Systematic Odometry Errors in Mobile Robots.* The University of Michigan, 1996, vol. 12, no. 6.
- [34] “dspic33fj32mc302/304, dspic33fj64mcx02/x04, and dspic33fj128mcx02/x04 data sheet,” Micochip, Tech. Rep., 2009.

## Apéndice A

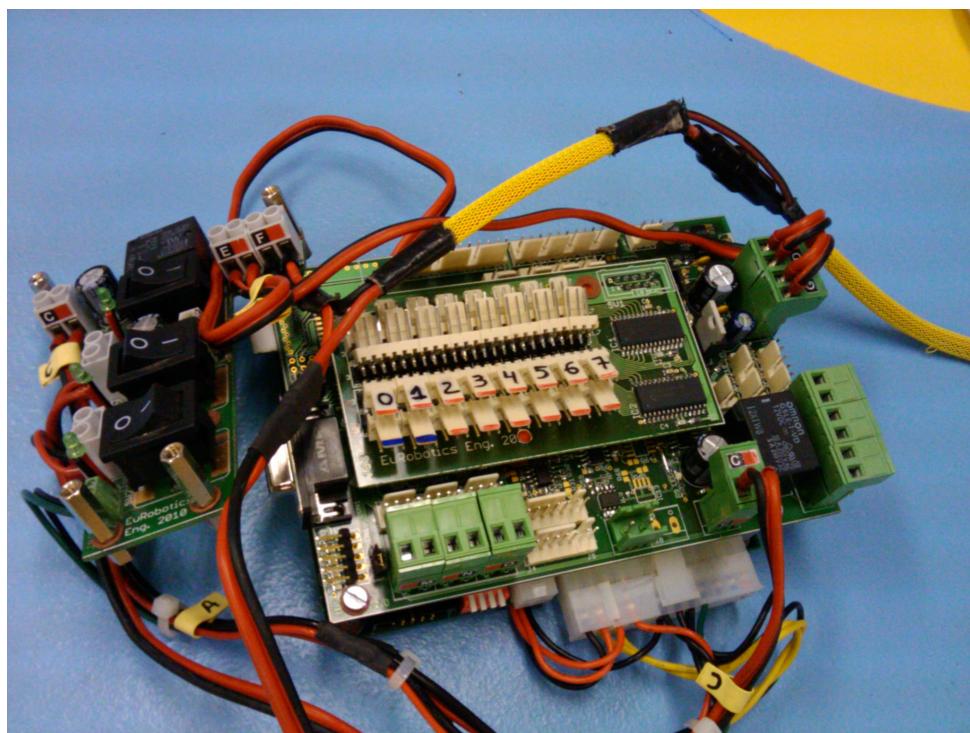
# Planos eléctricos

Los planos eléctricos y documentación de fabricación de la tarjetas electrónicas desarrolladas se encuentra en el repositorio hardware del equipo Eurobotics Engineering en GitHub.

Este repositorio es accesible desde la dirección web: <https://github.com/eurobotics>.

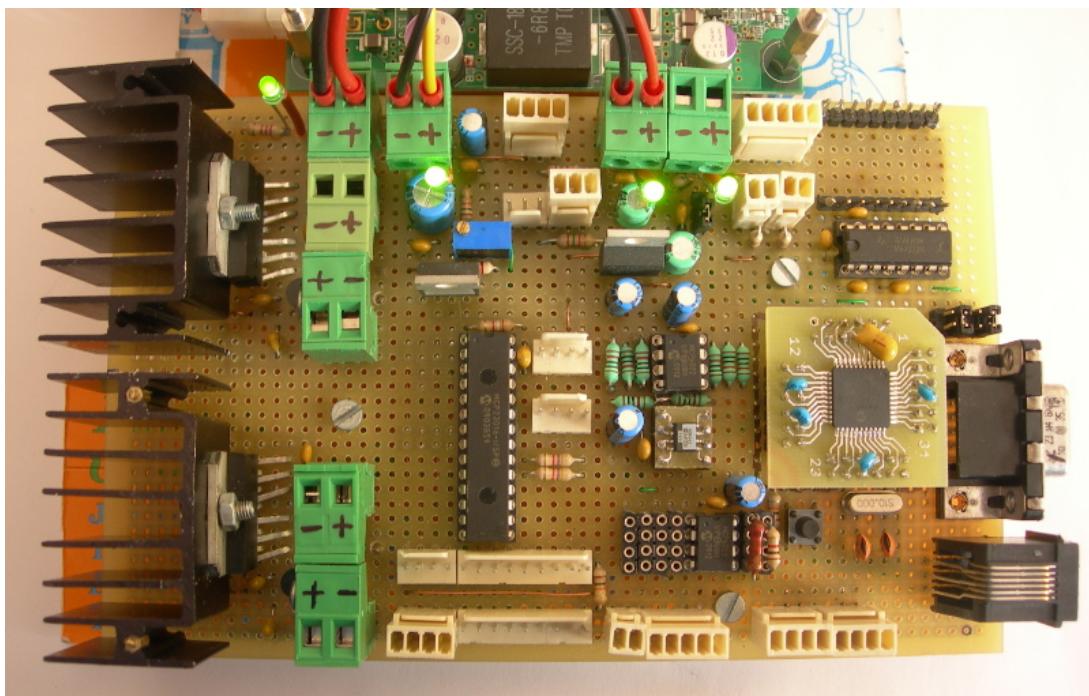
El CD-ROM adjunto a este libro contiene una copia de este repositorio.

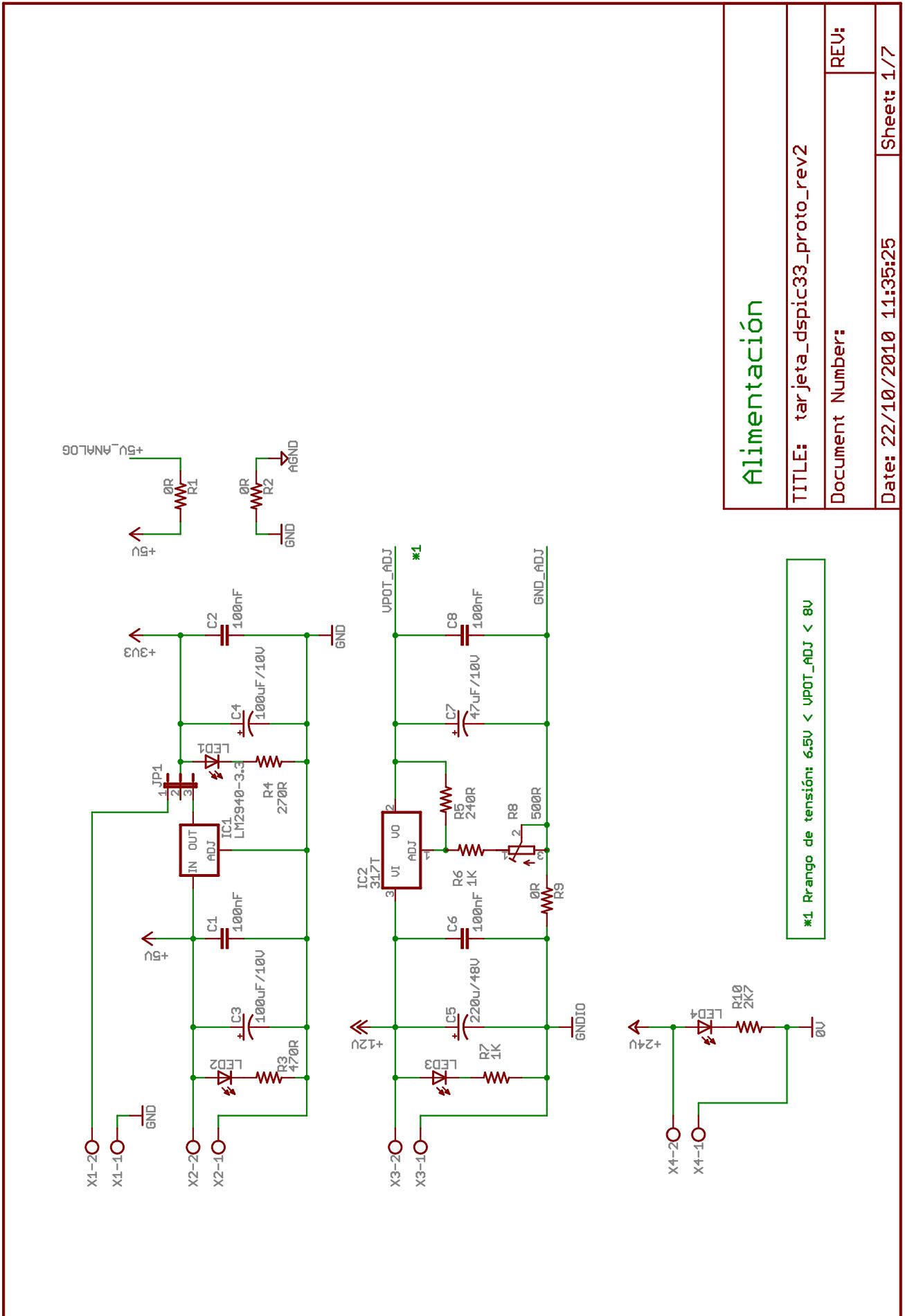
Además, en este capítulo, se ha considerado oportuno incluir una versión impresa los planos eléctricos de las tarjetas electrónicas.

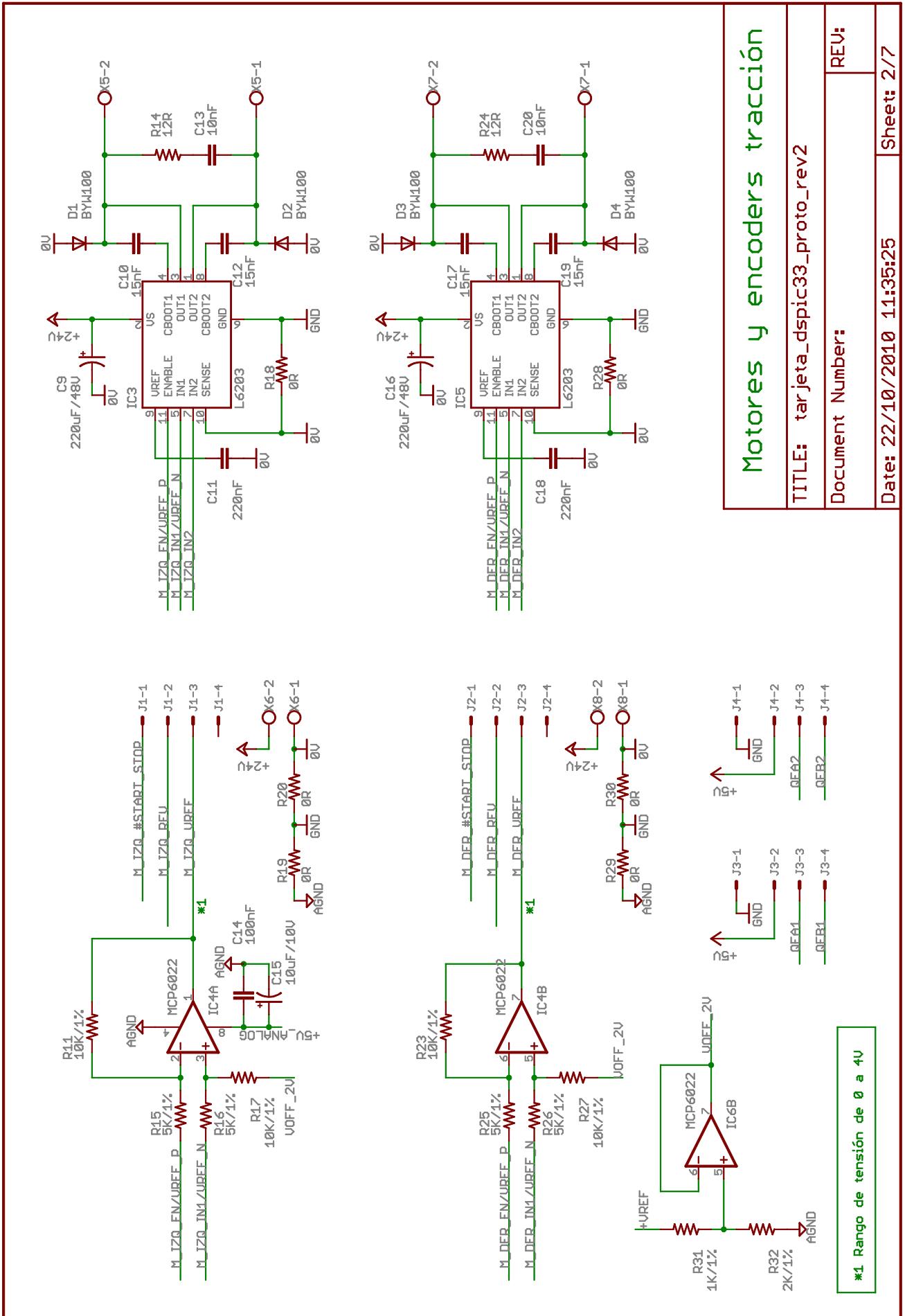


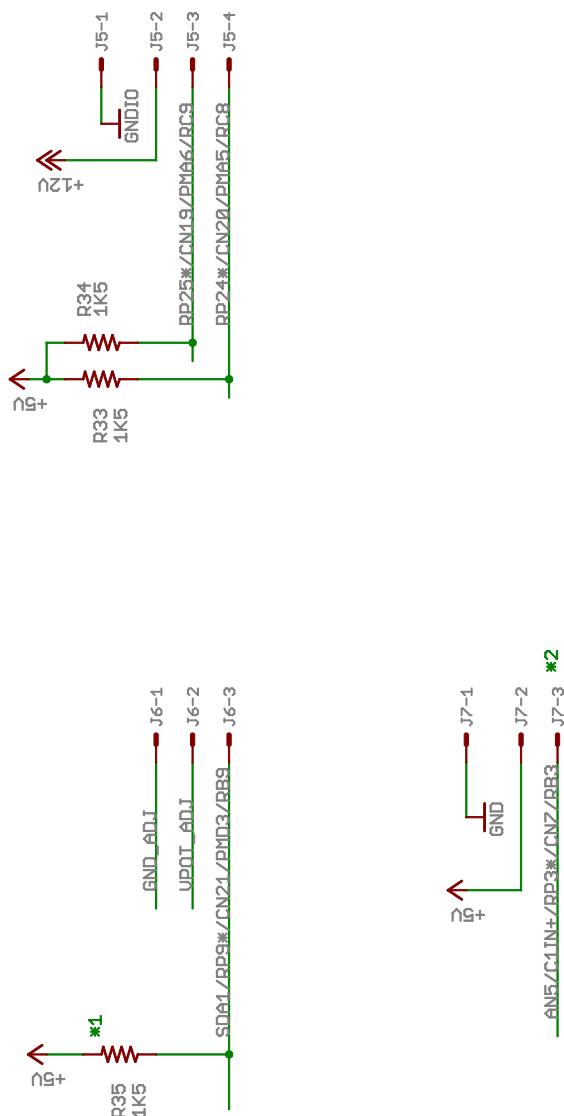
## A.1 Tarjeta *dspic33\_protoboard*

Diseño electrónico y PCB: Javier Baliñas Santos









## Baliza

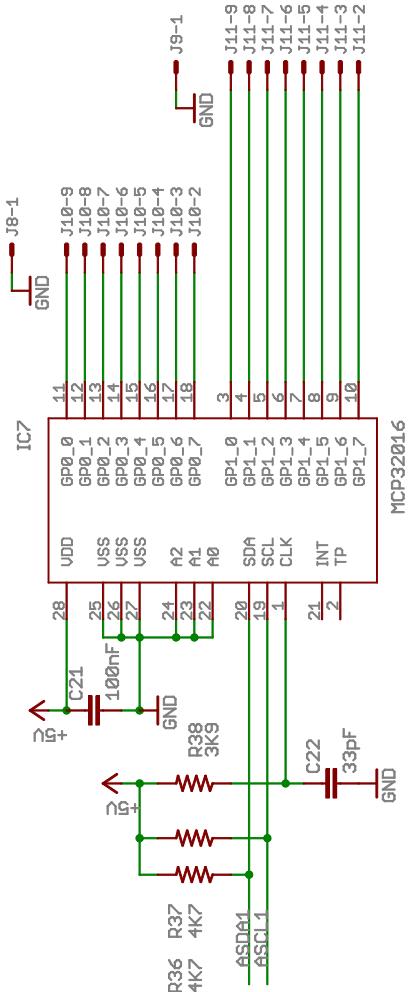
- \*1 Configurar pin del uC como colector abierto
- \*2 WARNING: no soporta tensiones mayores a 3Vctr abierto

TITLE: tarjeta\_dspic33\_proto\_rev2

Document Number:

REV:

Date: 22/10/2010 11:35:25 Sheet: 3/7



J10-1  
GND  
J10-9  
J10-8  
J10-7  
J10-6  
J10-5  
J10-4  
J10-3  
J10-2  
GND

J11-9  
J11-8  
J11-7  
J11-6  
J11-5  
J11-4  
J11-3  
J11-2  
GND

J13-4  
GND  
J13-2  
+5U  
J13-3  
GND  
J13-4  
+12U

J14-1  
+3U  
SD01  
SD04  
SCK1  
J14-4  
J14-5

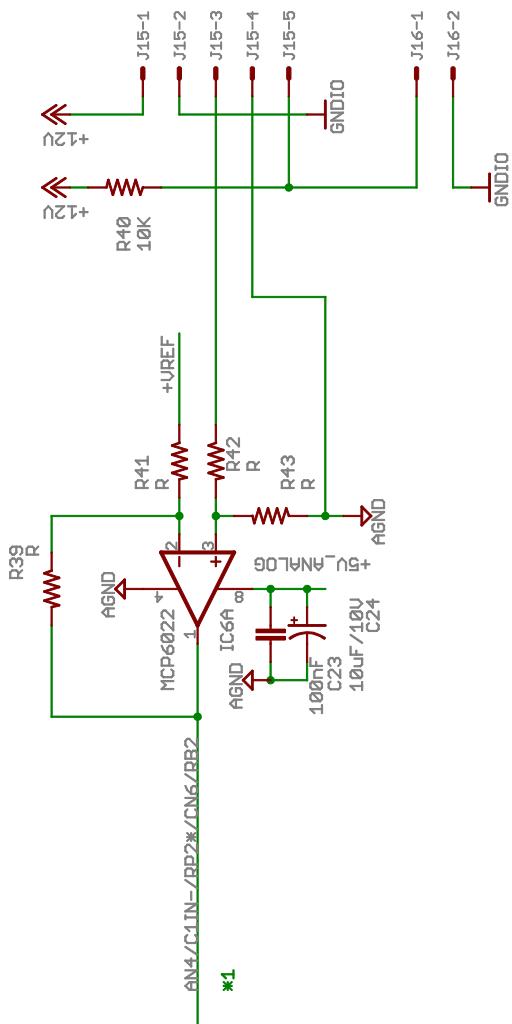
## Expansión E/S

**TITLE:** tarjeta\_dspic33\_proto\_rev2

**Document Number:**

**REV:**

**Date:** 22/10/2010 11:35:25    **Sheet:** 4/7



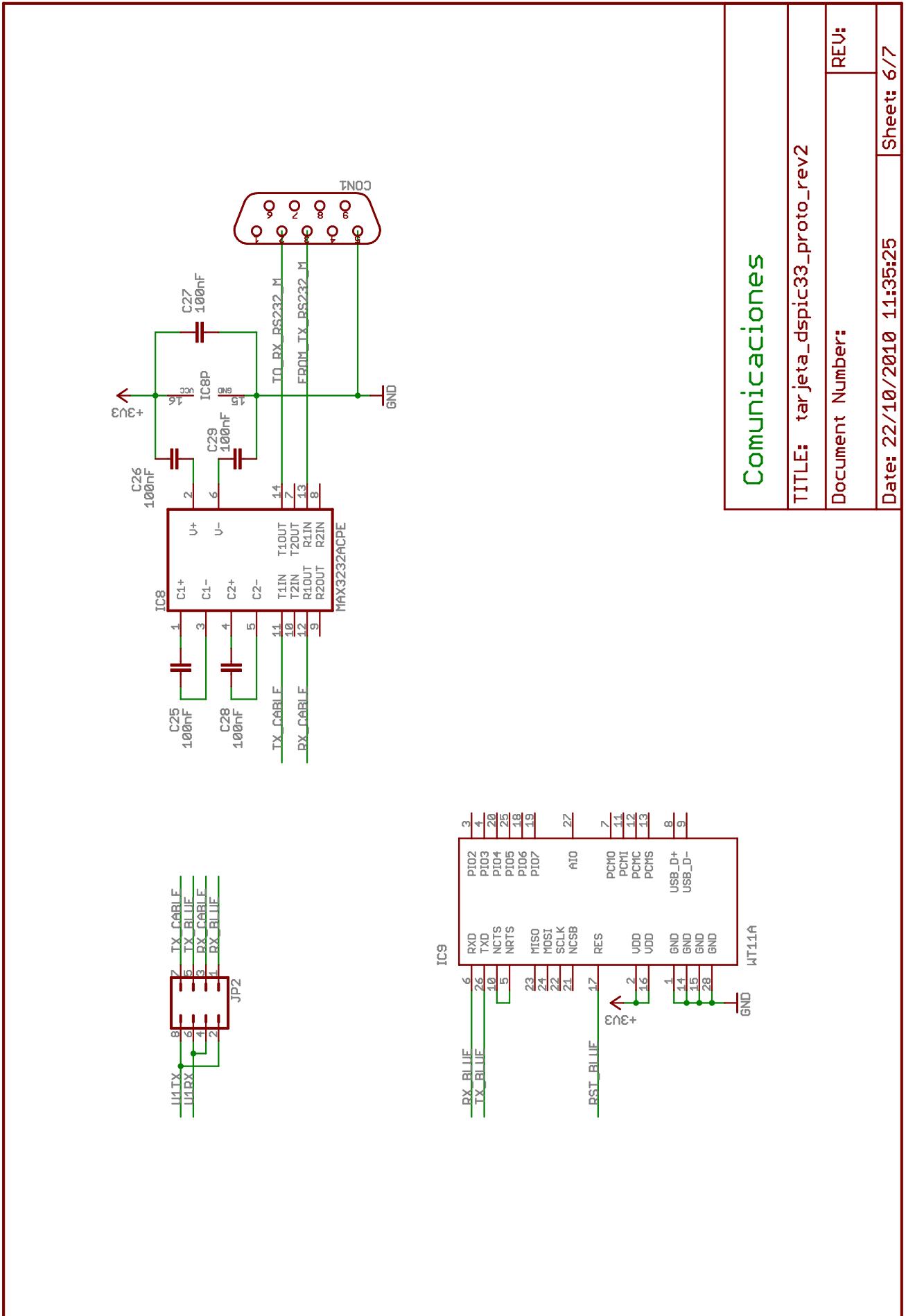
## Entradas analógicas

TITLE: tarjeta\_dspic33\_proto\_rev2

Document Number:

REV:

Date: 22/10/2010 11:35:25 Sheet: 5/7



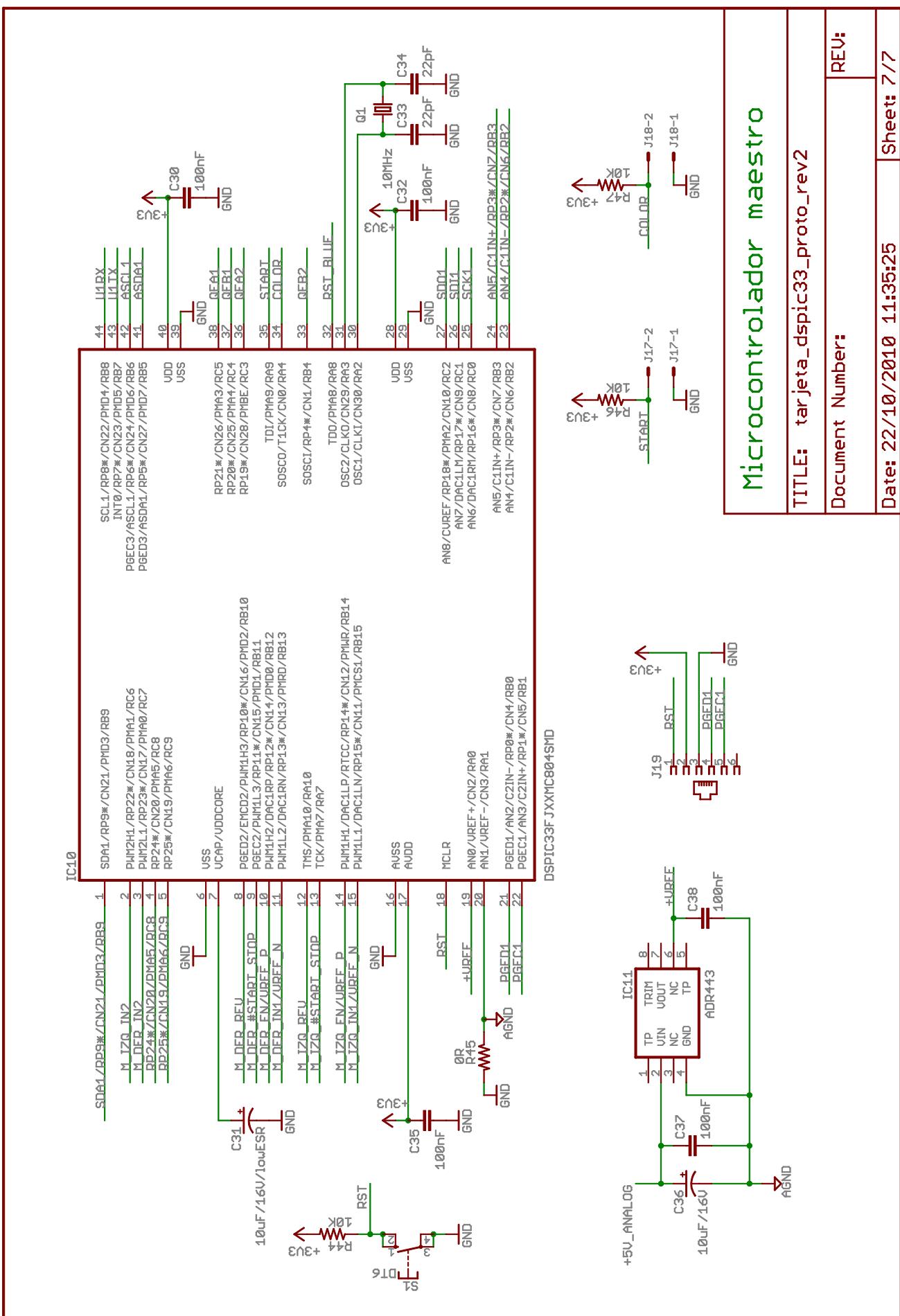
## Comunicaciones

TITLE: tarjeta\_dspic33\_proto\_rev2

Document Number:

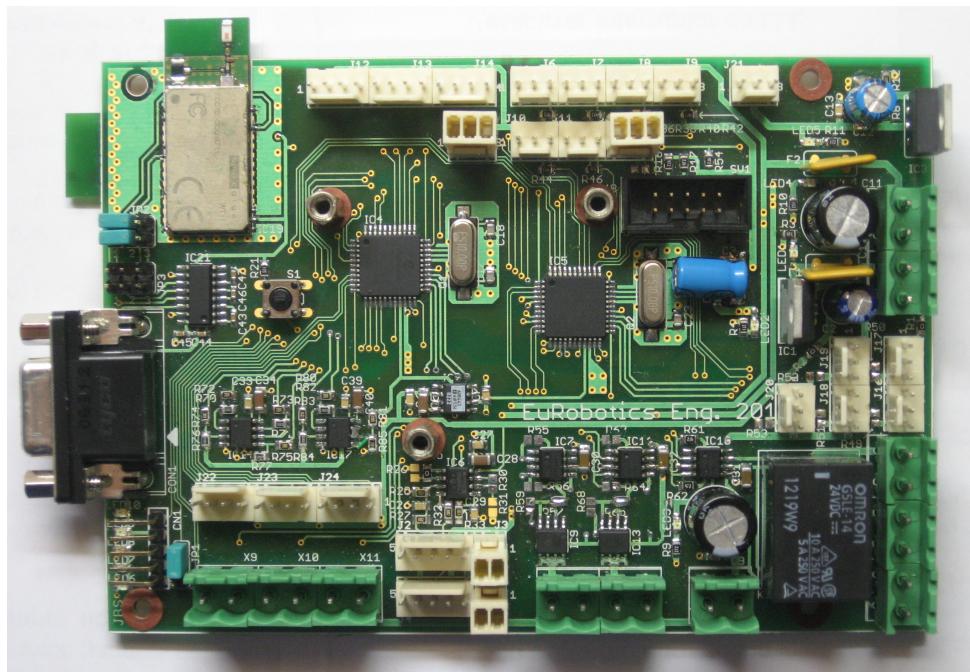
REV:

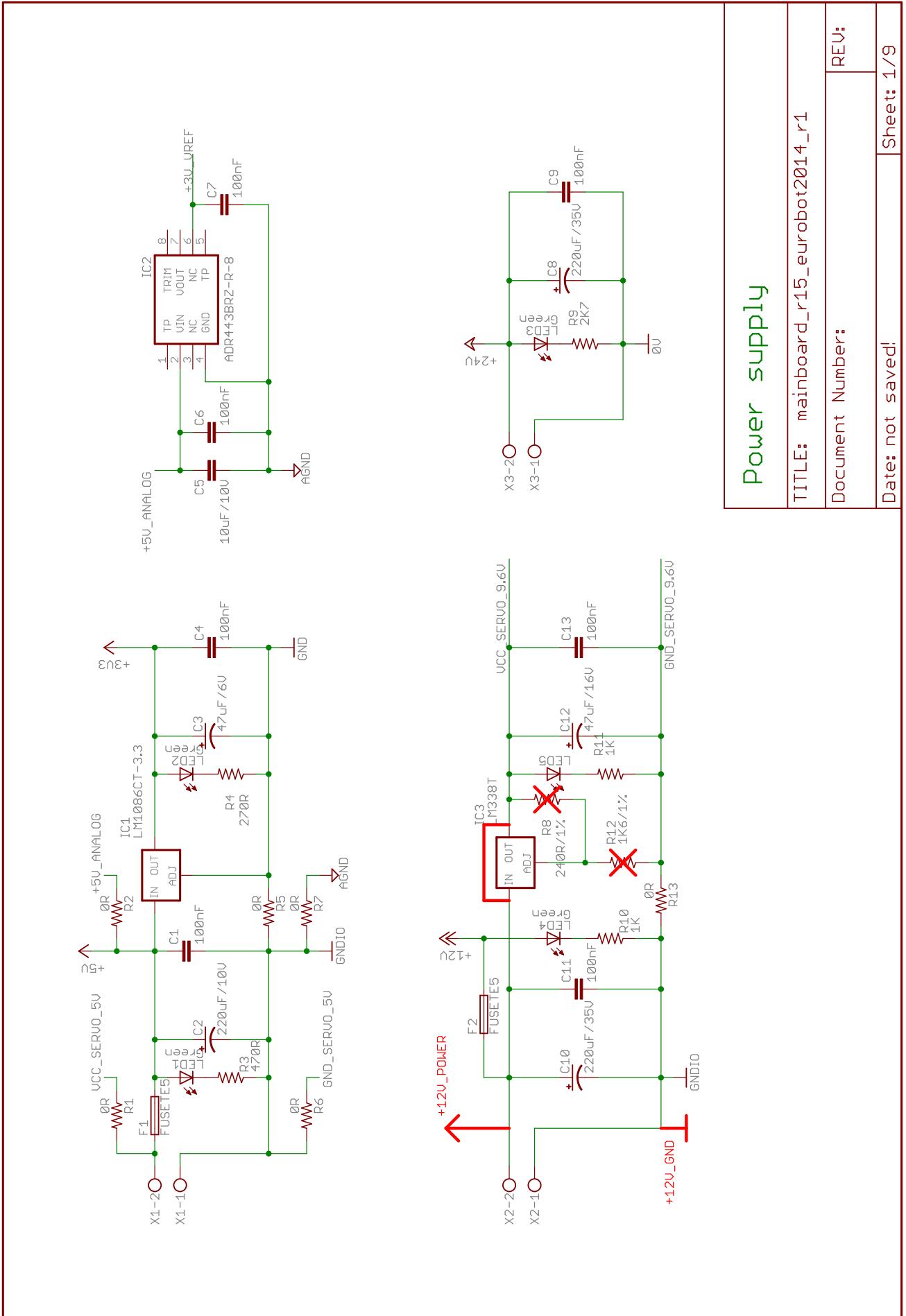
Date: 22/10/2010 11:35:25 Sheet: 6/7

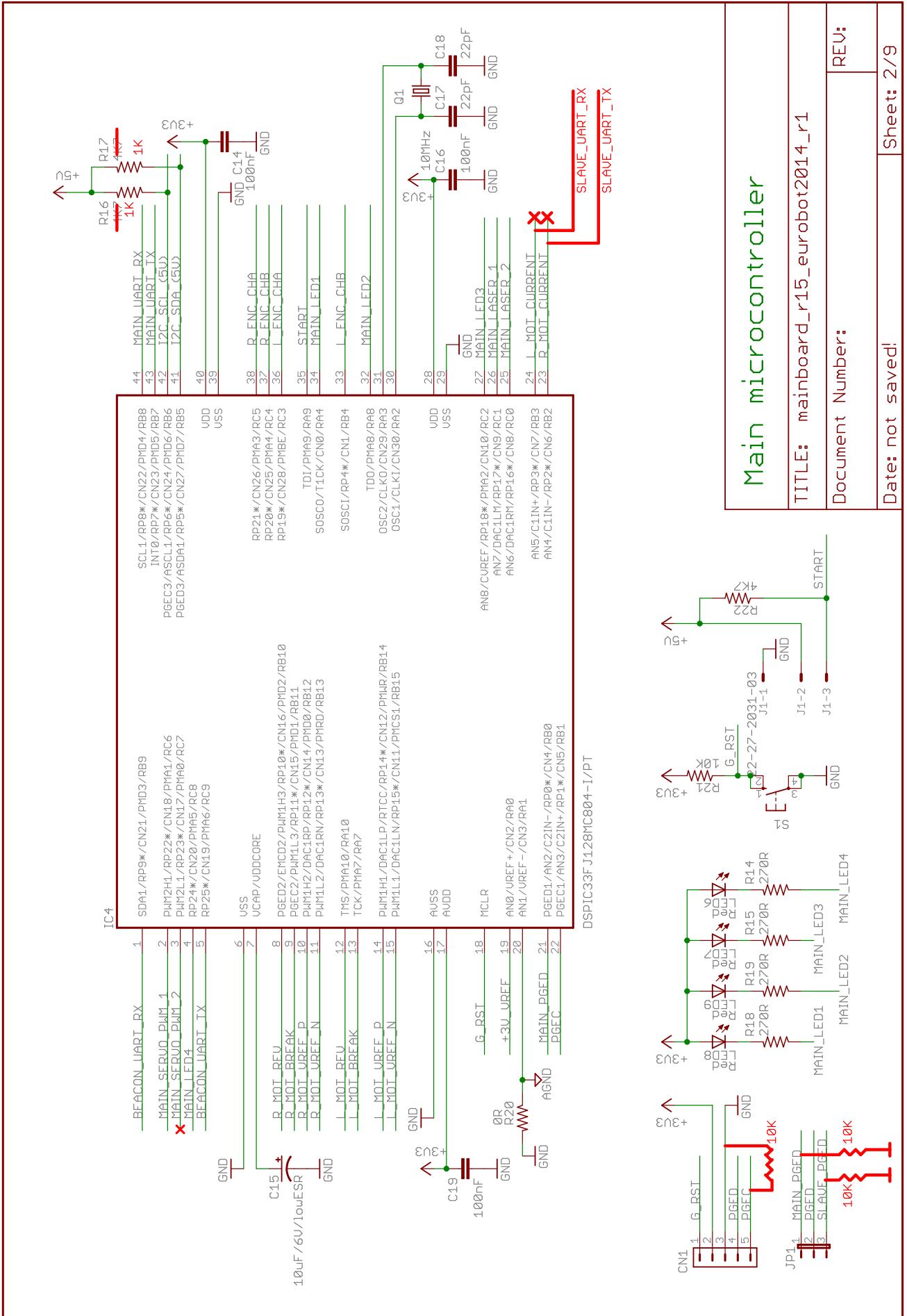


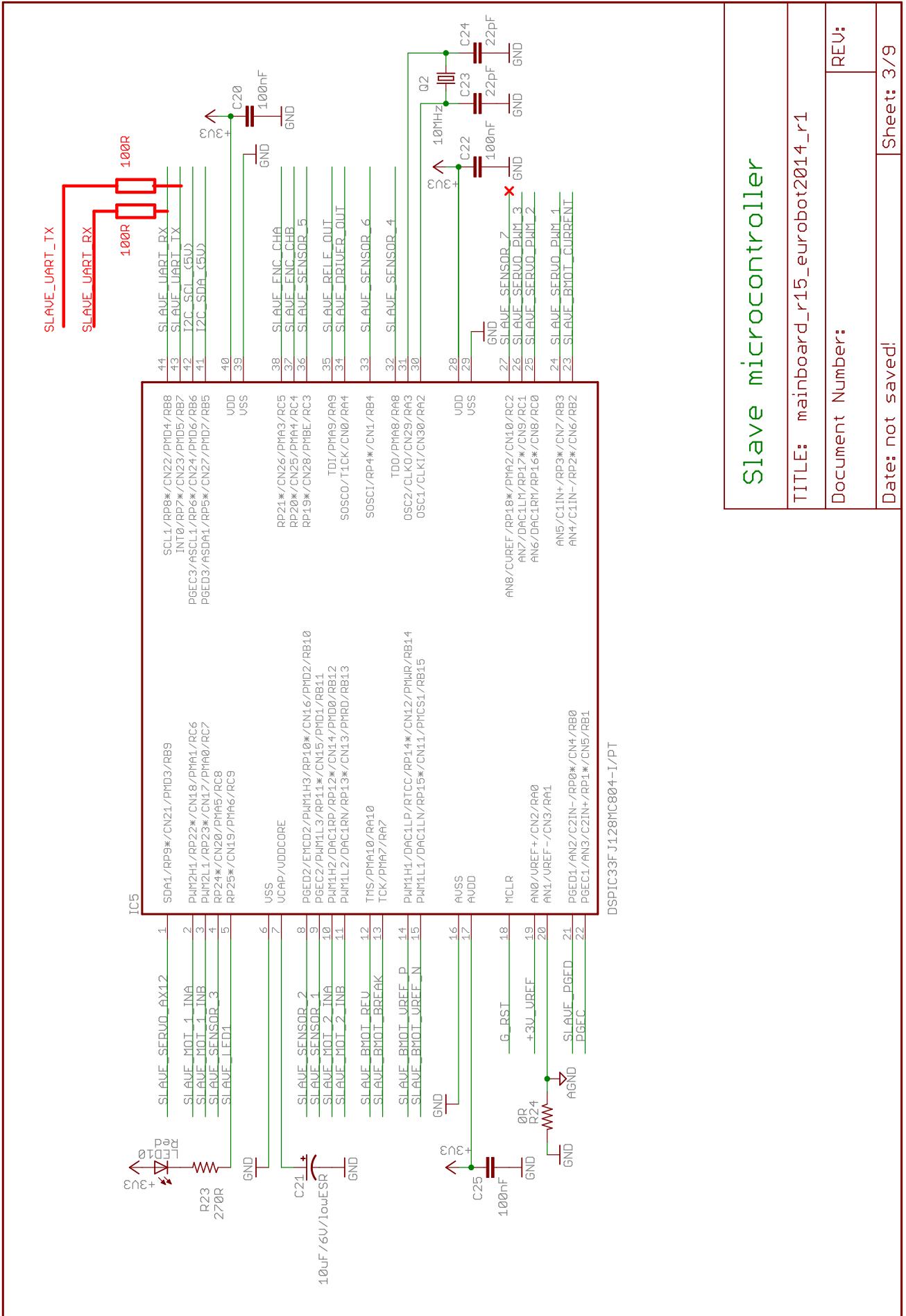
## A.2 Tarjeta *mainboard\_v02*

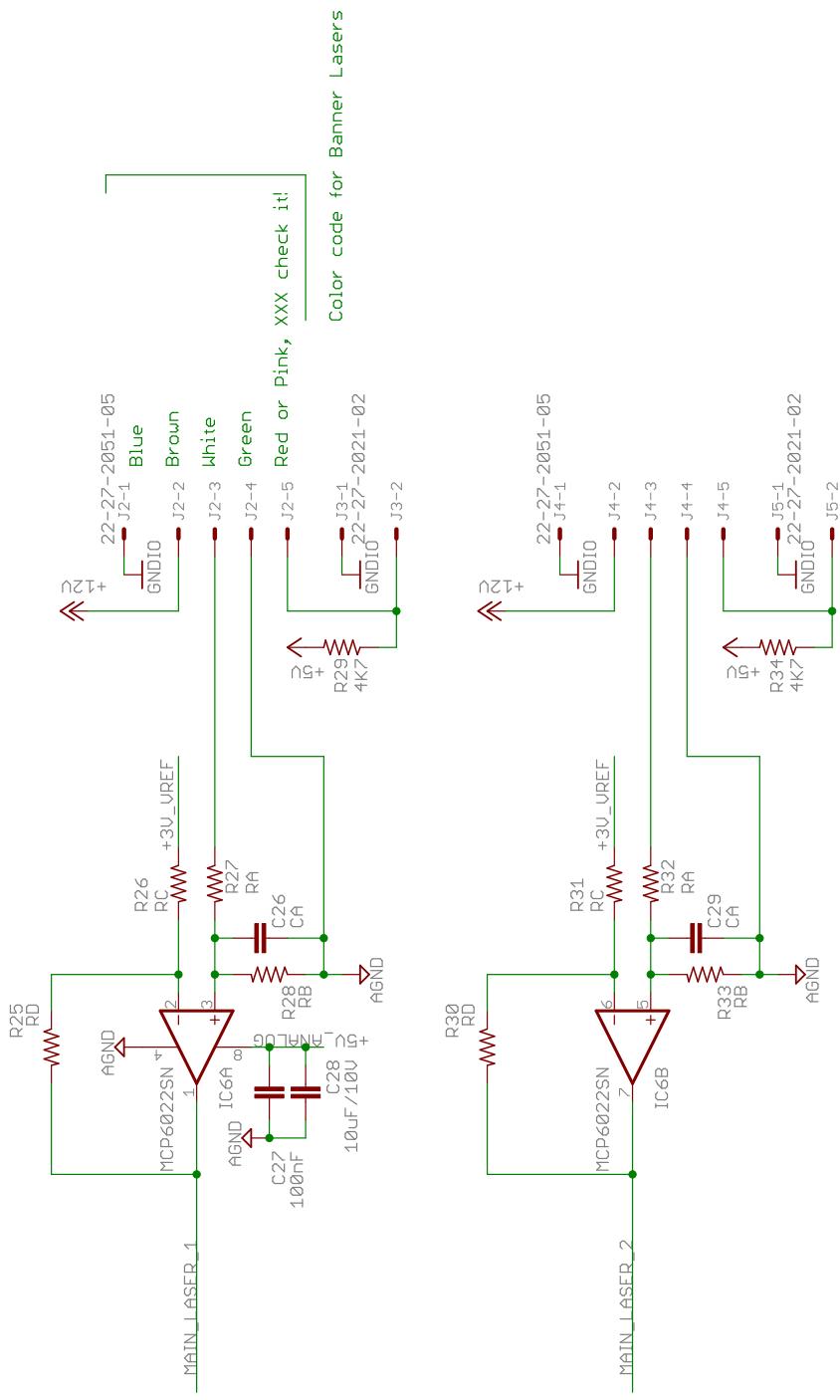
Diseño electrónico y PCB: Javier Baliñas Santos











### Analog Sensors

TITLE: mainboard\_r15\_eurobot2014\_r1

Document Number:

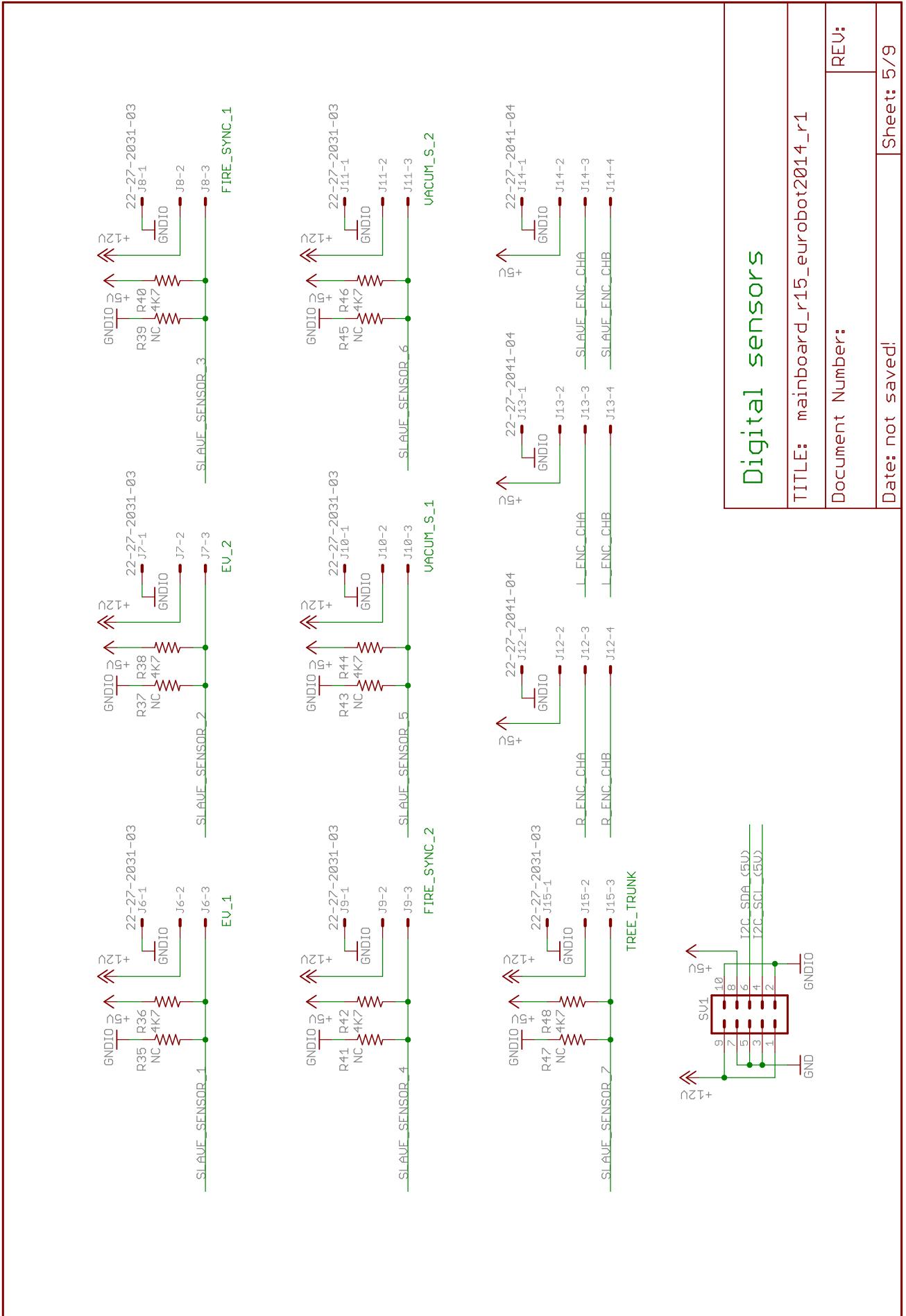
Date: not saved!

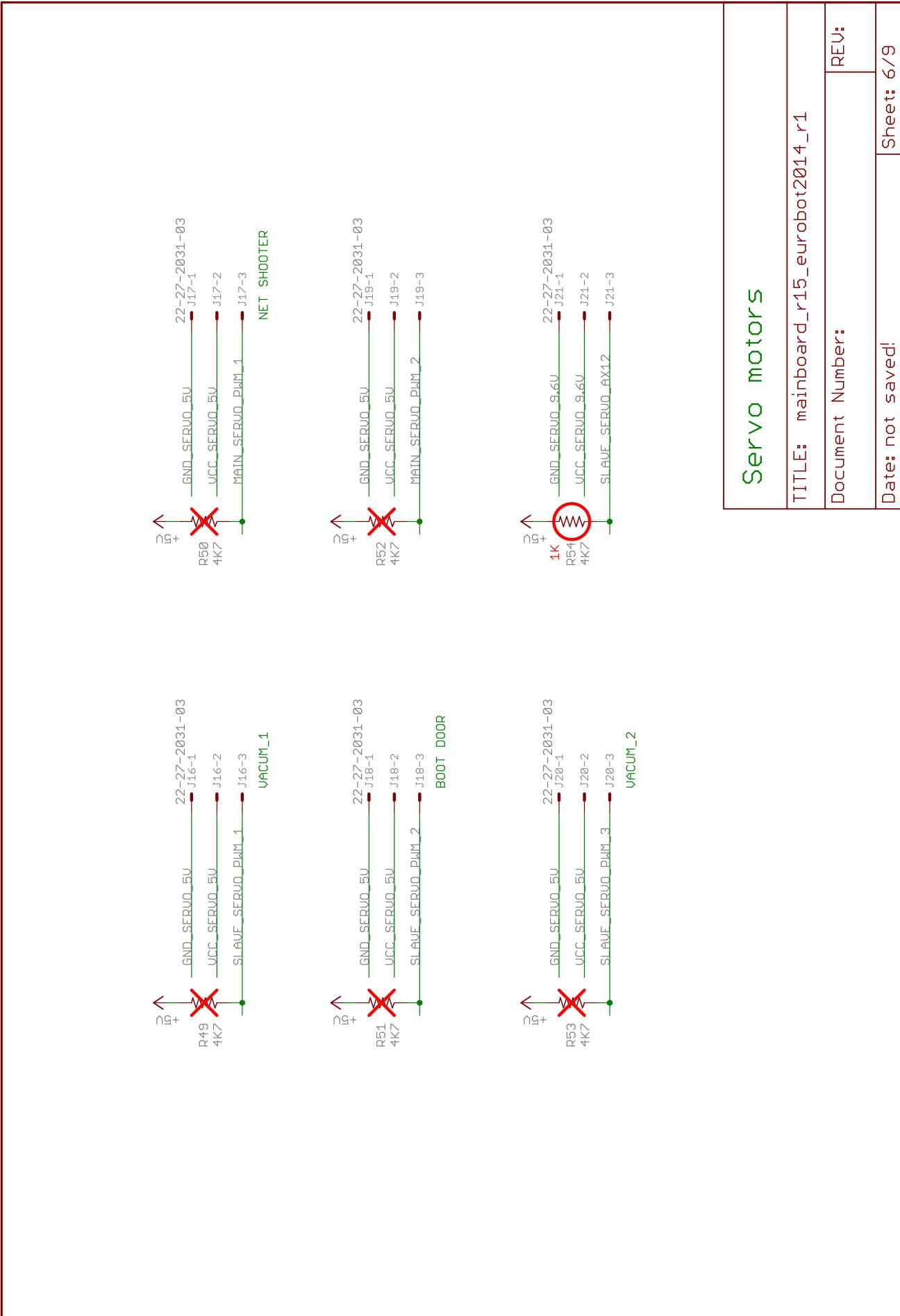
REV:

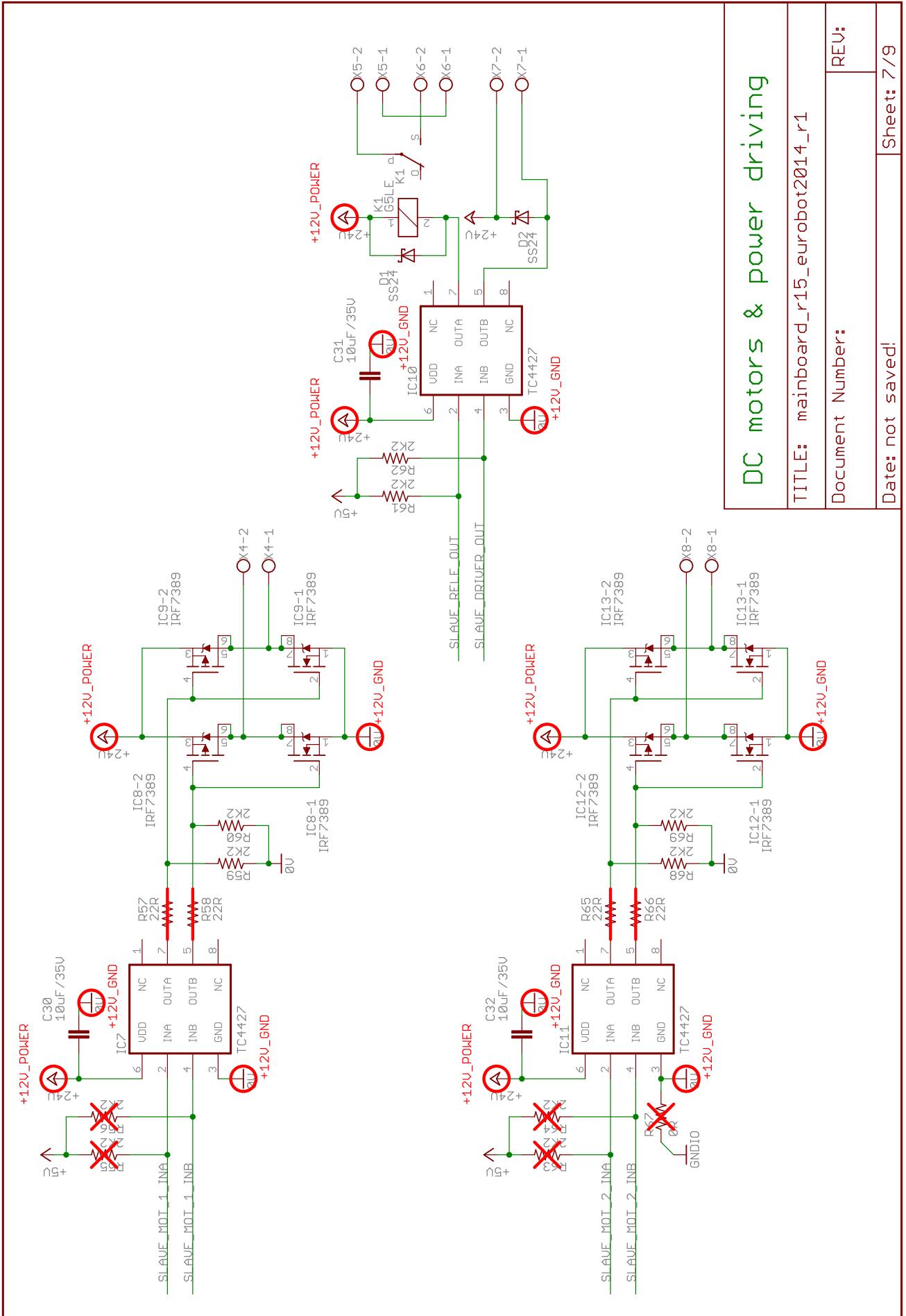
Sheet: 4/9

Output laser type	Component values				
	R <sub>A</sub>	R <sub>B</sub>	R <sub>C</sub>	R <sub>D</sub>	C <sub>A</sub>
Voltage	10k/1%	4k3/1%	NC	0R	1.5uF
Current	0R	187R/1%	1k1/1%	3k3/1%	56nF

Note: In voltage type R<sub>A</sub> = 2.33RB and f<sub>c</sub> = 661Hz  
Note: In current type RD = 3RC and f<sub>c</sub> = 567Hz







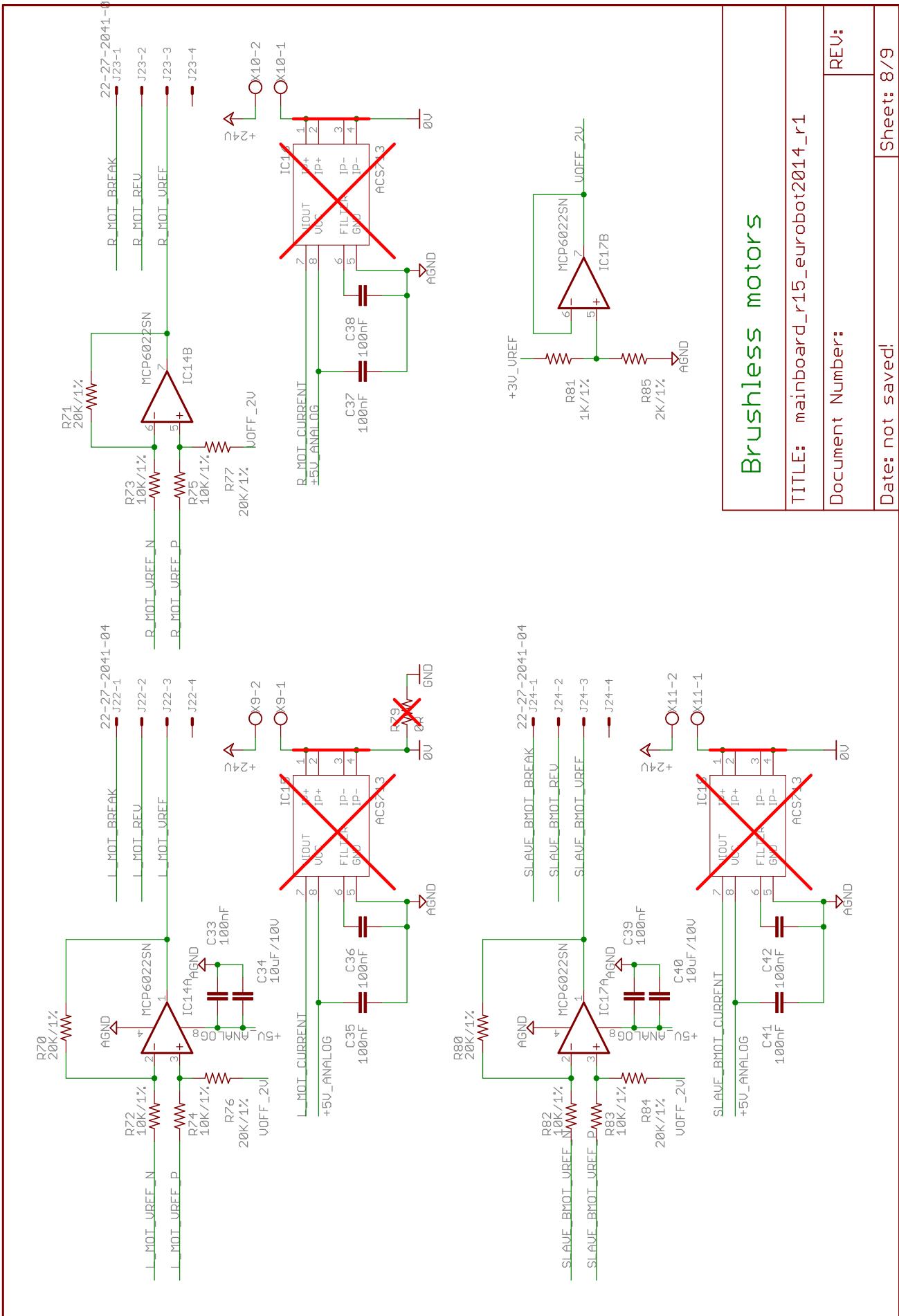
## DC motors & power driving

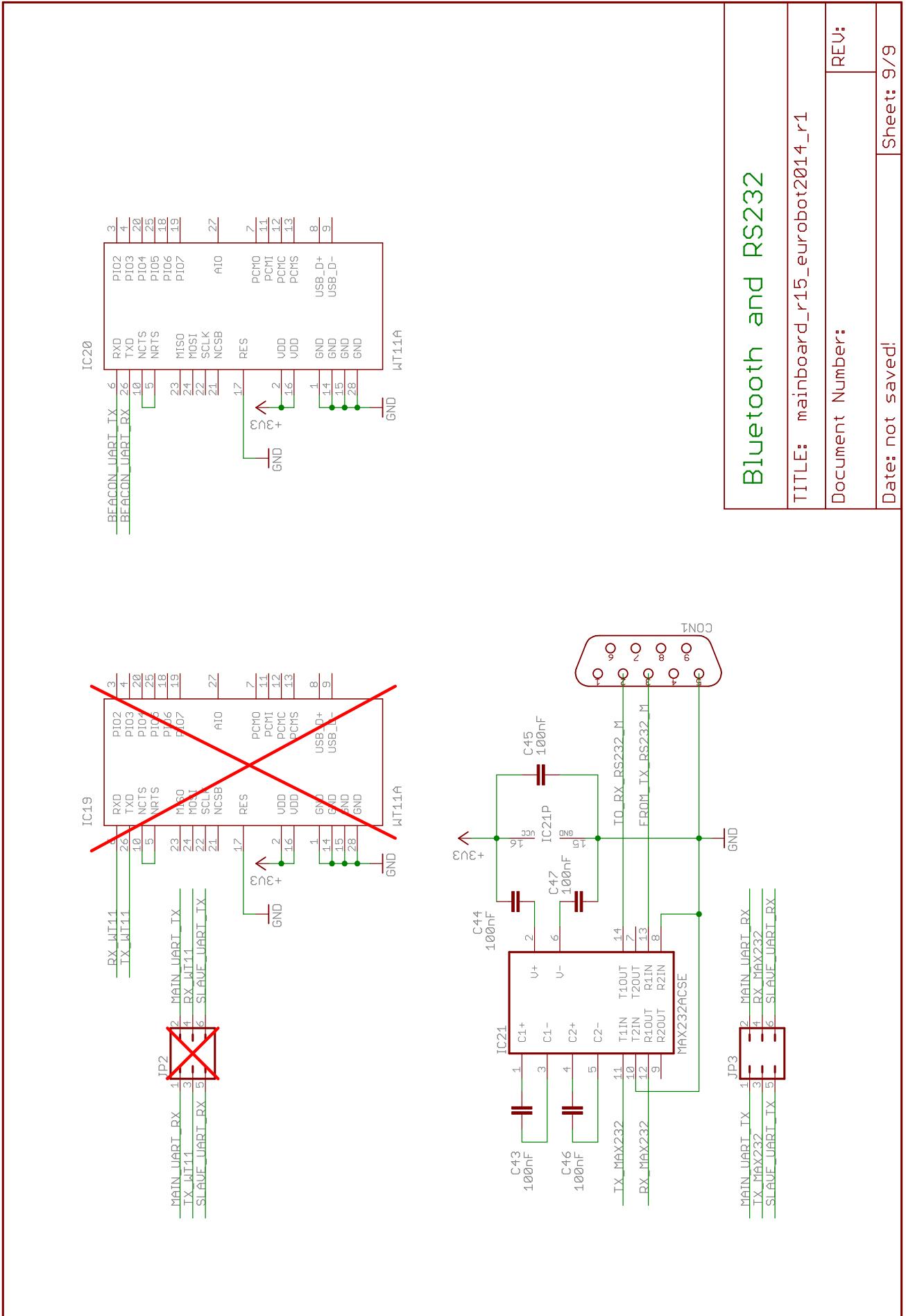
TITLE: mainboard\_r15\_eurobot2014\_r1

Document Number:

Date: not saved! Sheet: 7/9

REV:





## Bluetooth and RS232

TITLE: mainboard\_r15\_eurobot2014\_r1

Document Number:

Date: not saved!	Sheet: 9/9
------------------	------------

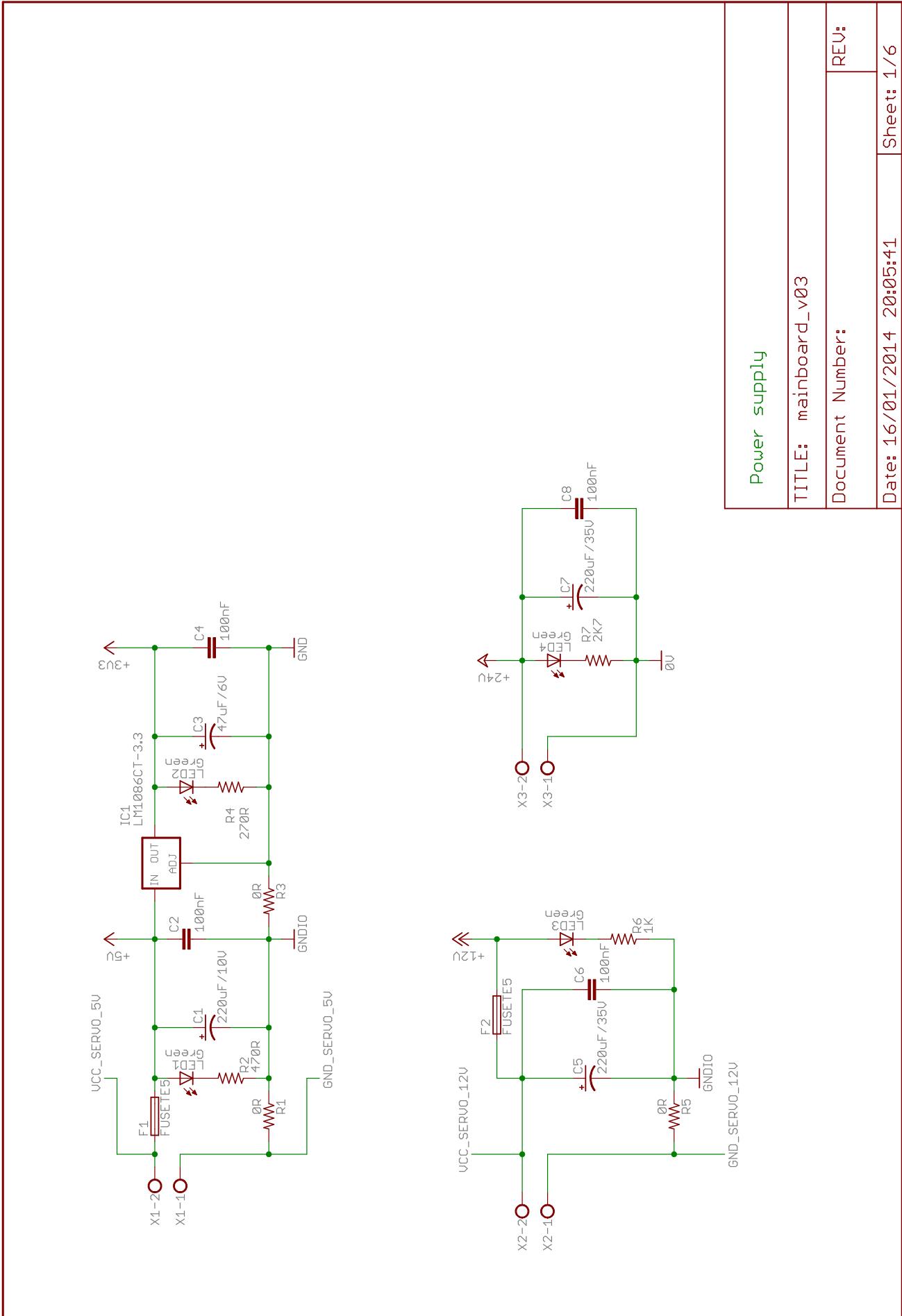
REV:

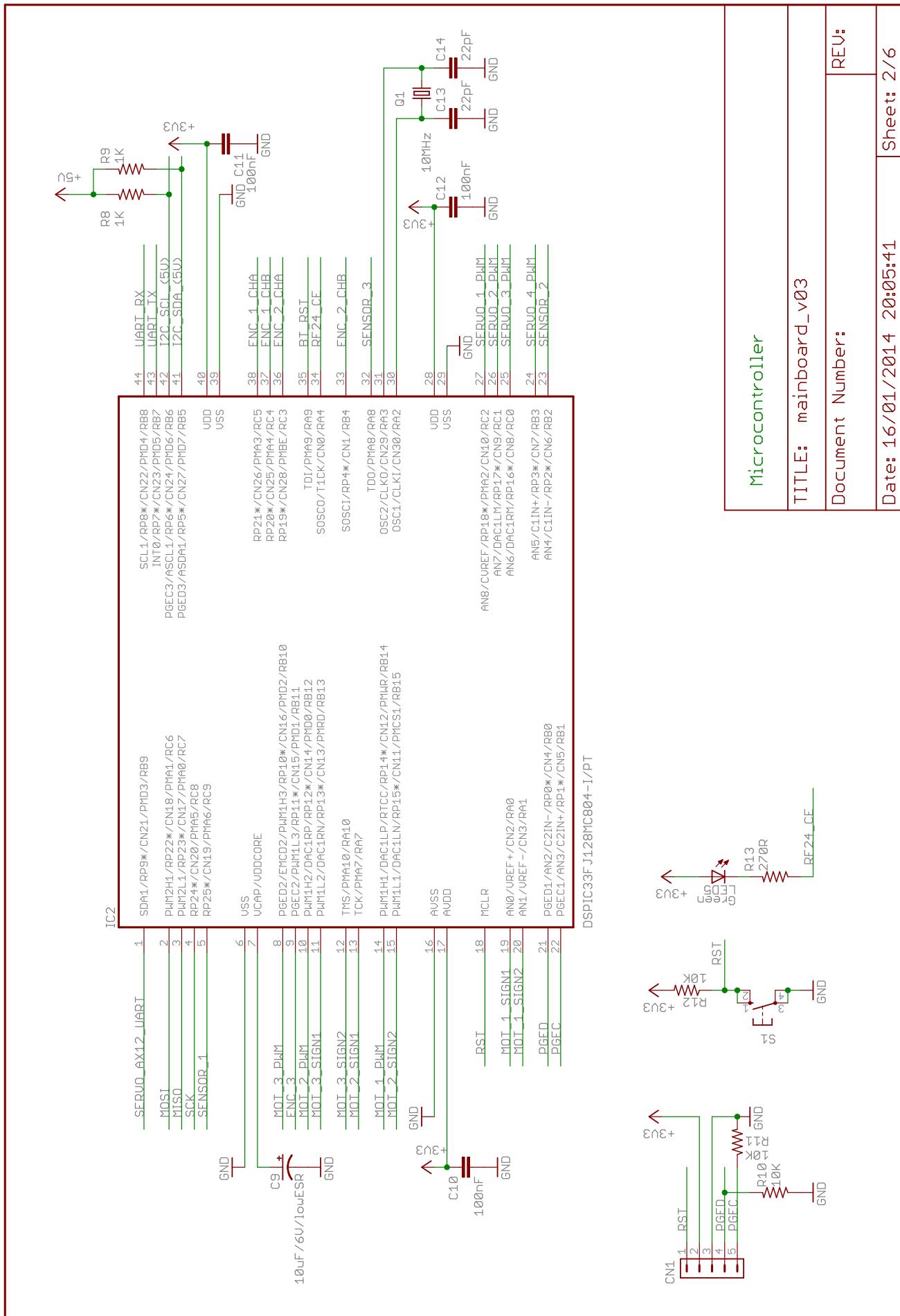
### A.3 Tarjeta *mainboard\_v03*

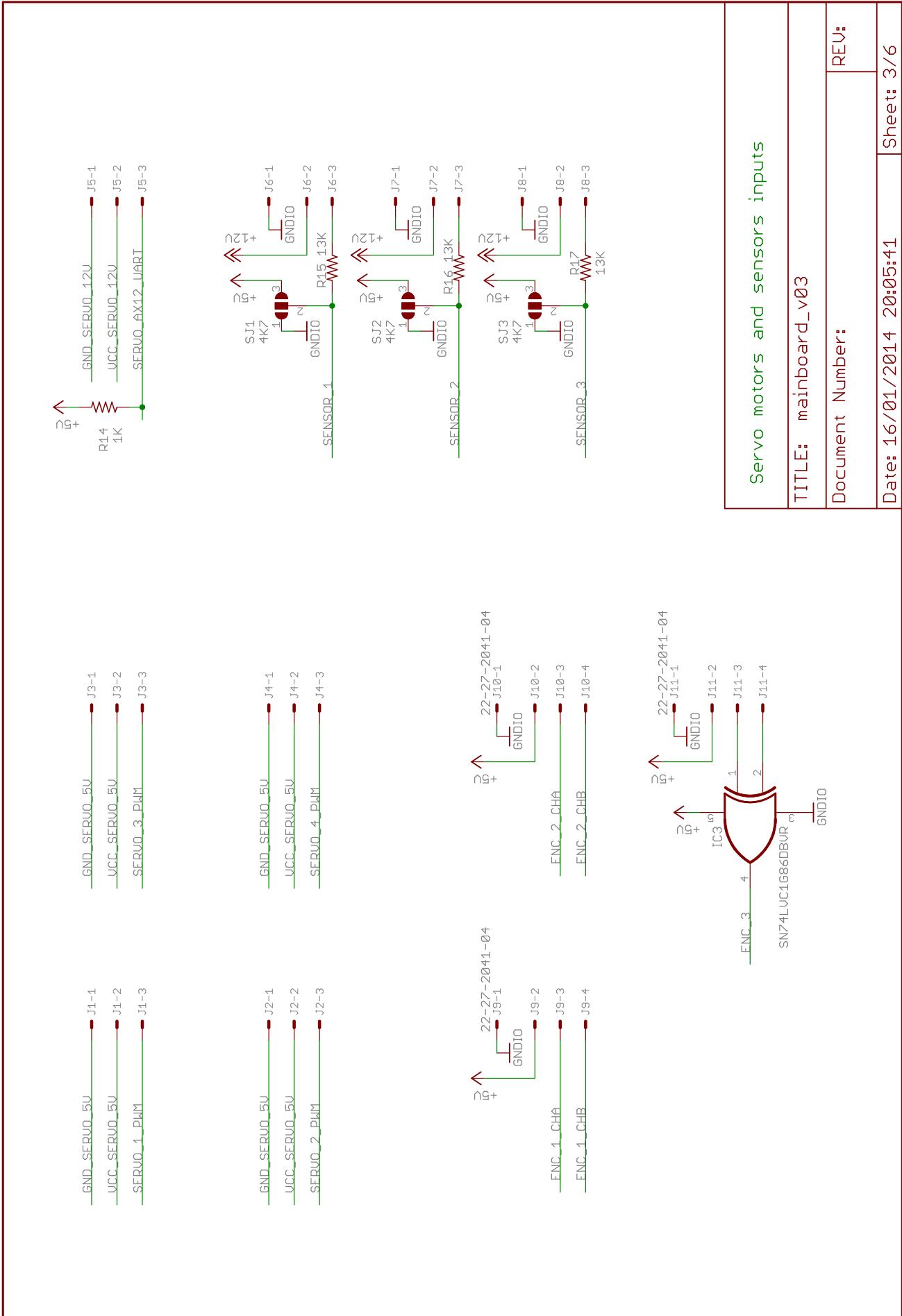
Diseño electrónico: Javier Baliñas Santos

Diseño PCB: Rubén Espino San José









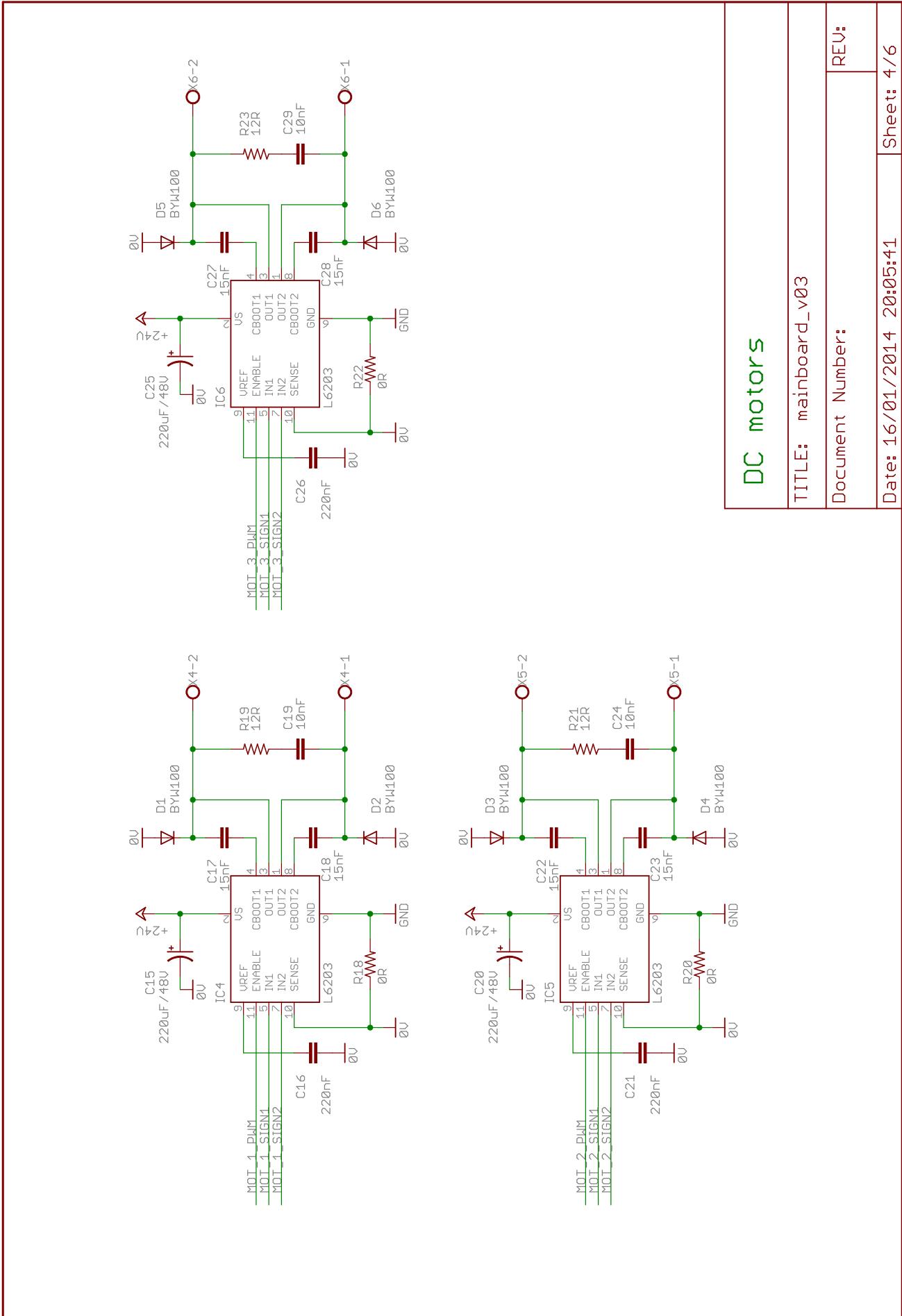
Servo motors and sensors inputs

TITLE: mainboard\_v03

Document Number:

REV:

Date: 16/01/2014 20:05:41 Sheet: 3/6

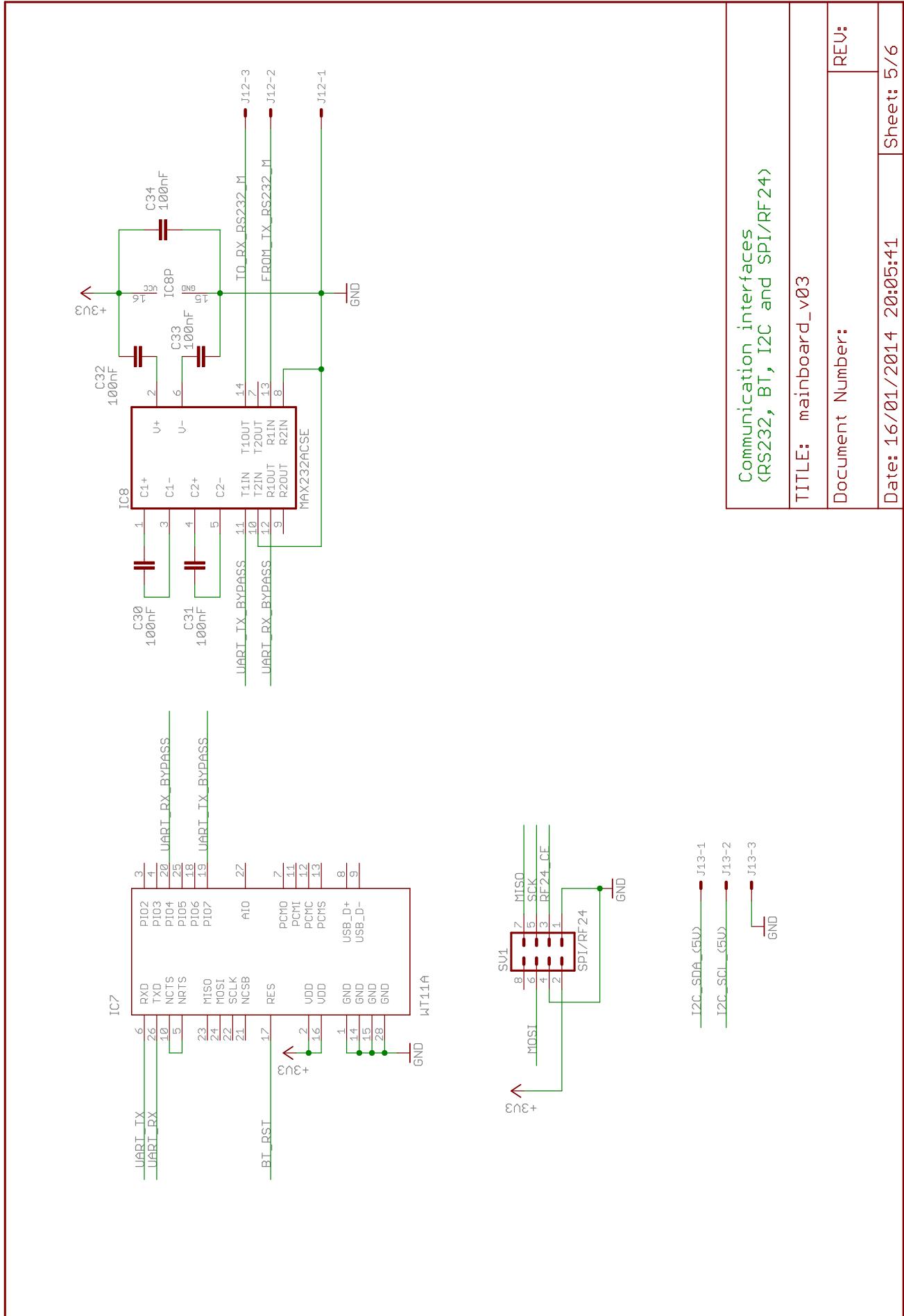


## DC motors

TITLE: mainboard\_v03  
Document Number:  
Date: 16/01/2014 20:05:41

REV:

Sheet: 4/6



Communication interfaces  
<RS232, BT, I2C and SPI/RF24>

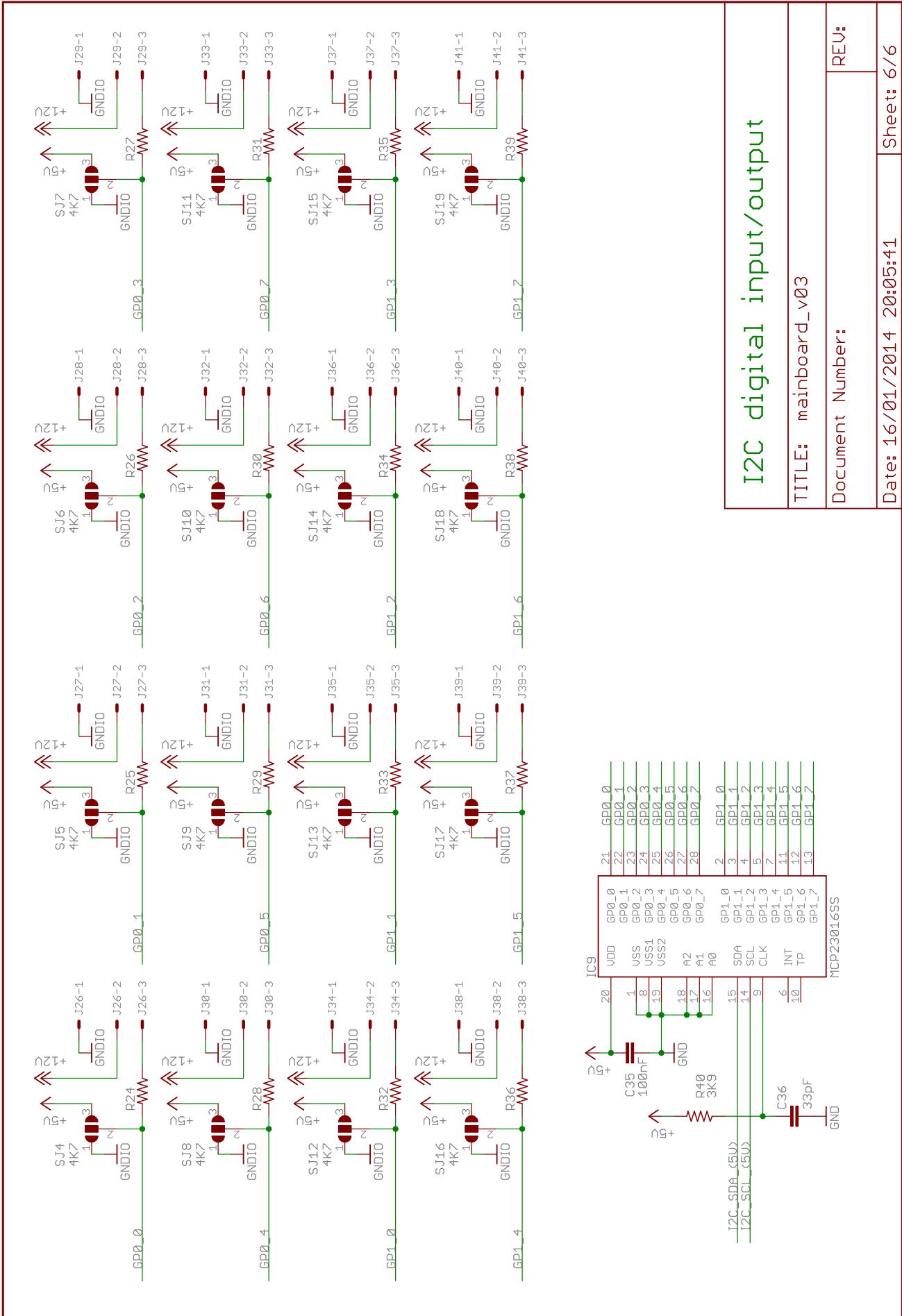
TITLE: mainboard\_v03

Document Number:

Date: 16/01/2014 20:05:41

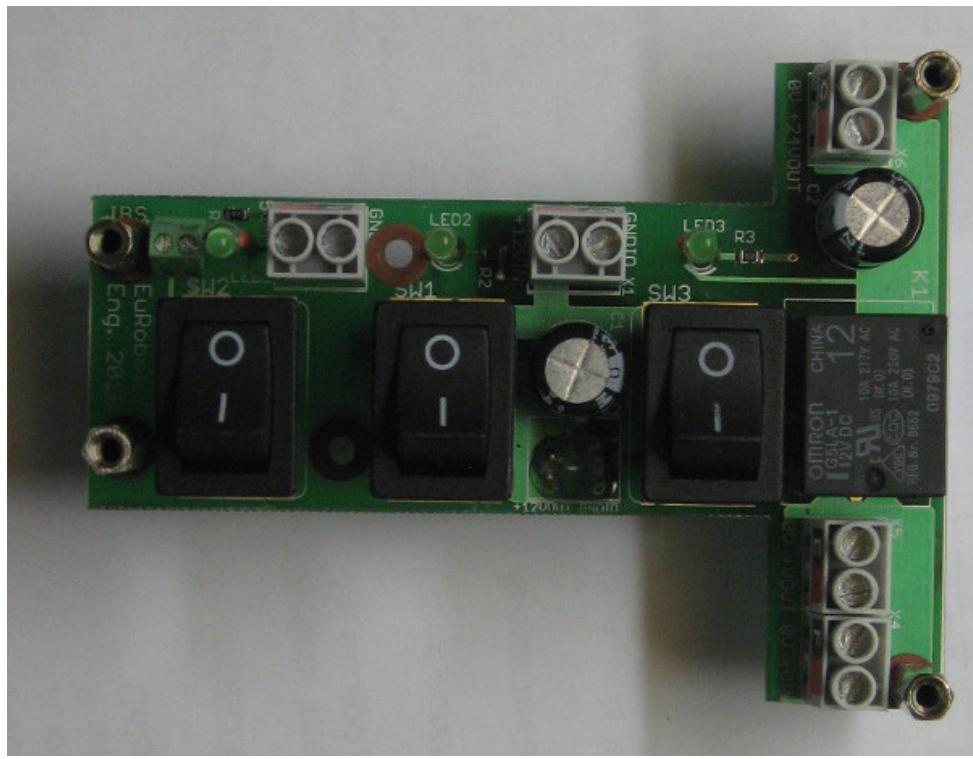
REV:

Sheet: 5/6

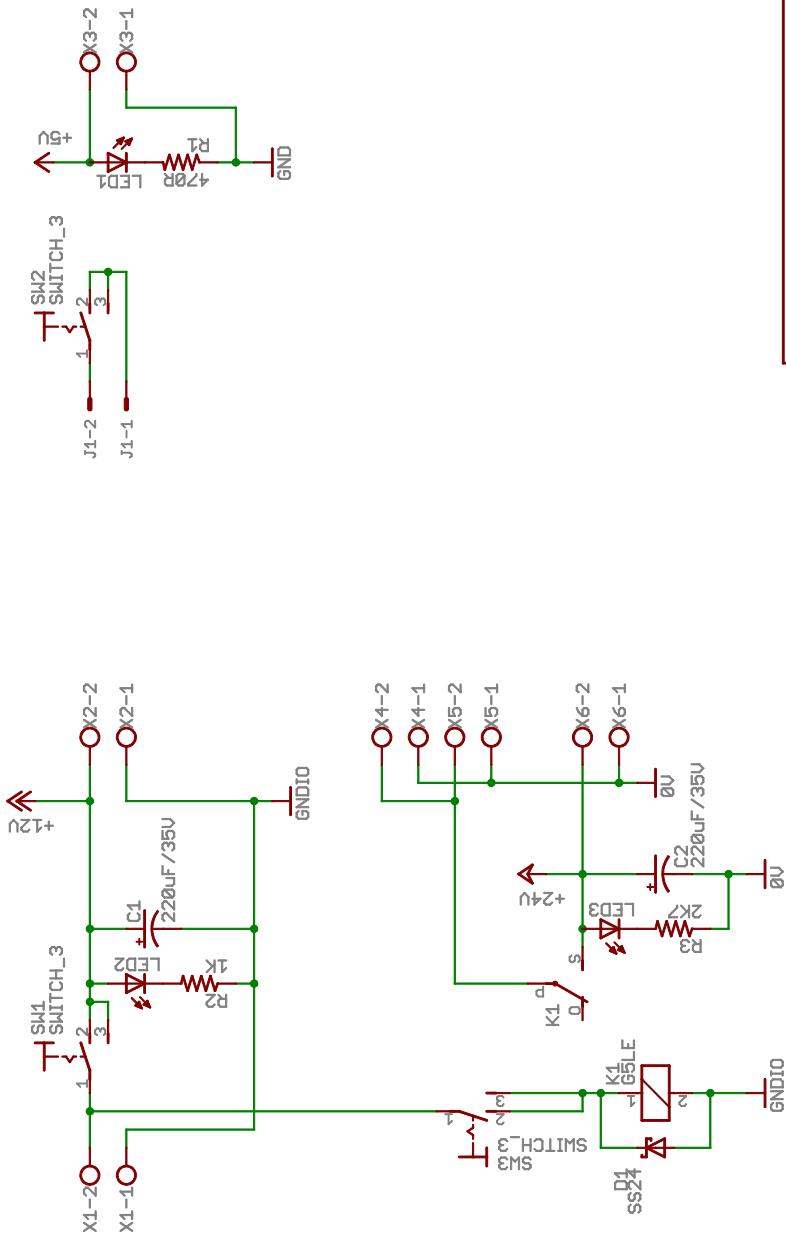


## A.4 Tarjeta *switchboard*

Diseño electrónico y PCB: Javier Baliñas Santos



## Power Supply ON/OFF



TITLE: switchboard\_eurobotics2010\_r3

Document Number:

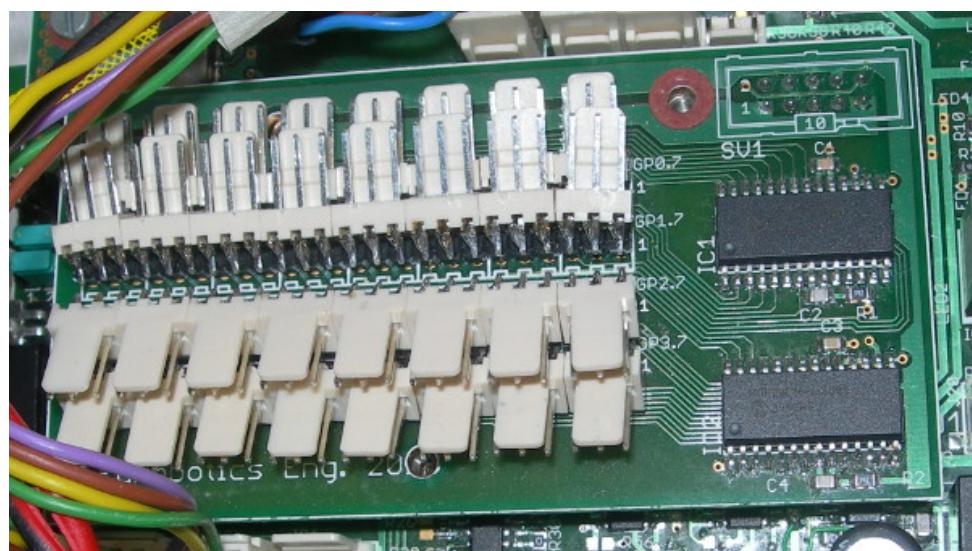
Date: 03/10/2010 23:44:54

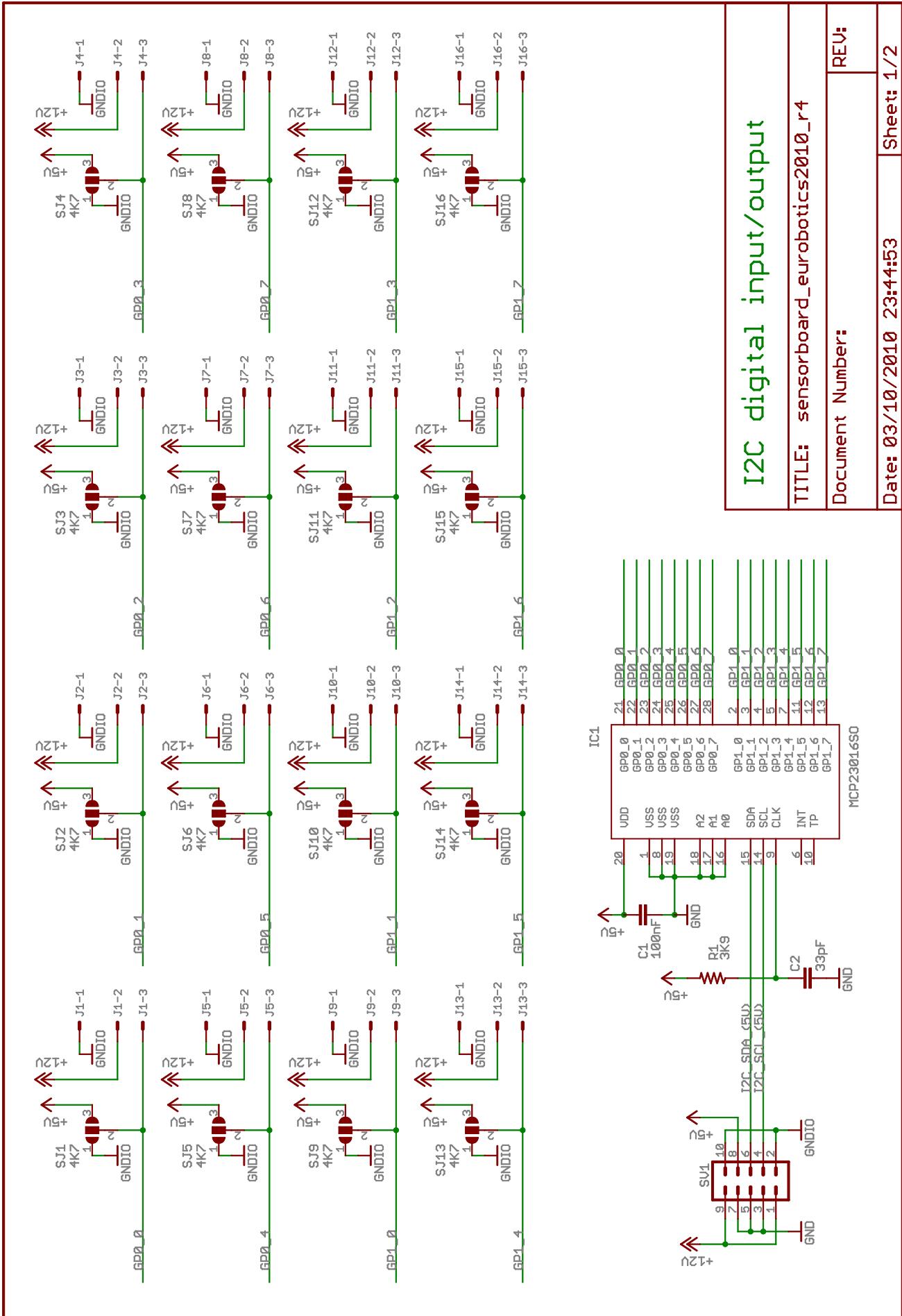
REV:

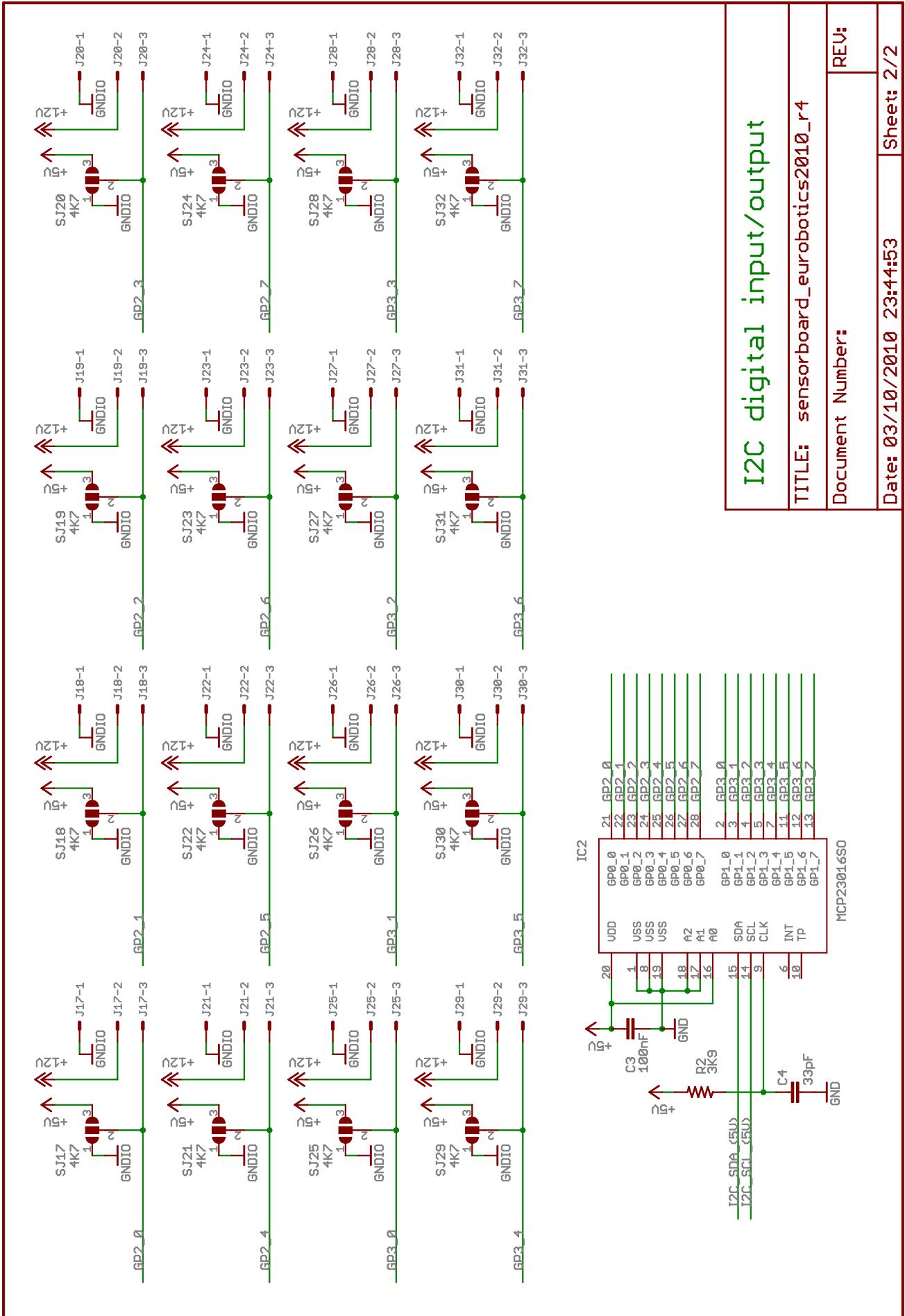
Sheet: 1/1

## A.5 Tarjeta *sensorboard*

Diseño electrónico y PCB: Javier Baliñas Santos

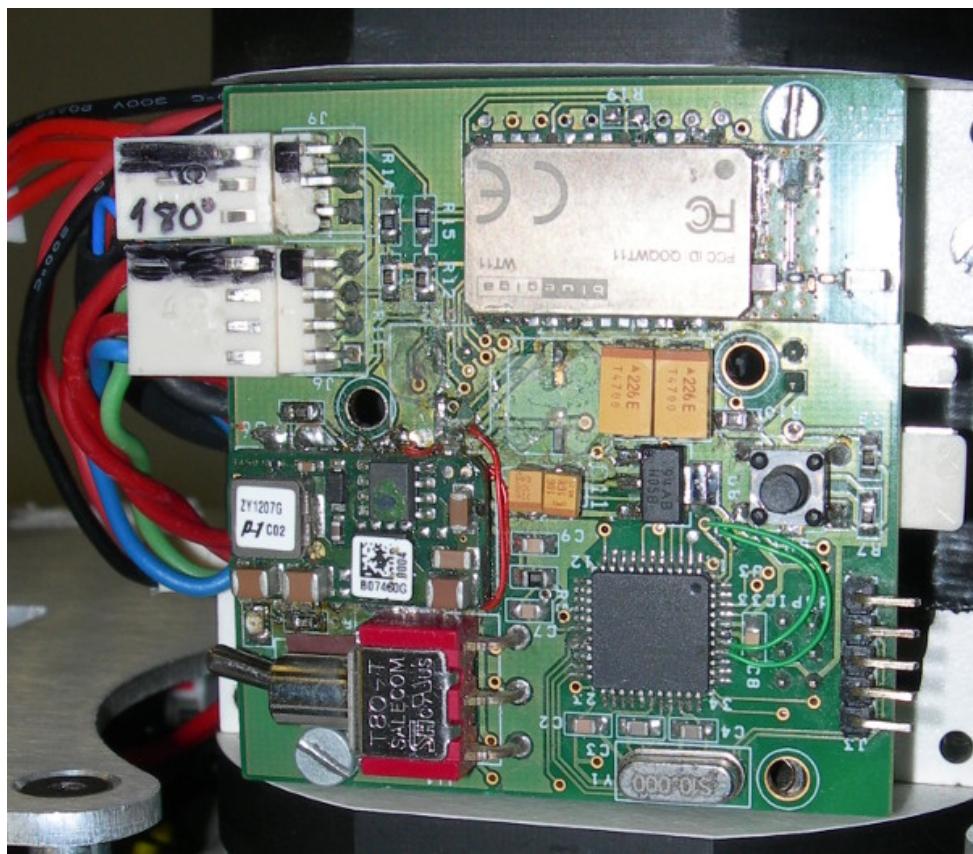


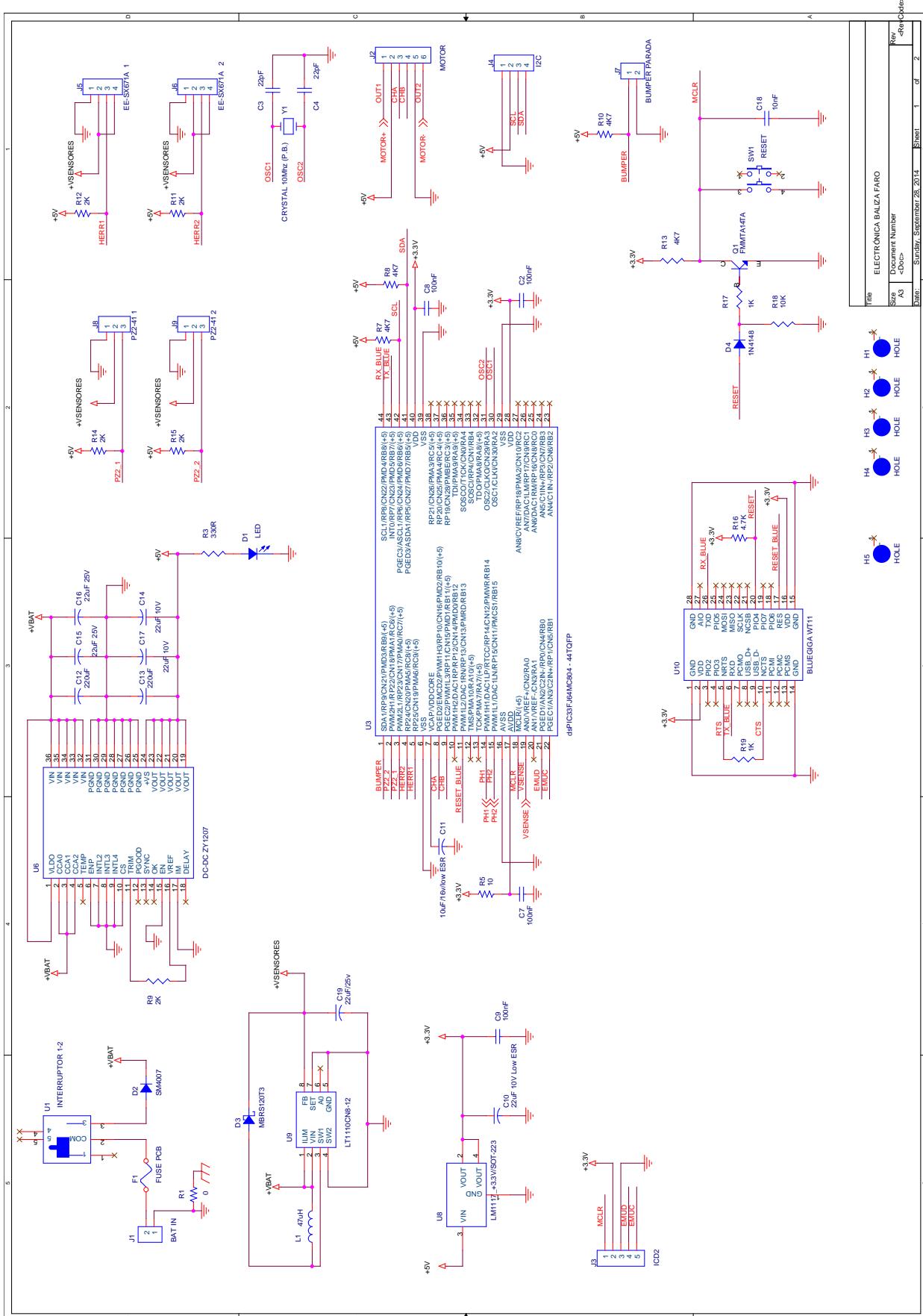


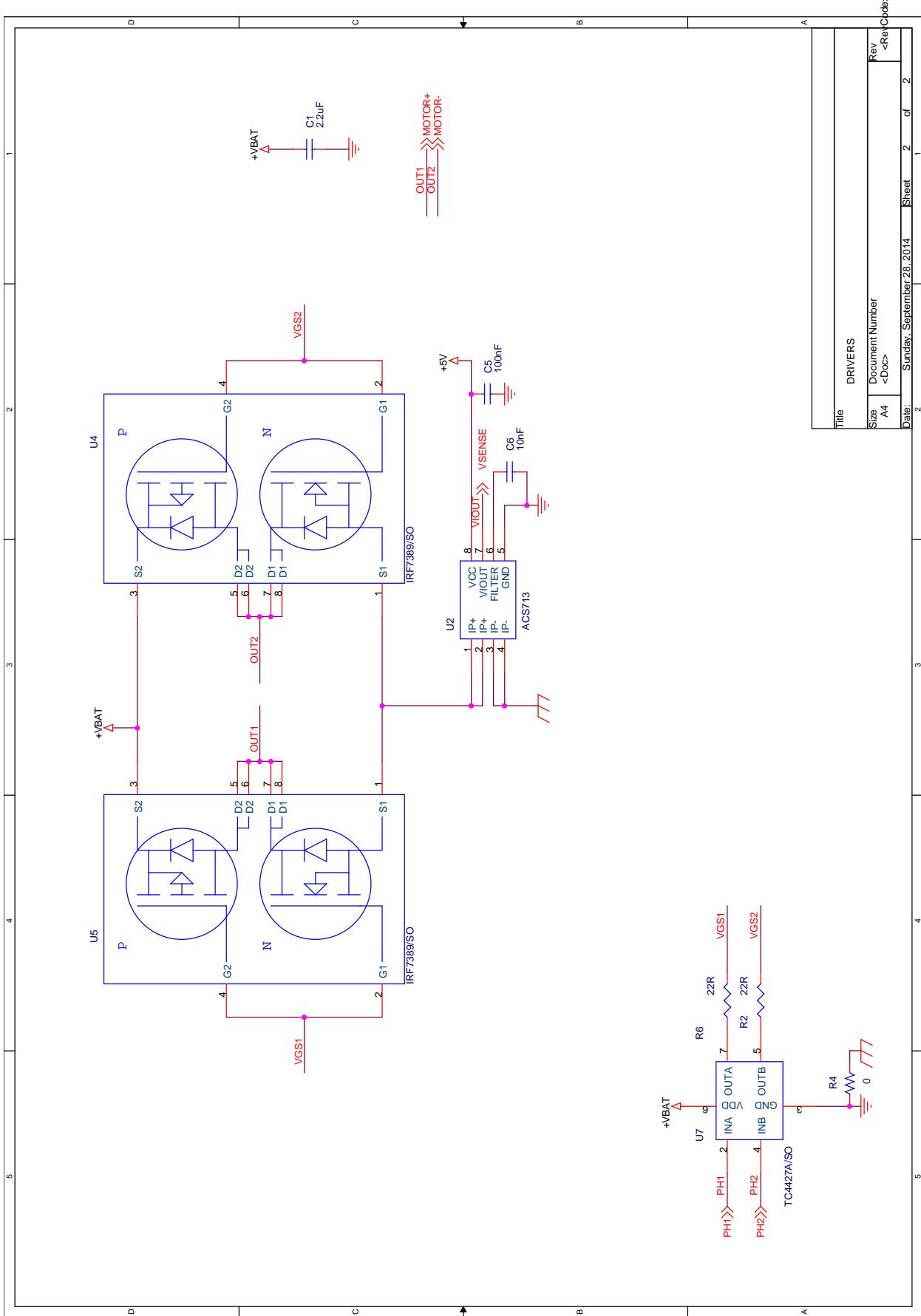


## A.6 Tarjeta *beaconboard*

Diseño electrónico y PCB: Diego Salazar Acucci



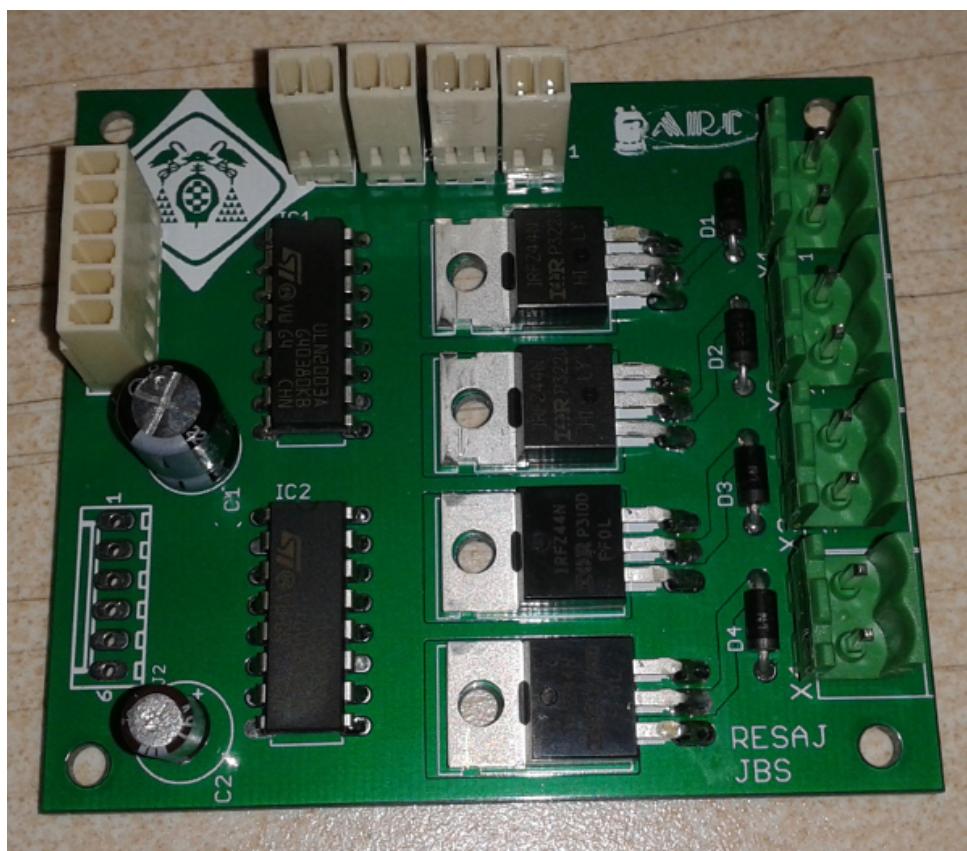


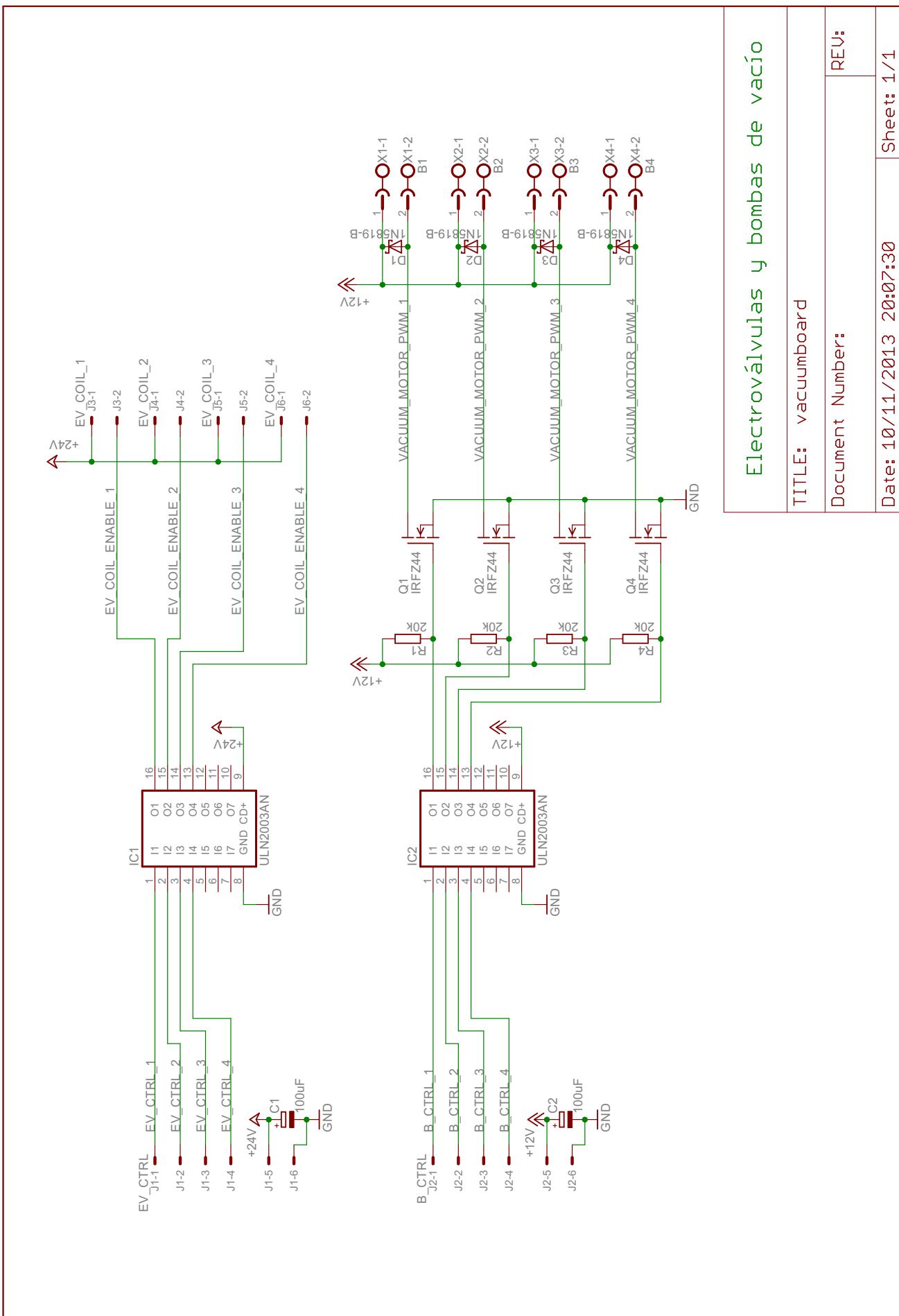


## A.7 Tarjeta *vacuumboard*

Diseño electrónico: Javier Baliñas Santos

Diseño PCB: Rubén Espino San José





## Electroválvulas y bombas de vacío

TITLE: vacuumboard

Document Number:

REV:

Date: 10/11/2013 20:07:30

Sheet: 1/1

## **Apéndice B**

# **Código fuente**

El código fuente del software de los robots desarrollados y utilizados en los ejemplos de este libro se encuentra en los repositorios GitHub del equipo Eurobotics Engineering.

Los repositorios son accesibles desde la dirección web: <https://github.com/eurobotics>.

El código fuente también se encuentra en el CD-ROM adjunto a este libro, el cual contiene una copia de los repositorios.

### **B.1 Librerías Aversive4dspic**

El código fuente correspondiente a las librerías Aversive4dspic se encuentra en el repositorio de nombre `aversive4dspic`.

Autores librerías Aversive: Christophe RIEHL and Olivier MATZ

Migración dsPIC: Javier Baliñas Santos

### **B.2 Código fuente de robots**

El repositorio del código fuente de los robots desarrollados se encuentra en los repositorios `eurobotXXXX_software` donde XXXX representa el año.

Autores: Silvia Santano Guillén, Javier Rodriguez Puigvert, Rubén Espino San José y Javier Baliñas Santos.

