



## Escuela de Ingenierías I.I.

Industrial, Informática y Aeroespacial

# GRADO EN INGENIERÍA EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Trabajo de Fin de Grado

ESTUDIO Y DESARROLLO DE UNA PLATAFORMA  
ROBÓTICA EDUCATIVA.

Autor: David Sánchez Falero

Tutor: Héctor Alaiz Moretón

## **RESUMEN**

El presente proyecto pretende abordar el diseño y desarrollo de una plataforma de robótica educativa, replicable y abierta.

En la primera parte del proyecto se plantea un estudio de la situación actual de la robótica, ilustrando su uso en el sector de la enseñanza y cuáles son los objetivos que se plantean con su uso.

En la segunda parte del proyecto se pretende abordar el desarrollo de una plataforma robótica educativa que se pueda replicar con facilidad y que sea abierta para que pueda ser modificada a las necesidades de los educadores y estudiada por parte de los alumnos.

## ÍNDICE

|   |    |
|---|----|
| CAPÍTULO 1. MARCO DEL PROYECTO.....   | 1  |
| 1.1 Estudio del uso de la robótica.....   | 1  |
| 1.2 Desarrollo de una plataforma de robótica educativa, replicable y abierta..... | 1  |
| CAPÍTULO 2. ESTADO DEL ARTE.....  | 2  |
| 2.1 Introducción .....  | 2  |
| 2.2 Robótica .....  | 2  |
| 2.3 Robótica móvil.....   | 3  |
| 2.3.1 Hardware de la robótica móvil .....   | 4  |
| 2.3.2 Sensores de la robótica móvil.....  | 7  |
| 2.3.3 Control de la robótica móvil actual .....                                   | 10 |
| 2.3.4 Robótica móvil en la sociedad .....   | 13 |
| 2.4 Robótica educativa.....   | 20 |
| 2.4.1 Robótica imprimible.....  | 21 |
| 2.4.2 Printbots como caso de éxito de la robótica imprimible.....                 | 21 |
| CAPÍTULO 3. ANÁLISIS DE REQUISITOS Y CONCEPTOS GENERALES.....                     | 24 |
| CAPÍTULO 4. SELECCIÓN DE COMPONENTES Y HERRAMIENTAS.....                          | 25 |
| 4.1 Selección de componentes. ....  | 25 |
| 4.1.1 Arduino DUE.....  | 25 |
| 4.1.2 Motor Pololu 37D mm Gearmotors 100:1.....                                   | 27 |
| 4.1.3 Dual MC33926 Motor Driver Carrier .....                                     | 28 |
| 4.1.4 Prusa i3 Hephestos .....  | 29 |
| 4.2 Entornos de desarrollo.....   | 31 |
| 4.2.1 IDE Arduino 1.5.1.r2.....   | 31 |
| 4.2.2 IDE Eclipse + Python 2.7.....   | 31 |
| 4.2.3 MATLAB R2010a .....   | 32 |
| 4.2.4 Autodesk Inventor Professional.....   | 32 |
| CAPÍTULO 5. DISEÑO MECANICO.....  | 34 |
| 5.1 Diseño de un robot de dos ruedas .....  | 34 |
| 5.2 Pasos previos al diseño mecánico.....   | 35 |
| 5.3 Diseño CAD de la plataforma. ....   | 35 |
| 5.3.1 Diseño de la base de la plataforma adaptada al proceso de fabricación.....  | 36 |

|              |  |    |
|--------------|--|----|
| 5.3.2        | Diseño de las protecciones laterales flexibles.....                      | 40 |
| 5.3.3        | Diseño de la cúpula.....   | 42 |
| 5.3.4        | Diseño de la cúpula adaptada al proceso de fabricación. ....             | 43 |
| 5.4          | Diseño final fabricado. ....   | 46 |
| CAPÍTULO 6.  | DISEÑO ELECTRÓNICO. ....   | 49 |
| 6.1          | Requisitos mínimos de diseño. ....                                       | 49 |
| 6.2          | Diseño básico de conexionado y alimentación. ....                        | 52 |
| CAPÍTULO 7.  | DISEÑO DE CONTROL. ....  | 54 |
| 7.1          | Arquitectura básica de control: control PID.....                         | 54 |
| 7.2          | Calculo y modelado de un motor de corriente continua.....                | 57 |
| 7.3          | Obtención del regulador PID para los motores de corriente continua. .... | 59 |
| 7.4          | Calculo de la posición de la plataforma en función de dos encoders. .... | 62 |
| 7.4.1        | Lectura y traducción de los encoders .....                               | 62 |
| 7.4.2        | Cálculo de la posición en función de los encoders. ....                  | 65 |
| CAPÍTULO 8.  | PRUEBAS EXPERIMENTALES. ....   | 70 |
| 8.1          | Movimiento de un metro en el eje X. ....                                 | 70 |
| 8.2          | Movimiento de un metro en los ejes X e Y .....                           | 73 |
| 8.3          | Movimiento compuesto por dos pasos de 1 metro.....                       | 76 |
| CAPÍTULO 9.  | CONCLUSIONES. ....   | 79 |
| CAPÍTULO 10. | BLIBIOGRAFÍA. ....   | 80 |

## ÍNDICE DE FIGURAS

|  |    |
|--|----|
| Figura 2-1 Configuraciones en robótica móvil.....  | 4  |
| Figura 2-2 ejemplos de configuración Skid steer.....   | 6  |
| Figura 2-3 ejemplo de configuración sincronizada.....  | 6  |
| Figura 2-4 ejemplo de configuración omnidireccional.....   | 7  |
| Figura 2-5 encoder de 3 canales. ....  | 8  |
| Figura 2-6 Nube de puntos obtenidos por una kinect.....  | 9  |
| Figura 2-7 Nube de puntos infrarrojos proyectados por una kinect .....                           | 10 |
| Figura 2-8 Ejemplo de mapa geométrico. ....  | 12 |
| Figura 2-9 Ejemplo de mapa semántico. ....   | 12 |
| Figura 2-10 Robot Nomad.....   | 13 |
| Figura 2-11 Proceso de transformación del chasis de Nomad .....                                  | 14 |
| Figura 2-12 Sample Return Rover cambiando su centro de gravedad.....                             | 14 |
| Figura 2-13 Robot de limpieza Roomba y robot de defensa y seguridad pública 510<br>PackBot. .... | 15 |
| Figura 2-14 Robot Guardian y Summit XL. Empresa Robotnik. ....                                   | 16 |
| Figura 2-15 Robot GMOD. Empresa Robotnik .....   | 16 |
| Figura 2-16 Sensores de coche autónomo de Google. ....   | 17 |
| Figura 2-17 Simulación de coche sin piloto de Google.....  | 18 |
| Figura 2-18 Grafica de desarrollo de un proyecto robótico. ....                                  | 19 |
| Figura 2-19 Complubot .....  | 20 |
| Figura 2-20 Orugator, MiniSkyBot y 4-Track .....   | 22 |
| Figura 2-21 Robot SkyBot original (con la tarjeta SkyPic) y su evolución: el MiniSkyBot ...      | 22 |
| Figura 2-22 descendencia y evoluciones del robot MiniSkyBot .....                                | 23 |
| Figura 4-1 Arduino Due .....   | 25 |
| Figura 4-2 100:1 Metal Gearmotor 37Dx57L mm con 64 CPR Encoder .....                             | 28 |
| Figura 4-3 Dual MC33926 Motor Driver Carrier .....   | 28 |
| Figura 4-4 Prusa i3 Hephestos .....  | 29 |
| Figura 4-5 Logo arduino.cc.....  | 31 |
| Figura 4-6 Logos de Eclipse Pydev y Python .....   | 31 |

|  |    |
|--|----|
| Figure 4-7 MATLAB r2010a .....   | 32 |
| Figura 4-8 Autodesk Inventor Professional .....  | 33 |
| Figura 5-1 Diseño CAD de la plataforma robotica final .....                              | 34 |
| Fihura 5-2 Diseño CAD de la base de la plataforma teorica.....                           | 35 |
| Figura 5-3 Diseño CAD de la base de la plataforma imprimible.....                        | 36 |
| Figura 5-4 Diseño CAD del soporte del motor.....   | 37 |
| Figura 5-5 Diseño CAD de la paste posterior de la base.....                              | 37 |
| Figura 5-6 Diseño CAD del apoyo frontal contra el suelo de la base. ....                 | 37 |
| Figura 5-7 Diseño CAD de la piza que ensambla todas las partes de la base.....           | 38 |
| Figura 5-8 Diseño CAD del montaje de la base imprimible. ....                            | 38 |
| Figura 5-9 Diseño CAD de la base de la plataforma con los motores y la electronica. .... | 39 |
| Figura 5-10 Diseño CAD de la protección lateral.....                                     | 40 |
| Figura 5-11 Diseño CAD de la protección frontal. ....                                    | 41 |
| Figura 5-12 Diseño CAD de la base de la plataforma con las protecciones colocadas. ....  | 41 |
| Figura 5-13 Diseño CAD de la plataforma con la cupula teórica. ....                      | 42 |
| Figura 5-14 Diseño CAD de la cúpula imprimible. ....                                     | 43 |
| Figura 5-15 Diseño CAD de la parte trasera de la cúpula. ....                            | 44 |
| Figura 5-16 Diseño CAD de la parte frontal izquierda de la cúpula.....                   | 44 |
| Figura 5-17 Diseño CAD de la parte frontal derecha de la cúpula. ....                    | 45 |
| Figura 5-18 Diseño CAD de la plataforma con la cúpula imprimible colocada. ....          | 46 |
| Figura 5-19 Imagen superior de la plataforma real. ....                                  | 47 |
| Figura 5-20 Imagen completa de la plataforma real. ....                                  | 47 |
| Figura 5-21 Imagen frontal de la plataforma real. ....                                   | 48 |
| Figura 5-22 Imagen lateral de la plataforma real. ....                                   | 48 |
| Figura 6-1 Imagen de los cables del encoder del motor. ....                              | 49 |
| Figura 6-2 Imagen del conjunto motor, adaptador y rueda. ....                            | 50 |
| Figura 6-3 Imagen de conexionado de la etapa de potencia. ....                           | 51 |
| Figura 6-4 Imagen del Arduino DUE utilizado.....   | 51 |
| Figura 6-5 Vista en detalle del encoder del motor. ....                                  | 52 |
| Figura 7-1 Estructura de un PID básico.....  | 54 |
| Figura 7-2 Estructura de un PIDcon back-calculation & tracking .....                     | 55 |

|   |    |
|---|----|
| Figura 7-3 Modelo básico para un motor de corriente continua .....                            | 57 |
| Figura 7-4 Modelado de un motor de corriente continua. ....                                   | 57 |
| Figura 7-5 modelo del motor de corriente continua. ....                                       | 60 |
| Figura 7-6 comparando una nube de puntos de la gráfica real con la del modelo calculado ..... | 60 |
| Figura 7-7 Respuesta del sistema a un regulador $K = 1$ .....                                 | 61 |
| Figura 7-8 Respuesta del sistema a un regulador $K = 0.0047$ .....                            | 61 |
| Figura 7-9 Lectura de los canales del encoder de los motores. ....                            | 63 |
| Figura 7-10 Esquema de las variables de la posición de las ruedas. ....                       | 65 |
| Figura 7-11 Esquema de las variables que intervienen en el cálculo del giro. ....             | 66 |
| Figura 7-12 Esquema de las variables que intervienen en el cálculo del giro detalladas. ..    | 67 |
| Figura 7-13 Esquema de las variables que intervienen en el incremento de posición. ....       | 68 |
| Figura 7-14 Esquema del trazado de una trayectoria punto a punto. ....                        | 69 |
| Figura 8-1 Recorrido teórico de la plataforma en la prueba 1. ....                            | 71 |
| Figura 8-2 Imagen de la plataforma al inicio de la prueba 1. ....                             | 72 |
| Figura 8-3 Imagen de la plataforma al final de la prueba 1.....                               | 72 |
| Figura 8-4 Recorrido teórico de la plataforma en la prueba 2. ....                            | 73 |
| Figura 8-5 Imagen de la plataforma al inicio de la prueba 2. ....                             | 74 |
| Figura 8-6 Imagen de la plataforma a la mitad de la prueba 2. ....                            | 74 |
| Figura 8-7 Imagen de la plataforma al final de la prueba 2.....                               | 75 |
| Figura 8-8 Recorrido teórico de la plataforma en la prueba 3. ....                            | 76 |
| Figura 8-9 Imagen de la plataforma al inicio de la prueba 3. ....                             | 77 |
| Figura 8-10 Imagen de la plataforma a la mitad de la prueba 3. ....                           | 77 |
| Figura 8-11 Imagen de la plataforma al final de la prueba 3.....                              | 78 |

## **ÍNDICE DE TABLAS**

|                 |    |
|-----------------|----|
| Tabla 7-1 ..... | 58 |
|-----------------|----|

## CAPÍTULO 1.

### MARCO DEL PROYECTO.

---

El presente proyecto pretende abordar, por una parte, el estudio del uso de la robótica en el ámbito educativo para la mejora de las capacidades cognitivas y lógicas de los niños y, por otra parte, abordar el diseño y desarrollo de una plataforma de robótica educativa, replicable y abierta.

En los apartados siguientes, se planteará una presentación más detallada de los objetivos que se pretenden alcanzar en cada una de las dos partes que componen el presente proyecto.

#### 1.1 Estudio del uso de la robótica.

Esta primera parte del proyecto pretende hacer un estudio de la situación actual de la robótica. También se estudiara su uso en el sector de la enseñanza, donde se está aplicando y cuáles han sido las causas que han llevado y los beneficios que han llevado a la implantación de este tipo de métodos educativos.

Se abordará también un estudio de cómo la robótica ayuda en la interacción con niños con problemas médicos, tanto de carácter mental como físico, y que programas se están desarrollando actualmente con éxito.

#### 1.2 Desarrollo de una plataforma de robótica educativa, replicable y abierta.

En esta segunda parte del proyecto se pretende abordar el desarrollo de una plataforma robótica educativa que se pueda replicar con facilidad y que sea abierta para que pueda ser modificada a las necesidades de los educadores y estudiada por parte de los alumnos.

Para la construcción de la estructura y carcasa de la plataforma se utilizarán técnicas de diseño e impresión 3D, para minimizar al máximo los tiempos de fabricación y facilitar el poder replicarla y modificarla con el menor coste posible.

Su diseño estará enfocado para que los niños puedan construirlo, programarlo y modificarlo tanto como crean necesario, potenciando habilidades como la creatividad y así como habilidades ingenieriles como la geometría, la visión espacial, el diseño 3D o el diseño software.

Los retos que representa esta parte del proyecto no radican en una complejidad técnica elevada, sino en la capacidad de simplificar al máximo la interfaz de la plataforma para que pueda ser utilizada por niños de corta edad y, a la vez, no comprometer la escalabilidad de la misma para que pueda ser utilizada por un niño de edad más avanzada.

## CAPÍTULO 2.

### ESTADO DEL ARTE.

---

Lo primero que se debe plantear para poder entender la motivación y el desarrollo del presente proyecto es la evolución de la robótica en la sociedad, como se está desarrollando en el ámbito educativo y como ha ayudado el desarrollo de la impresión 3D y el open source a la implementación de la misma como un medio educativo real y accesible a una gran parte del sector educativo.

También se tiene que estudiar cómo ha evolucionado el campo de la robótica móvil tecnológicamente hasta llegar al punto actual en el que cualquier persona que pretenda desarrollar una plataforma puede realizarlo sin incurrir en un gasto de fabricación excesivo o a tiempos de diseño elevados.

#### 2.1 Introducción

Este primer capítulo será un apartado previo para el entendimiento de este trabajo fin de grado sobre robótica, explicando qué es la robótica y en concreto, la robótica móvil y como esta se utiliza en entornos educativos.

#### 2.2 Robótica

En la historia, el hombre siempre ha intentado realizar máquinas y dispositivos capaces de imitar las funciones y movimientos de los seres vivos. Estas máquinas eran las que se llamaban autómatas, y como ejemplos podemos citar el León mecánico, construido por Leonardo Da Vinci en el siglo XVI, que abría su pecho mostrando el escudo del rey Luis XII, o el Hombre de palo, construido por J. Turriano para el emperador Carlos V que andaba y movía la cabeza, ojos, boca y brazos.

Durante los siglos XVII y XVIII el gremio de la relojería comenzó a construir muñecos animados, o aplicaciones prácticas como la hiladora mecánica de Crompton, o el telar mecánico de Jacquerd. A partir de este último fue cuando se empezó a utilizar dispositivos automáticos para la producción, dando paso a la automatización industrial.

La palabra robot, se comenzó a utilizar en el mundo del teatro y de los escritores del género literario de la ciencia ficción, refiriéndose al trabajo realizado de manera forzada, es decir, un trabajo físico que el hombre no desea hacer. Como ejemplo de ciencia ficción se podría mencionar al escritor Isaac Asimov por la publicación sobre las tres leyes de la robótica.

Las primeras máquinas que comenzaron a llamarse robots fueron en 1958, como telemanipuladores, con el objetivo de manipular elementos radiactivos sin riesgo para el operario. Esto era un dispositivo mecánico maestro-esclavo, donde el manipulador

maestro estaba situado en una zona segura y movido por el operario, unido mecánicamente al esclavo que realizaba fielmente los movimientos de éste.

Años más tarde, se comenzó a hacer uso de la tecnología electrónica y el servocontrol, sustituyendo la transmisión mecánica por otra eléctrica. Ejemplos de estos telemanipuladores, fue la de R. Mosher, quien diseño un dispositivo denominado Handy-Man que consiste en dos brazos mecánicos teleoperados mediante un maestro del tipo exoesqueleto. Junto a la industria nuclear, se comenzó a interesar la industria submarina y la espacial.

La sustitución del operador por un programa de ordenador que controlare los movimientos del manipulador dio paso al más puro concepto de robot, comenzando con una serie de robots industriales de todas las áreas.

### 2.3 Robótica móvil

La robótica móvil es un tipo de robots de servicio que nos permite explorar el entorno donde un humano no puede llegar y brindan la posibilidad de navegar en distintos terrenos.

Las aplicaciones pueden ser múltiples, exploración minera, exploración planetaria, misiones de búsqueda y rescate de personas, limpieza de desechos peligrosos, automatización de procesos, vigilancia, reconocimiento de terreno y usados como plataformas móviles que incorporan un brazo manipulador.

Las funciones son por tanto, tareas que un humano no puede hacer, o bien por la lejanía, la peligrosidad de la tarea, el coste o en tareas que sean repetitivas.

El primer robot móvil que se desarrolló fue en SRI (Standford Research Institute) de 1966 a 1972, un robot llamado Shakey, que consistía en una plataforma móvil independiente controlada por visión mediante una cámara y dotada de un detector táctil. A partir de ese momento, la investigación y diseño de robots móviles creció de manera exponencial.

Los robots móviles están compuestos, como todos los robots, por una estructura mecánica, sensores, actuadores, fuentes de alimentación y un controlador, que llevará a cabo toda la lógica, desde recibir las señales de un controlador externo hasta convertir esta señal en una señal de actuación, que mas tarde se enviará a una controladora de motores y actuará en consecuencia.

El control reactivo asocia la información sensorial con la acción en el comportamiento de la respuesta motora, para producir respuestas puntuales en un entorno dinámico y poco estructurado.

La actividad sensorial tiene que proporcionar la información necesaria para satisfacer una respuesta a bajo nivel de un motor.

La robótica móvil actualmente está desarrollándose para multitud de aplicaciones, y diversos entornos, tanto en espacios abiertos y desconocidos (Como puede ser un Rover en la robótica espacial) o en espacios cerrados conocidos o no conocidos (El robot tiene

que ir de una habitación a otra para alguna tarea, como por ejemplo una silla de ruedas autónoma).

Del mismo modo, el hardware del robot y el control mediante software es muy diferente dependiendo de la aplicación. Este capítulo trata de hacer un resumen del estado actual del desarrollo en robótica móvil.

### 2.3.1 Hardware de la robótica móvil

En la robótica móvil el primer paso es elegir el tipo de actuador que se va a utilizar, entendiéndose como actuador el que genera el movimiento de los elementos del robo según las órdenes dadas por la unidad de control. En la robótica móvil, este actuador será normalmente un actuador eléctrico, eligiendo el más conveniente según la potencia, controlabilidad, precisión, velocidad, coste...

Los actuadores eléctricos, y en concreto, los motores de corriente continua, son los más utilizados debido a su facilidad de control y su fácil acoplamiento a un encoder. Los encoders se emplean como sensor de posición del robot, dando la posibilidad de conocerla de forma relativa, como ocurre con los encoders incrementales, o de forma absoluta. Básicamente son discos que se acoplan al eje del motor y tienen una serie de marcas donde dejan pasar una luz emisora y un receptor contará los pulsos cada vez que la luz atraviese la marca.

Los motores paso a paso apenas se utilizan en la robótica móvil debido a la velocidad lenta a la que se mueven. A cambio, se obtiene un par mucho mayor que en los motores de corriente continua y sería posible su control en cadena abierta, es decir, sin necesidad de obtener la información de la posición y corregir su velocidad (Realimentación), ya que su posicionamiento es exacto, por lo que se podría utilizar para un terminal del robot en el que no se necesitara excesiva precisión y sin grandes potencias, como una pinza en el extremo de un brazo robótico.

El primer paso que se da en la robótica móvil es elegir el tipo de configuración que se desea para el robot. Es decir, como están dispuestos los distintos actuadores del los que se compone, existiendo seis configuraciones: Ackerman, triciclo clásico, tracción diferencial, skid-steer, síncrona y tracción omnidireccional.

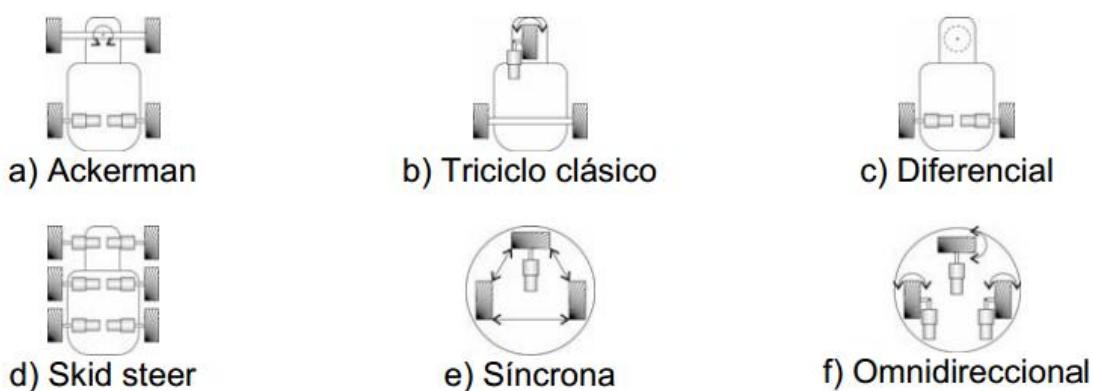


Figura 2-1 Configuraciones en robótica móvil

### **2.3.1.1 Configuración Ackerman.**

Es la usada en la industria del automóvil. Posee dos ruedas traseras de tracción y dos ruedas delanteras para la dirección. Esta configuración esta creada para evitar el derrape de las ruedas, lo que se consigue haciendo que la rueda delantera interior posea un ángulo ligeramente mayor que el ángulo de la rueda exterior.

### **2.3.1.2 Triciclo clásico.**

Consta de tres ruedas, dos pasivas que no se acoplan a ningún motor (que sirven de soporte) y la dirección y tracción es proporcionada por la delantera. En este caso, no se pueden realizar giros de 90º.

### **2.3.1.3 Configuración diferencial.**

Consta de dos ruedas colocadas en el eje perpendicular a la dirección del robot. Cada rueda es controlada por un motor, de tal forma que el giro del robot queda determinado por la diferencia de velocidad de las ruedas.

### **2.3.1.4 Skid steer.**

Esta configuración funciona igual que la configuración diferencial, con la diferencia de que tiene más ruedas a cada lado. Esta configuración se suele encontrar sobre todo en aplicaciones relacionadas con la exploración, vehículos de minería, montacargas...

Esta configuración tiene el beneficio de la configuración diferencial pero tiene el inconveniente de que para lograr un cambio en la dirección del vehículo, las ruedas deben deslizarse lateralmente, por tanto será muy complicado realizar un modelo cinemático y un control dinámico.



Figura 2-2 ejemplos de configuración Skid steer

#### **2.3.1.5 Configuración sincronizada.**

Conformado por tres o más ruedas, acopladas mecánicamente y dotadas de tracción, de tal forma que todas rotan en la misma dirección y a la misma velocidad.



Figura 2-3 ejemplo de configuración sincronizada.

### 2.3.1.6 Configuración omnidireccional.

Este tipo de configuración está provista de ruedas omnidireccionales, lo que hace que el robot pueda moverse en cualquier dirección. La rueda omnidireccional es una rueda estándar a la que se ha dotado de una corona de rodillos. Los ejes de giro de los rodillos son perpendiculares a la dirección normal de avance.

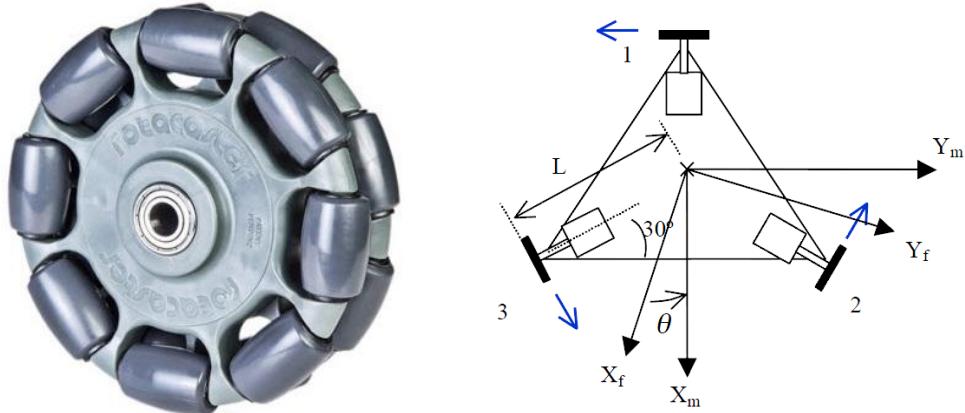


Figura 2-4 ejemplo de configuración omnidireccional.

El modelo matemático de cada configuración no nos interesa en este proyecto, ya que suponemos que el robot no es ideal y que el modelo cinemático y dinámico de una estructura skid steer como es la que usamos, es muy difícil de conseguir y actuaremos en consecuencia de ello, pero si nos interesa, podemos consultar la bibliografía.

### 2.3.2 Sensores de la robótica móvil

Para conseguir que el robot realice la tarea con una determinada precisión y velocidad debe conocer el entorno del sistema en el que se quiera actuar y corregir si se diese el caso. Existen dos grupos de sensores, los sensores internos, relacionada con el estado del robot, y los externos, relacionada con el entorno de su alrededor.

De los sensores internos, los sensores de posición básicos son los encoders, tanto los incrementales como los absolutos basados en un código Gray, es decir, las marcas del disco donde pasa la luz emisor está basada en el código Gray de manera que no es necesario ningún contador para detectar el sentido de giro.

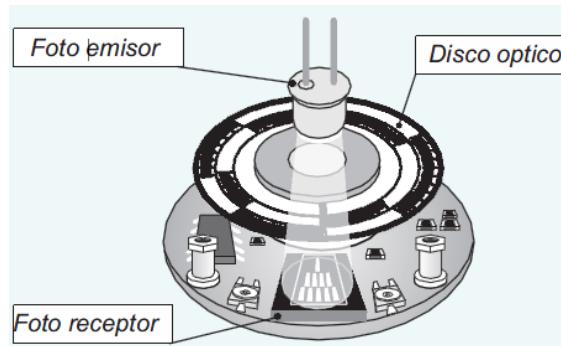


Figura 2-5 encoder de 3 canales.

Los sensores de velocidad serán necesarios en el caso de que se quiera mejorar el comportamiento dinámico del robot, y se usa un tacogeneratriz que proporciona una tensión proporcional a la velocidad de giro en su eje.

Los sensores externos pueden medir una variedad muy amplia de magnitudes físicas, como la temperatura, humedad, sensores de presión, magnetismo, gravedad, proximidad...

Se analizará los sensores que interesa para la robótica móvil, como son los sensores de presencia mecánicos, o fin de carrera, donde un dispositivo se coloca al final de la trayectoria del movimiento e indica en qué momento llega. El sensor de presencia puede ser óptico, un emisor emite un rayo infrarrojo y si existiera un objeto cercano este haría reflejar la señal para que el receptor obtenga la distancia, o bien el receptor está en el objeto como en el caso de las barreras láser. Los sensores inductivos y capacitivos funcionan de distinta forma pero obtienen de igual manera la presencia cercana de objetos.

Sensores que miden la aceleración, acelerómetros, como un sensor para medir los cambios en los ejes del robot. Estos sensores, junto con un giroscopio en los tres ejes, sería posible realizar un control de equilibrio del robot en cualquier plano. En el caso de la robótica móvil, normalmente no es necesario ya que el robot camina sobre una superficie plana o casi plana.

Sensores para posicionamiento global GPS (*Global Positioning System*) que permiten determinar en todo el mundo la posición de objetos, normalmente con una precisión de metros. El GPS funciona con una red de satélites con trayectorias sincronizadas para cubrir toda la superficie de la tierra. El GPS lanza señales a los satélites y calculando el tiempo que tarda en responder los satélites, calcula la posición por triangulación.

Para los sensores de distancia se pueden utilizar sensores de ultrasonido, donde un emisor emite una onda de radio y cuando se refleja en un objeto puede determinar la distancia a la que está con el tiempo que tarda el sonido en ir y volver. Los sensores de

distancia también pueden ser infrarrojos, funcionando de igual manera que en los sensores de presencia.

Existen sensores de distancia industriales, como es el caso de los sensores de medición de la marca Sick, que funcionan emitiendo rayos infrarrojos en prácticamente la circunferencia completa.

Los sensores de distancia no tienen por qué ser sólo en dos dimensiones, existiendo sensores tridimensionales, siendo uno de ellos el sensor de Kinect, que ha crecido recientemente dado su bajo coste y su comercialización.

**Kinect** es un dispositivo desarrollado por PrimeSense y distribuido por Microsoft para la videoconsola Xbox 360 y desde junio del 2011, para PC a través de Windows 7 y Windows 8.

Inicialmente permitía controlar e interactuar con la consola XBOX sin necesidad de tener un contacto físico con un controlador de videojuegos mediante una interfaz que reconoce gestos, comandos de voz y objetos e imágenes.

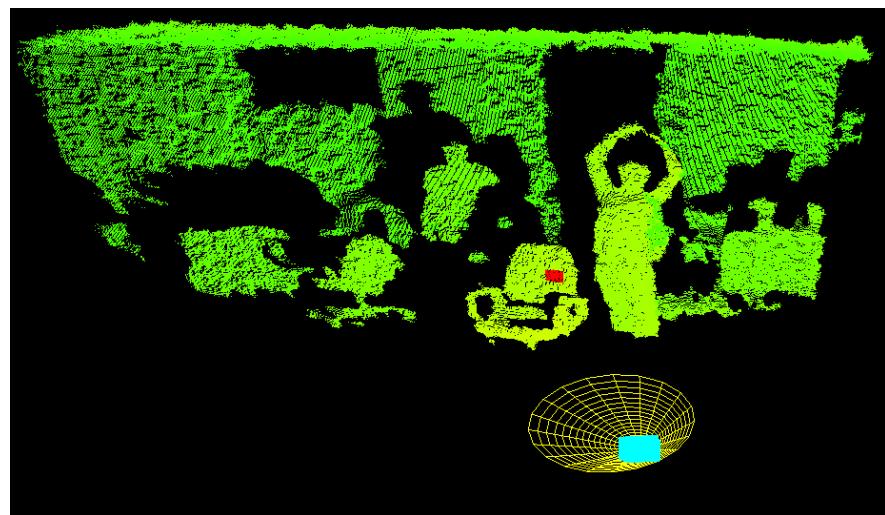


Figura 2-6 Nube de puntos obtenidos por una kinect

Para que todo esto sea posible, Kinect incluye una cámara de vídeo RGB, una cámara infrarroja de profundidad, un array de micrófonos y altavoces, un acelerómetro y un pequeño motor que le permite hacer movimientos de inclinación, de lo que para el ámbito de la robótica móvil, interesa en especial el emisor de infrarrojos y la cámara infrarroja.



Figura 2-7 Nube de puntos infrarrojos proyectados por una kinect

En la última imagen podemos apreciar los puntos de infrarrojo que emite el sensor de Kinect y como lo hace en todas direcciones, al mismo tiempo que podemos observar que no lo hace de forma completamente uniforme, pero sí podremos obtener la posición en coordenadas cartesianas XYZ de cada punto. Se explicará cómo funciona detenidamente en el apartado de Hardware.

### 2.3.3 Control de la robótica móvil actual

El control del movimiento de los robots móviles con ruedas, a grosso modo, se puede clasificar en cuatro tareas fundamentales; localización, planificación de trayectoria, seguimiento de la misma y evasión de obstáculos.

En relación puntual al tópico de seguimiento de trayectoria, existen diferentes métodos para lograr esta tarea, siendo de los más empleados el de persecución pura (*pure pursuit*) éste genera arcos entre el punto de desplazamiento del móvil y los puntos de la trayectoria a seguir, los arcos se generan de 10 a 15 veces por segundo lo que resulta en un seguimiento suave y con muy buenos resultados. Otro método para el seguimiento de trayectoria es conocido como ajuste de polinomios de quinto orden (*Quintic Polynomial Fit*). Otra técnica recientemente empleada es mediante linealización de entrada-salida la cual impone que las variables de estado tiendan asintóticamente a la trayectoria deseada, generando una dinámica remanente asociada a una variable de estado, la cual resulta ser estable.

En lo que respecta a la evasión de obstáculos, existen distintos métodos para llevar a cabo esta tarea. Los más relevantes son: por detección de bordes, por descomposición en celdas, por construcción de mapas y por campos potenciales artificiales.

En el método por detección de bordes, el algoritmo implementado, permite que el robot detecte los bordes verticales de un posible obstáculo; sin embargo, dependiendo de los sensores que hagan esta detección, es necesario que el robot se detenga cuando detecte dicho borde y compute la información necesaria para confirmar la presencia del obstáculo.

El método por descomposición en celdas, divide en celdas el espacio de trabajo del robot en un plano bidimensional, a cada celda se le asigna un valor que permite saber si existe dentro de la misma algún obstáculo. La desventaja de este algoritmo es el tiempo computacional requerido para dividir el espacio de trabajo en celdas, aunado al requerimiento excesivo de sensores para brindar una considerable precisión al dividir el espacio de trabajo.

En el método por construcción de mapas, el algoritmo implementado permite crear un grafo que conecta cada punto del espacio libre de trabajo, facilitando la generación de un camino que permita al robot navegar desde su punto inicial a su punto final eliminando la incertidumbre de posibles obstáculos.

Finalmente, respecto al método de campos potenciales artificiales, la mayor parte de la literatura sobre evasión de obstáculos es referida a este. Fue desarrollado por Khatib, y se supone al móvil como una partícula, de igual forma los obstáculos que lo rodean se consideran como partículas que ejercen una fuerza de repulsión sobre el móvil, mientras que, su punto de arribo, es una fuerza atractiva, consigiéndose de esta forma establecer un campo potencial que representa el ambiente en el que debe de seguir su trayectoria el robot. Esta trayectoria es generada a partir del mismo método, donde los obstáculos pueden previamente considerarse en el algoritmo o bien detectarse mediante algún tipo de sensor.

Normalmente uno de los mayores problemas que conciernen a la navegación de robots móviles consiste en la determinación de su localización con un determinado grado de precisión. No basta con situar una referencia global, si no que es imprescindible conocer la posición relativa respecto a posibles obstáculos, tanto móviles como estáticos, de su entorno. Para ello existen diferentes alternativas, como utilizando mapas que se introducen a priori en el robot, o bien se elaboran a medida que se mueve por un lugar, como si fuese un robot de exploración. Es lo que se conoce como SLAM (*Simultaneos Localization and Mapping*).

El origen de tener que usar SLAM para la localización y construcción de mapas es debido al ruido de los sensores de posición del robot, en concreto, con la odometría. Pero el uso

de esta herramienta no es siempre positivo, ya que si se utilizara un mapeo del mapa en el momento en el que se mueve sería necesario un alto coste computacional para guardar todos los puntos del mapa, y este mapa no tiene porque ser estático siempre, ya que pueda haber variaciones en él, como por ejemplo cuando se mapea un espacio cerrado, y una persona esté paseando continuamente por él.

Se pueden distinguir tres tipos de mapas: Geométrico, topológico y semántico. El nivel geométrico es el más utilizado y consiste en mapa métrico donde se representen los segmentos básicos de un entorno, o un mapa métrico discretizado, donde se efectúa una descomposición en celdillas. En el nivel topológico, se representarán nodos y conexiones entre ellos, y el nivel semántico es cuando se elimina la información geométrica.

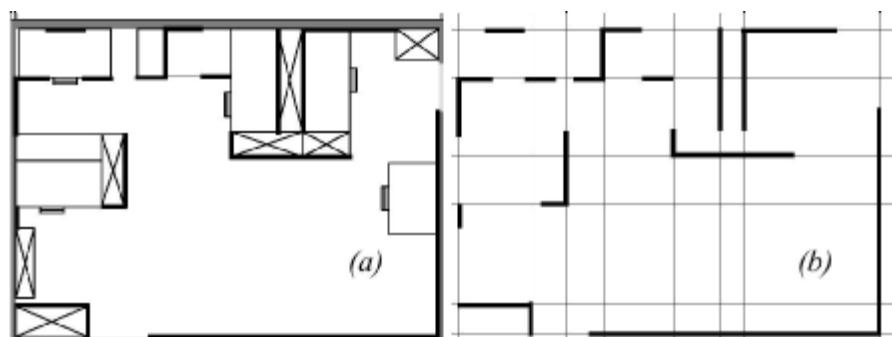


Figura 2-8 Ejemplo de mapa geométrico.

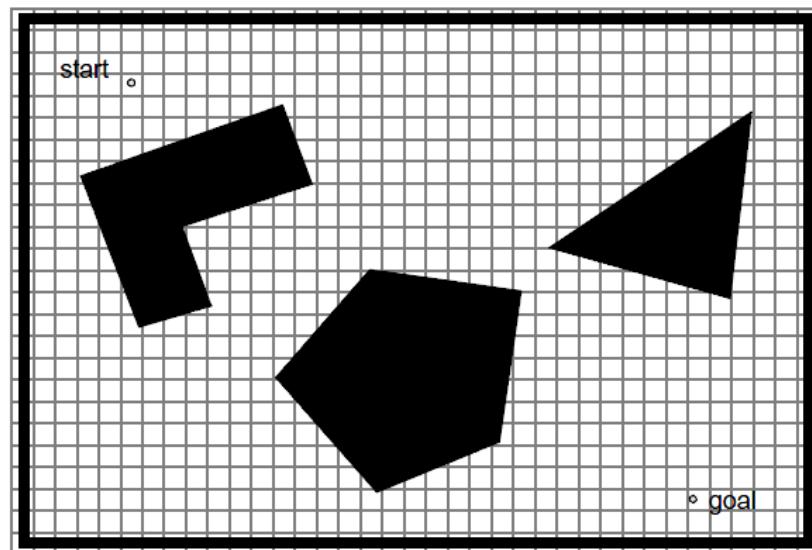


Figura 2-9 Ejemplo de mapa semántico.

### 2.3.4 Robótica móvil en la sociedad

Anteriormente se ha mencionado algunas de las posibles aplicaciones de la robótica móvil en las que un humano no puede acceder por peligrosidad por ejemplo, o tareas repetitivas.

Una de las aplicaciones más famosas de la robótica móvil es los robots de exploración espacial, como Nomad creado por el Instituto de Robótica de la Universidad Carnegie Mellon para la NASA que posteriormente fue adaptado para buscar meteoritos en la Antártida.



Figura 2-10 Robot Nomad

Nomad fue una revolución ya que podía modificar su chasis y configuración de avance según sea el terreno.

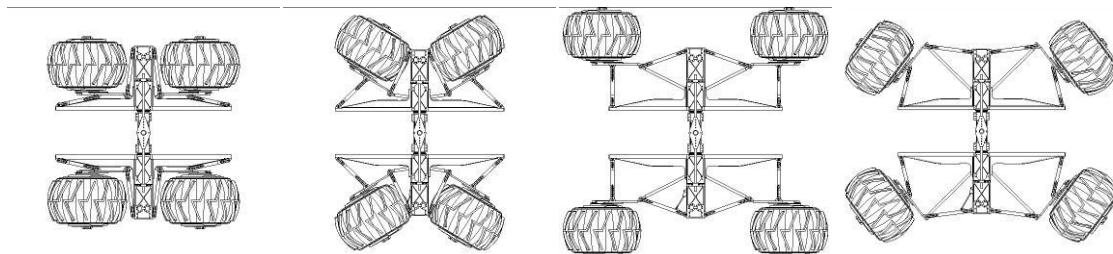


Figura 2-11 Proceso de transformación del chasis de Nomad

El robot Sample Return Rover (SSR) es un robot de cuatro ruedas desarrollado por el Jet Propulsion Laboratory (JPL) en colaboración con el Massachusetts Institute of Technology (MIT). El robot tiene como gran aportación la de modificar su configuración para reposicionar el centro de gravedad según la situación lo requiera.



Figura 2-12 Sample Return Rover cambiando su centro de gravedad.

La empresa iRobot ha desarrollado un robot capaz de ocuparse de la tarea repetitiva de la limpieza del hogar, o un robot de servicio para búsqueda, reconocimiento y desactivación de explosivos (Figura de la derecha)



Figura 2-13 Robot de limpieza Roomba y robot de defensa y seguridad pública 510 PackBot.

Otra empresa que se dedica al desarrollo de I+D en el entorno de la robótica es Robotnik [RBK02]. Esta empresa, se centra en robots autónomos de transporte interior y aplicaciones del área de servicio, productos de robótica como robots móviles, brazos robot, humanoides y robots realizados a medida.

Los robots se sirven de la arquitectura software de ROS beneficiándose así de las ventajas de código abierto.

Algunos de los ejemplos son el robot Guardian (Figura de la izquierda) cuyas aplicaciones pueden ser muy variadas, como la navegación en interiores y exteriores, localización, robots SWARM, vigilancia, medición remota y búsqueda y desactivación de explosivos. El robot Summit XL tiene una cinemática diferencial al igual que el robot de Pioneer 3AT basada en 4 motores de alto rendimiento sin escobillas, por lo que puede ir mucho más rápido y gracias a su estructura, puede soportar cargas más pesadas, poseyendo cámaras para la teleoperación o navegando autónomamente gracias al estándar para acoplarle sensores infrarrojos (Figura de la derecha)



Figura 2-14 Robot Guardian y Summit XL. Empresa Robotnik.

El robot GMOD de la misma empresa, es un robot manipulador móvil, que permite sumar las ventajas de la plataforma móvil y el brazo robótico. De este modo, el manipulador móvil permite que el brazo interactúe en espacios a los que no podría acceder por sí mismo, al mismo tiempo que la plataforma móvil adquiere nuevas posibilidades de manipulación, como el manejo de objetos o la elevación de cargas.



Figura 2-15 Robot GMOD. Empresa Robotnik

Una aplicación de la robótica móvil que más ha llamado la atención ha sido los coches sin conductor, aplicación potenciada por los desarrollos de Google, cuyo proyecto consiste en combinar la información obtenida del servicio de mapas de Street View con inteligencia artificial, donde se le envía al control información obtenida con sensores de cámaras de video, un sensor láser de 360º encima del vehículo, sensores radar y de posición con localización GPS.

## Autonomous Driving

Google's modified Toyota Prius uses an array of sensors to navigate public roads without a human driver. Other components, not shown, include a GPS receiver and an inertial motion sensor.

### LIDAR

A rotating sensor on the roof scans more than 200 feet in all directions to generate a precise three-dimensional map of the car's surroundings.

### VIDEO CAMERA

A camera mounted near the rear-view mirror detects traffic lights and helps the car's onboard computers recognize moving obstacles like pedestrians and bicyclists.



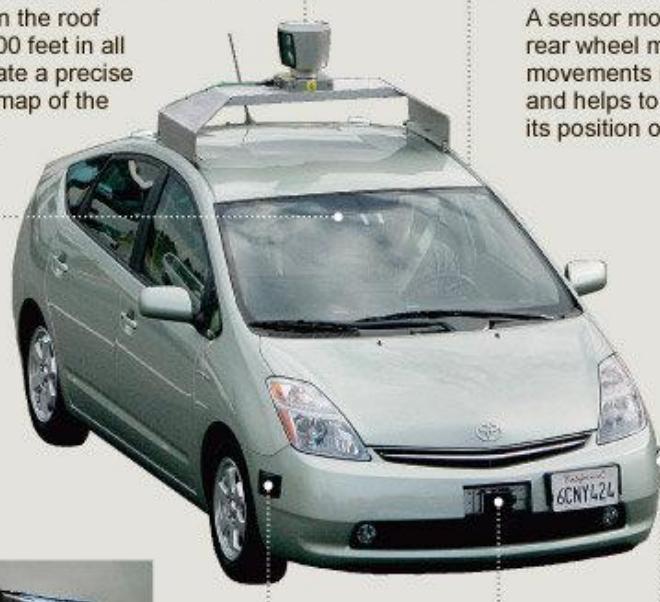
### POSITION ESTIMATOR

A sensor mounted on the left rear wheel measures small movements made by the car and helps to accurately locate its position on the map.



### RADAR

Four standard automotive radar sensors, three in front and one in the rear, help determine the positions of distant objects.



Source: Google

THE NEW YORK TIMES; PHOTOGRAPHS BY RAMIN RAHIMIAN FOR THE NEW YORK TIMES

Figura 2-16 Sensores de coche autónomo de Google.

Los creadores de este software son algunos de los ingenieros que desarrollaron otros vehículos en los desafíos de DARPA, carreras de vehículos autónomos organizadas por el gobierno de los EEUU.

A partir de marzo de 2012 entró en vigor una ley en Nevada relativa a los vehículos sin conductor, autorizando a Google a hacer sus pruebas experimentales.

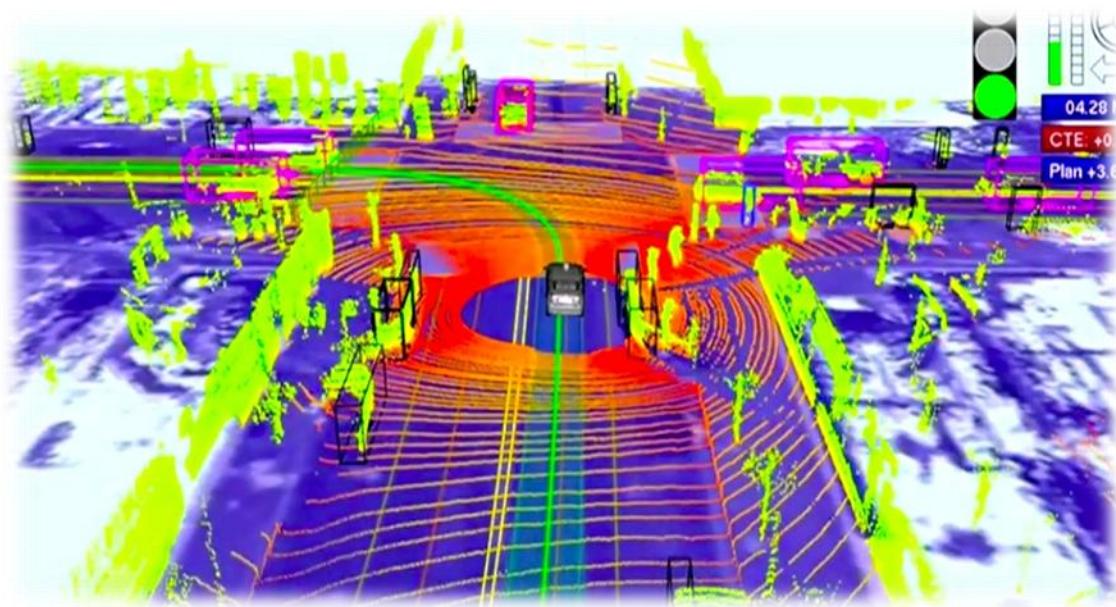


Figura 2-17 Simulación de coche sin piloto de Google.

Sin embargo, el mercado de la robótica móvil para uso civil es aún muy pobre, y su tasa de crecimiento lenta. Sin embargo, las expectativas para los próximos años son muy favorables, como en el caso de muchos robots aspiradoras autónomos, como el robot Roomba de la empresa iRobot, y muchos robots de ese tipo que empiezan a ser comerciales, además de este último caso de coches autónomos, que se espera grandes avances próximamente.

La predisposición del público a adquirir este tipo de dispositivos es, en general, buena, gracias a la imagen que se le ha dado del mundo cinematográfico.

La clave para conseguir posicionarse en un mercado de estas características suele radicar en ser el primero en ofrecer un producto exclusivo orientado a un perfil de un cliente determinado y conseguir mantener esa ventaja a lo largo del tiempo. Todo producto experimenta cuatro etapas diferenciadas durante su vida, introducción, crecimiento, madurez y declive.

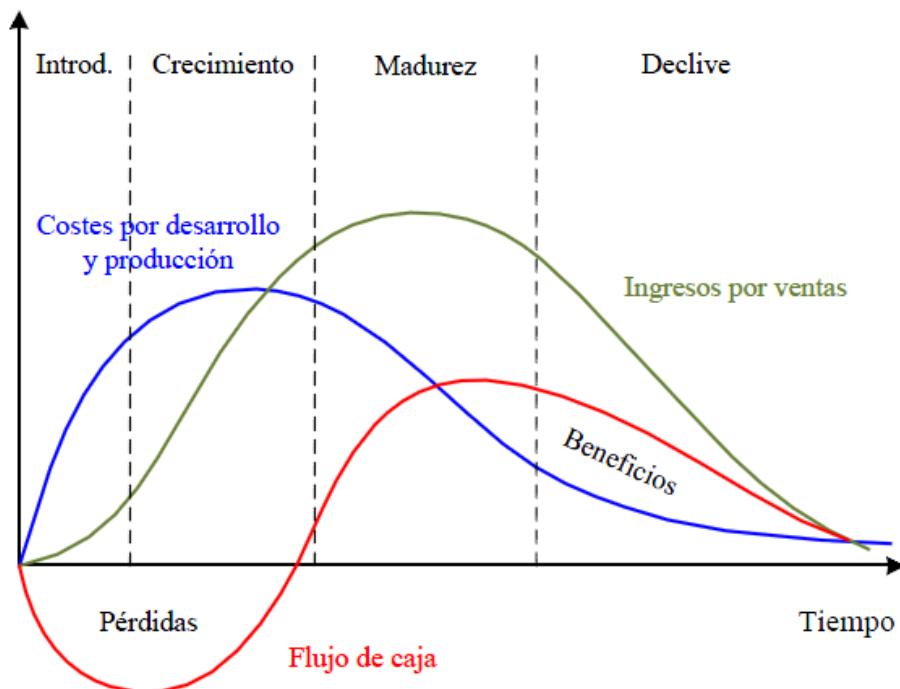


Figura 2-18 Grafica de desarrollo de un proyecto robótico.

Durante la primera etapa, las ventas no son suficientes para cubrir los gastos de desarrollo, por lo que se producirán pérdidas. Una vez dado a conocer el producto en el mercado, la demanda comenzará a crecer por lo que será necesario invertir en la mejora de los procesos productivos con el fin de poder satisfacerla eficientemente, y debido a la experiencia, las pérdidas por los costes de producción se irán reduciendo.

Cuando se consolida en el mercado, se estabilizarán las ventas en la etapa de madurez, el cual comenzará a reportar mayor beneficio ya que los costes de producción seguirán disminuyendo. Finalmente, llegará el declive del producto por su obsolescencia.

En conclusión, la robótica móvil es un sector con expectativas de futuro. Para conseguir que una empresa tenga éxito, tendrá que hacer un estudio de la demanda de la sociedad y comenzar el desarrollo de prototipos. La robótica es muy versátil, por lo que se podrá adaptar a un amplio perfil de clientes.

## 2.4 Robótica educativa.

La Robótica educativa es definida como una disciplina que permite concebir, diseñar y desarrollar robots educativos para que los estudiantes se inicien desde muy jóvenes en el estudio de las ciencias y la tecnología.

Surge con la finalidad de explotar el deseo de los educadores por interactuar con un robot para favorecer los procesos cognitivos. También se define esta disciplina como “la actividad de concepción, creación y puesta en funcionamiento, con fines didácticos, de objetos tecnológicos, que son reproducciones reducidas muy fieles y significativas de los procesos y herramientas robóticas que son usadas cotidianamente, sobre todo, y que cada vez son más comunes en nuestro entorno social, productivo y cultural”.

Numerosas investigaciones ya demuestran el interés global por la inserción de herramientas robóticas en las aulas de clase. Desde el año 1975, en la Universidad Du Maine, en Le Mans, Francia, aparece una primera utilización con fines educativos de la robótica, con el desarrollo de un sistema de control automatizado para la administración de experiencias en laboratorio, para prácticas de psicología experimental (Nonnon et Laurencenlle, 1984).

En 1998 se inició el proyecto “Robótica y Aprendizaje por Diseño”, realizado conjuntamente por el Centro de Innovación Educativa de la Fundación Omar Dengo y el Ministerio de Educación Pública de Costa Rica (Fundación Omar Dengo, 2004).

En España, redes educativas como COMPUBLOT (2008) implementan aulas de robótica y cursos de formación para niños en nivel de formación primaria.



Figura 2-19 Complubot

Para verificar los objetivos de la robótica educativa como disciplina integradora de distintas áreas del conocimiento es necesario el desarrollo de dos procesos individuales, pero altamente dependientes.

Por una parte, se deben establecer funciones desde el punto de vista de ingeniería para el estudio y proceso de concebir, diseñar y construir mecanismos robóticos; y una segunda función, desde el punto de vista didáctico, para constatar que efectivamente dichos mecanismos cumplen los fines educativos para los cuales fueron desarrollados, lo que involucra investigaciones en las disciplinas del conocimiento de la educación, enseñanza y aprendizaje.

Las seis principales áreas de trabajo que se han propuesto en la robótica pedagógica son las siguientes.

- ✓ Apoyo en la enseñanza de primaria y secundaria.
- ✓ Adultos en formación profesional.
- ✓ La robótica aplicada a las personas discapacitadas.
- ✓ La robótica como herramienta de laboratorio.
- ✓ La robótica pedagógica para facilitar el desarrollo de los procesos cognitivos y de representación.
- ✓ Análisis y reflexiones sobre la Robótica Educativa y sus aplicaciones.

Como apoyo a la enseñanza de primaria y secundaria, se han conseguido considerables aportes en el aprendizaje de conceptos principalmente relacionados con las matemáticas, las ciencias y la programación, utilizando herramientas que resulten interesantes para los alumnos y que faciliten sus procesos de aprendizaje. La aplicación de esta disciplina pretende explotar lo atractivo que resulta para los educandos la idea de aprender jugando.

#### **2.4.1 Robótica imprimible.**

La robótica educativa requiere plataformas que sean sencillas de utilizar, siendo a la vez muy completas y manteniendo un precio reducido, pero en muchas ocasiones es necesario dejar de lado al menos uno de estos aspectos.

Los robots imprimibles han supuesto un gran cambio, ahora es posible desarrollar robots bajo demanda, sin necesidad de recurrir a caros equipos comerciales.

Pero los robots libres e imprimibles suponen además una revolución en el paradigma de la robótica, comienzan a formarse proyectos de hardware abierto que evolucionan gracias a la comunidad. Conforme más personas trabajan con estos diseños, van apareciendo mejoras en ocasiones muy ingeniosas, procedentes de cualquier parte del mundo.

En cuanto a aspectos didácticos, fabricar un robot imprimible permite desarrollar competencias tan importantes como el diseño mecánico y electrónico, procesos de fabricación o programación.

Para ilustrar este nuevo paradigma educativo, podemos estudiar el caso de los Prinbots.

#### **2.4.2 Printbots como caso de éxito de la robótica imprimible.**

Desde que surgieron las primeras impresoras 3D y fueron llegando a los centros de investigación, se han usado para diversos objetivos, y entre ellos está muchas veces el fabricar piezas para robots.

Según el tipo de impresión 3D utilizada, estas piezas pueden ser simples pruebas de concepto o bien partes definitivas listas para el ensamblado.

El problema con estos robots es que sus planos son, en general, privativos, ya que se encuentran restringidos al laboratorio, centro o empresa que ha desarrollado sus piezas.

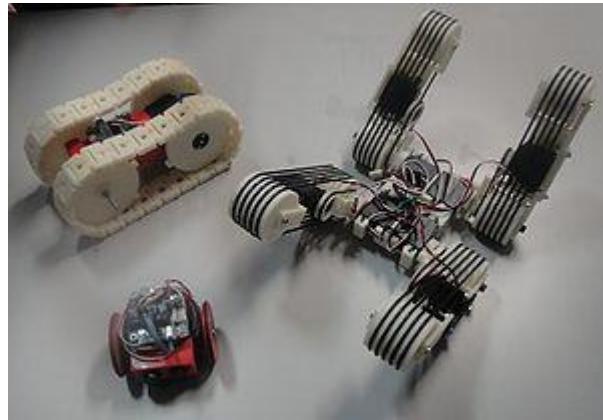


Figura 2-20 Orugator, MiniSkyBot y 4-Track

#### 2.4.2.1 MiniSkybot

El primer Printbot fue el robot MiniSkyBot 1.0, terminado en Enero de 2011, que es descendiente del SkyBot original. El diseño mecánico del SkyBot fue realizado por Andrés Prieto Moreno. La electrónica del SkyBot, al igual que el MiniSkyBot completo, por Juan González Gómez.

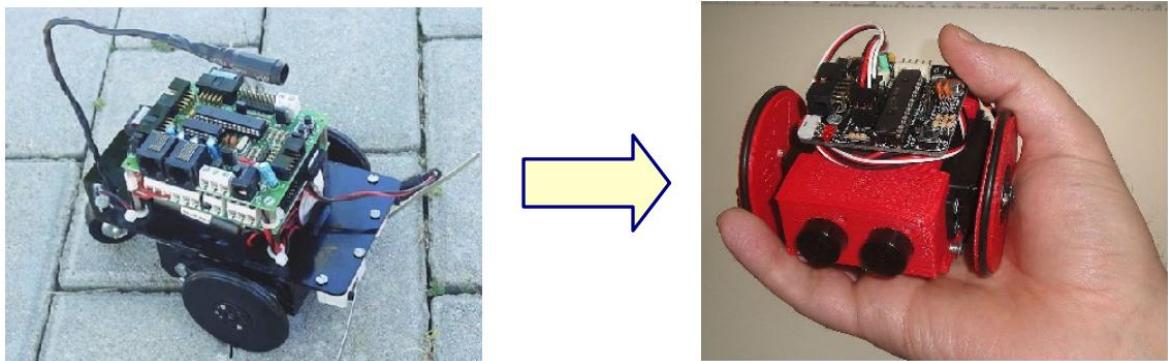


Figura 2-21 Robot SkyBot original (con la tarjeta SkyPic) y su evolución: el MiniSkyBot

El MiniSkybot tiene un chasis completamente imprimible, utiliza la electrónica de control SkyMega (que es compatible con el IDE Arduino), servos trucados para la rotación continua como motores, y dispone de sensores de distancia por ultrasonidos.

#### 2.4.2.2 Evoluciones del MiniSkybot

Poco después, han ido surgiendo nuevos diseños, creando un subconjunto de robots adaptados a las necesidades de cada usuario.

Esta diversidad ha sido posible gracias a la posibilidad de acceder a los diseños y poder modificarlos, dotando de una agilidad a los estudiantes que antes no podía ser explotada.

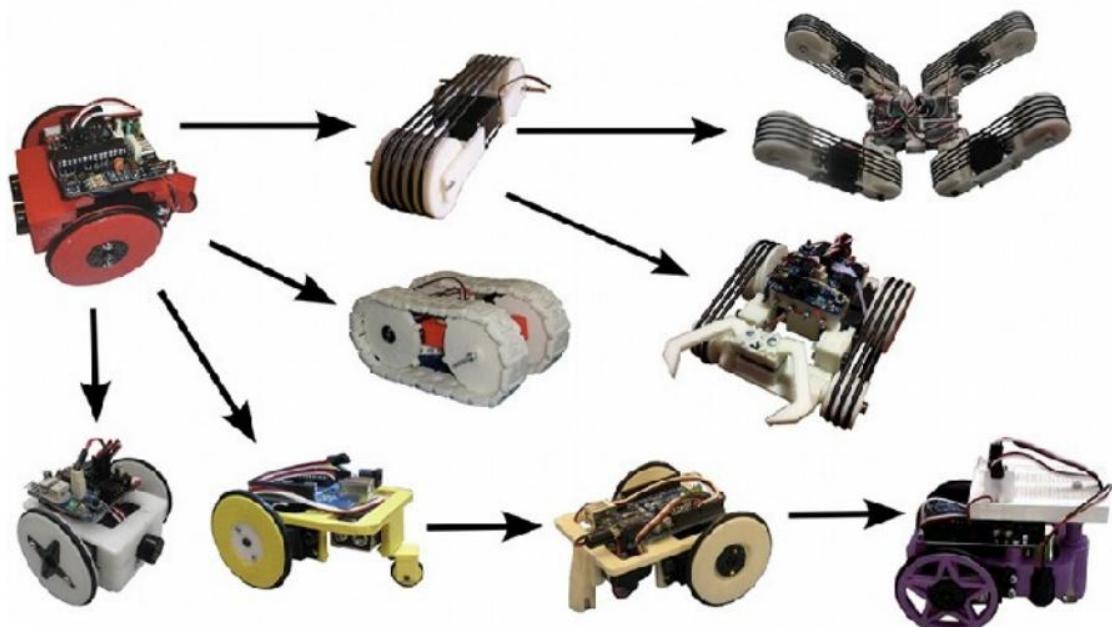


Figura 2-22 descendencia y evoluciones del robot MiniSkyBot

Otro concepto importante es el de “hardware libre”, que son proyectos cuyos planos son libres, y además están realizados con herramientas de diseño libres. Al igual que ha sucedido con GNU/Linux frente al software privativo, el hardware libre está suponiendo una gran revolución en el mundo de la robótica tanto educativa como comercial.

De todo esto se puede sacar una conclusión clara sobre la robótica imprimible y es que los robots libres unidos a la tecnología de la impresión 3D casera (printbots) han supuesto una revolución debido a sus propiedades de tele-copia y evolución gracias a la comunidad y que por lo tanto es necesario fomentar el uso de los printbots para la educación.

## CAPÍTULO 3.

### ANÁLISIS DE REQUISITOS Y CONCEPTOS GENERALES.

---

Para el desarrollo de la presente plataforma robótica lo primero que se hizo fue establecer los requisitos mínimos que se querían alcanzar para hacer un diseño dirigido por los requisitos.

Este proceso consiste en ir realizando un diseño incremental en el cual se diseña pensando en uno solo de los requisitos y luego se va adaptando a nuevos. Este método se pudo aplicar porque el diseño que se plantea no tiene una complejidad elevada y los requisitos no son muchos.

La elección de este sistema de desarrollo se debió a la posibilidad de ir validando el modelo por fases y detectar los posibles fallos de diseños en un entorno más acotado.

Los requisitos que se plantearon para el diseño de la plataforma fueron los siguientes:

- ✓ Imprimible: este requisito se estableció para que el modelo fuera replicable, fácil de fabricar, fácil de testear y, en última instancia, para minimizar al máximo el lapso de tiempo entre el diseño mecánico y la obtención del modelo físico para sus posteriores pruebas.
- ✓ Base robótica circular impulsada por dos ruedas motrices independientes: este requisito se estableció debido a la funcionalidad de la misma. Se plantea para la resolución de laberintos, y ya que tiene que tener la capacidad de poder moverse con total libertad en pasillos, la configuración más adecuada es una base circular que sea capaz de girar sobre su propio eje.
- ✓ La interfaz de usuario de la plataforma tienen que ser lo más simple posibles: este requisito se estableció debido al público al cual está destinado. Si se pretende que niños de menos de 8 años sean capaces de interactuar con la plataforma de manera eficiente e intuitiva, se necesita simplificar la interfaz de mando lo máximo posible.

## CAPÍTULO 4.

### SELECCIÓN DE COMPONENTES Y HERRAMIENTAS.

#### 4.1 Selección de componentes.

##### 4.1.1 Arduino DUE

El Arduino DUE tiene 64 GPIO en total y un microprocesador de 32-bit. En la siguiente imagen (figura 4-1) se puede ver una foto de este microcontrolador, donde se aprecia la colocación de cada entrada y salida, así como sus diferentes pines de comunicación.

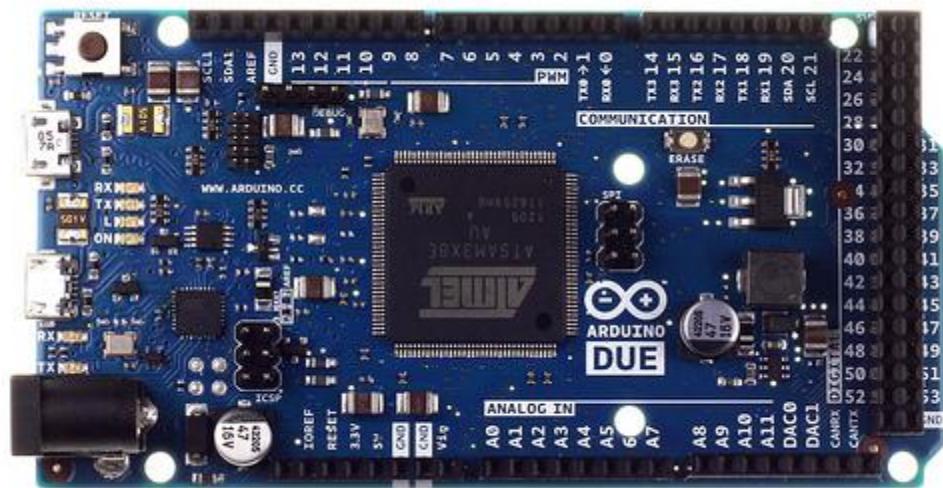


Figura 4-1 Arduino Due

##### 4.1.1.1 Especificaciones técnicas.

Microcontrolador: AT91SAM3X8E

Tensión de trabajo: 3.3V

Tensión recomendada de entrada: 7-12V

Tensión de entrada (min/max): 6-20V

Digital I/O Pins: 54 (of which 6 provide PWM)

Pines de entradas analógicas: 12

Pines de salidas analógicas: 2 (DAC)

Corriente total soportada en todas las líneas DC I/O: 130 mA

DC intensidad para 3.3V Pin: 800 mA

DC intensidad para 5V Pin: teórica 1A, recomendada 800 mA

Memoria Flash: 512 KB

SRAM: 96 KB (64 + 32 KB)

Velocidad del reloj: 84 MHz

Debug access: JTAG/SWD connector

Controlador DMA, que puede aliviar a la CPU de realizar tareas que requieren mucha memoria.

#### **4.1.1.2 Memoria**

El SAM3X tiene 512KB de memoria para almacenar código. La SRAM disponible es de 96Kb. Toda la memoria disponible (Flash, RAM y ROM) puede accederse directamente como una dirección de memoria plana.

Es posible borrar la memoria Flash de SAM3X con el botón integrado en la placa “erase”. Este nos permitirá eliminar el Sketch cargado desde la MCU. Para eliminarlo, deberemos mantener pulsado el botón “erase” durante unos segundos.

Entradas y salidas:

Digital I/O: pins del 0 al 53.

Cada uno de los pines de Arduino Due pueden ser usados como entradas o salidas, usando las funciones pinMode(), digitalWrite(), y digitalRead(). Ellas trabajan a 3.3V. Cada pin puede suministrar (soportar) una corriente de 3mA o 15 mA dependiendo del PIN, o recibir de 6 mA o 9 mA, dependiendo del PIN. Estos pines también poseen una resistencia del Pull Down desactivada por defecto de 100 KOhm. Además, algunos de estos pines tienen funciones específicas.

Serial: 0 (RX) y 1 (TX)

Serial 1: 19 (RX) y 18 (TX)

Serial 2: 17 (RX) y 16 (TX)

Serial 3: 15 (RX) y 14 (TX)

Usados para recibir (RX) y transmitir (TX) datos serie TTL (con niveles de 3.3 V). Pins 0 y 1 están unidos con los correspondientes pines USB-TTL serie del chip ATmega16U2.

PWM: Pins 2 a 13 y 44 a 46

Salidas PWM de 8 bit de resolución mediante la función analogWrite(). La resolución de las salidas analógicas puede cambiarse mediante la función analogWriteResolution().

SPI: Conector SPI (Conector ICSP en otras placas Arduino)

Estos pines soportan la comunicación SPI usando la Librería SPI. Los pines SPI están situados en el conector central de seis pines en el centro de la placa lo cual es físicamente compatible con Arduino Uno, Leonardo y Mega2560. EL conector SPI puede usarse para comunicarse solo con otros dispositivos SPI, no para programar el SAM3X con la técnica de programación circuito en serie. El puerto SPI del Due tiene otras características avanzadas que pueden usarse con Los métodos SPI extendidos para Due.

CAN: CANRX y CANTX

Estos pines soportan la comunicación CAN serie, pero todavía no está soportado con las APIs de Arduino.

“L” LED: 13

Este es un LED smd conectado al Pin 13. Cuando el pin está en HIGH, el Led se enciende, cuando el pin está en LOW, el LED se apaga, el pin 13 también es una salida PWM, por lo que podremos variar su intensidad.

TWI 1: 20 (SDA) y 21 (SCL)

TWI 2: SDA1 y SCL1.

Soporta comunicación TWI usando la Librería Wire.

Entradas analógicas: pines de A0 a A11

El Arduino Due trae 12 entradas analógicas, cada una de las cuales proporciona una resolución de 12 bit (4096 valores diferentes). Por defecto, la resolución de la lectura está establecida a 10 bit para que sea compatible con las aplicaciones diseñadas para otras placas Arduino. Es posible cambiar esta resolución ADC mediante la función `analogReadResolution()`. Las entradas analógicas de DUE, miden desde tierra hasta un valor máximo de 3,3v, si aplicamos más de 3,3v podemos dañar el chip SAM3X. La función `analogReference()` es ignorada en Arduino Due. El pin AREF está conectado a la referencia analógica del chip SAM3X con una resistencia puente, si queremos usar el pin AREF, debemos desoldar esa resistencia de la PCB.

DAC1 y DAC2

Estos pines nos proporcionan una salida analógica con una resolución de 12 bit (4096 niveles) con la función `analogWrite()`. Estos pines pueden usarse para crear una salida de audio mediante la librería Audio library.

#### 4.1.2 Motor Pololu 37D mm Gearmotors 100:1

Los motores elegidos para el sistema motriz principal han sido los motores Pololu de la familia 37D mm Gearmotors. Más concretamente su versión con reducción de 100:1, ya que son los que mejor se adaptan a nuestras pretensiones tanto de velocidad como de potencia, llegando a un compromiso óptimo entre ambas.

Características:

Dimensiones: 37x57mm

Reductora: 100:1

Peso: 300gramos aprox.

Alimentación recomendada: 12V

Potencia: 16kg/cm (12V/5A)

Velocidad: 100RPM (12V/300mA sin carga)

En la siguiente imagen (figura 4-2) podemos ver tanto una foto de este motor, como un plano esquemático con las dimensiones del mismo.

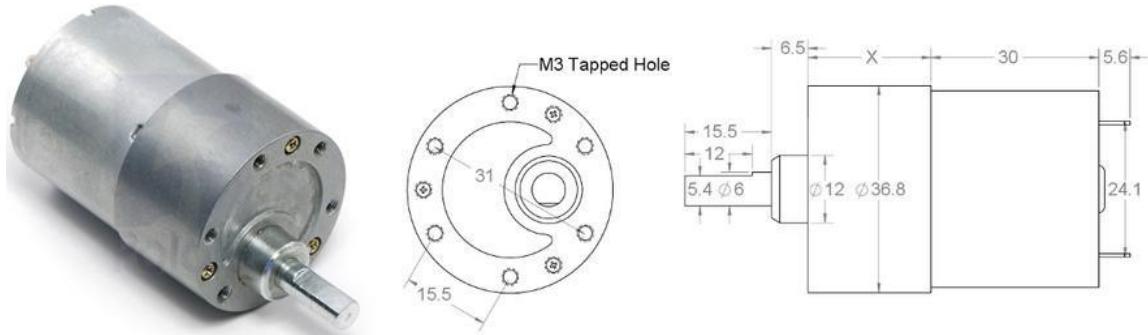


Figura 4-2 100:1 Metal Gearmotor 37Dx57L mm con 64 CPR Encoder

#### 4.1.3 Dual MC33926 Motor Driver Carrier

El dual MC33926 está compuesto por dos MC33926 con puente en H. Es capaz de suministrar hasta 3 A de corriente continua por canal de manera independiente a dos motores de corriente continua con una tensión de alimentación de 5 a 28 V.

Es capaz de tolerar corrientes de pico de hasta 5 A por canal durante unos segundos. El MC33926 funciona con señales de ancho de pulso (PWM) de un máximo de 20 kHz.

En la siguiente imagen (figura 4-3) podemos ver una imagen de esta etapa de potencia junto a unas indicaciones de conexión básico, dándonos una idea generas de las características del mismo.

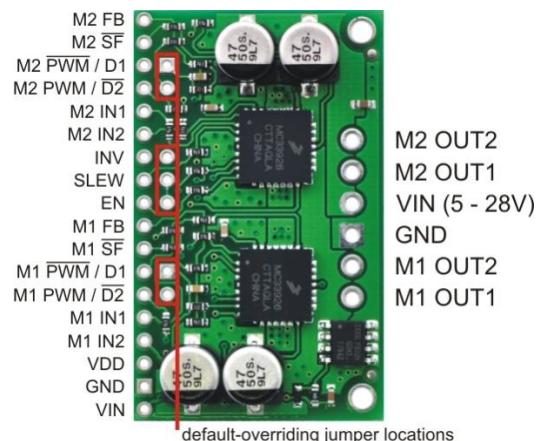


Figura 4-3 Dual MC33926 Motor Driver Carrier

#### 4.1.4 Prusa i3 Hephestos

La impresora 3D Prusa i3 Hephestos es un proyecto libre diseñado y desarrollado por el departamento de Innovación y Robótica de bq. Hephestos toma la base de la Prusa i3 y añade varias mejoras extraídas de otras impresoras como la PowerCode, usuarios de la comunidad RepRap, modificaciones de estas piezas y diseños propios del departamento.



Figura 4-4 Prusa i3 Hephestos

##### 4.1.4.1 Características técnicas.

###### 4.1.4.1.1 Dimensiones

Dimensiones impresora: (x)460 x (y)370 x (z sin rollo)510 (z con rollo)583 mm

Dimensiones área de impresión: (x)215 x (y)210 x (z)180 mm

Dimensiones caja: (x)400 x (y)400 x (z)250mm

###### 4.1.4.1.2 Mecánica general

Marco y base de aluminio pintado al polvo

Barras de cromo duro para los carros X, Y, Z

Rodamiento lineal de bolas LM8UU para X, Y, Z

Rodamiento axial de bolas B623ZZ para las poleas X, Y

Cadenas portacables Igus

Acoplamientos flexibles para las varillas roscadas del eje Z

Sistema de nivelado de base de impresión con 4 puntos y amortiguación

Sistema de cambio rápido de base de impresión con Clips

Ventiladores brushless axiales con rodamientos de bolas.

#### **4.1.4.1.3 Resolución de impresión**

Muy alta: 60 micras

Alta: 100 micras

Media: 200 micras

Baja: 300 micras

#### **4.1.4.1.4 Mecánica extrusor**

Extrusor de diseño propio

Boquilla de 0.4mm

Disipador de aletas con ventilador axial

Tobera de refrigeración de pieza

#### **4.1.4.1.5 Velocidad de impresión**

Velocidad recomendada: 50 mm/s

Velocidad máxima recomendada: 80 mm/s

#### **4.1.4.1.6 Electrónica**

Ramps 1.4

Mega 2560

Pantalla de LCD con encoder rotativo y con pulsador para la navegación

Base fría de cristal tamaño 220 x 220 x 3 mm

Fuente de alimentación de 220 AC 12 DC 100W

Termistor 100k en extrusor

Cartucho calefactor 40W 12V

#### **4.1.4.1.7 Software**

Firmware derivado de Marlin

Entorno recomendado: Cura Software

Archivos admitidos: .gcode

#### **4.1.4.1.8 Comunicaciones**

Lector de tarjetas SD estándar

Puerto USB tipo B

#### **4.1.4.1.9 Materiales**

Filamento PLA de 1.75 mm

## 4.2 Entornos de desarrollo

### 4.2.1 IDE Arduino 1.5.1.r2

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.



Figura 4-5 Logo arduino.cc

El software de arduino está constituido por un entorno de desarrollo (IDE) y un conjunto de librerías propias. El IDE de desarrollo está escrito en Java y se basa en el entorno de programación de Processing. Las librerías están escritas en C y C++ y compiladas usando avr-gcc y AVR Libc.

### 4.2.2 IDE Eclipse + Python 2.7

Eclipse es un software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

Fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. En la actualidad, su desarrollo está a cargo de la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

PyDev es un IDE de Python para Eclipse, que puede ser usada para desarrollo sobre Python, Jython y IronPython.



Figura 4-6 Logos de Eclipse Pydev y Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses "Monty Python". Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

#### 4.2.3 MATLAB R2010a

MATLAB, figura 4-7, (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Mac OS X.

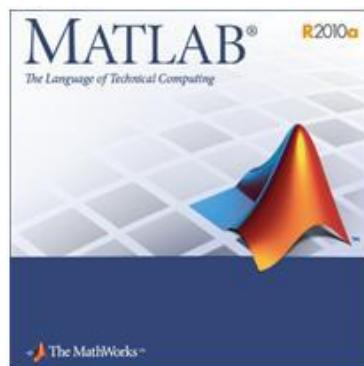


Figure 4-7 MATLAB r2010a

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

#### 4.2.4 Autodesk Inventor Professional

El software de CAD

3D Autodesk® Inventor® ofrece un conjunto de herramientas fáciles de usar para diseño mecánico, documentación y simulación de productos en 3D.



Figura 4-8 Autodesk Inventor Professional

Autodesk Inventor se basa en técnicas de modelado paramétrico. Los usuarios comienzan diseñando piezas que se pueden combinar en ensamblajes. Corrigiendo piezas y ensamblajes pueden obtenerse diversas variantes. Como modelador paramétrico, no debe ser confundido con los programas tradicionales de CAD. Inventor se utiliza en diseño de ingeniería para producir y perfeccionar productos nuevos, mientras que en programas como Autocad se conducen solo las dimensiones. Un modelador paramétrico permite modelar la geometría, dimensión y material de manera que si se alteran las dimensiones, la geometría actualiza automáticamente basándose en las nuevas dimensiones. Esto permite que el diseñador almacene sus conocimientos de cálculo dentro del modelo, a diferencia del modelado no paramétrico, que está más relacionado con un “tablero de bocetos digitales”.

Los bloques de construcción cruciales de Inventor son las piezas. Se crean definiendo las características, que a su vez se basan en bocetos (dibujos en 2D). La ventaja de este diseño es que todos los bocetos y las características se pueden corregir más adelante, sin tener que hacer de nuevo la partición entera. Este sistema de modelado es mucho más intuitivo que en ambientes antiguos de modelado, en los que para cambiar dimensiones básicas era necesario generalmente suprimir el archivo entero y comenzar de cero.

Como parte final del proceso, las partes se conectan para hacer ensamblajes. Los ensamblajes pueden consistir en piezas u otros ensamblajes. Las piezas son ensambladas agregando restricciones entre las superficies, bordes, planos, puntos y ejes.

Este método de modelado permite la creación de ensamblajes muy grandes y complejos, especialmente porque los sistemas de piezas pueden ser puestos juntos antes de que se ensamblen en el ensamblaje principal; algunos proyectos pueden tener muchos sub-ensamblajes parciales.

Inventor utiliza formatos específicos de archivo para las piezas (.IPT), ensamblajes (.IAM) , vista del dibujo (.IDW y .DWG) y presentaciones (IPN), pero el formato del archivo de AutoCAD .DWG puede ser importado/exportado como boceto.

## CAPÍTULO 5.

### DISEÑO MECANICO.

#### 5.1 Diseño de un robot de dos ruedas

Debido a su gran maniobrabilidad en espacios reducidos y a la robustez de estos diseños, se decidió por la utilización de esta configuración para la motricidad de la plataforma robótica.

Debido a la búsqueda de una plataforma fácil de manejar y lo más simple posible, se decidió también porque la plataforma tuviera una forma cilíndrica con el eje motriz en el centro.

Teniendo estas dos consideraciones se llegó a un diseño en el cual la forma de la plataforma no supusiese un problema a la hora de controlarlo y en la que la configuración de dos ruedas le dotaba de gran movilidad.

Tras realizar el diseño que en los puntos siguientes aparecen en esta memoria, se llegó al diseño de la plataforma que vemos en la figura 5-1.

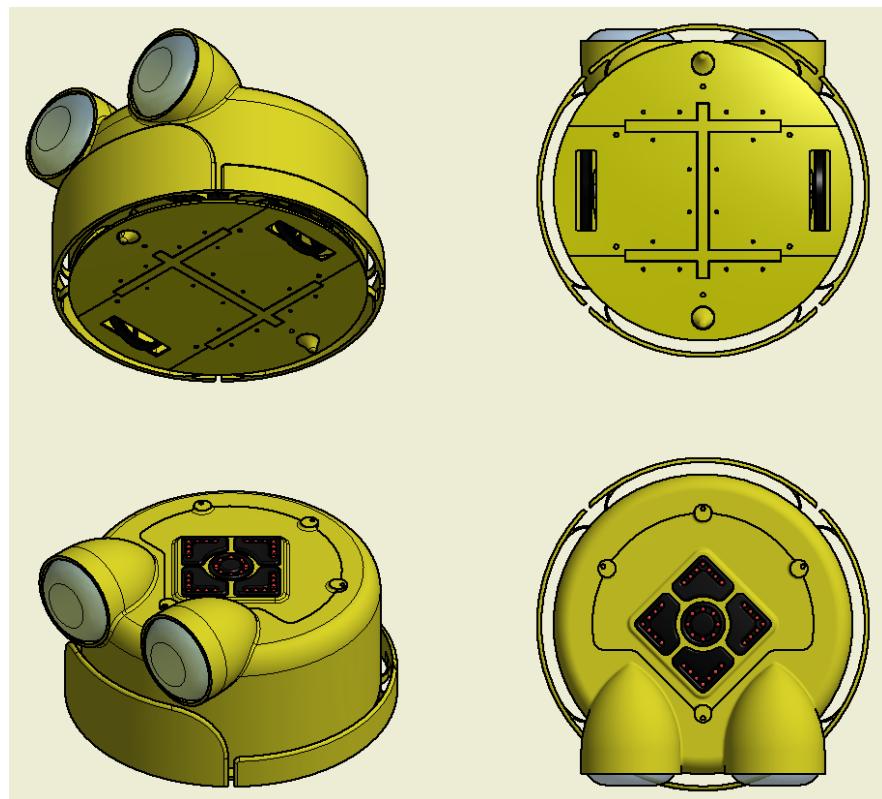


Figura 5-1 Diseño CAD de la plataforma robótica final

## 5.2 Pasos previos al diseño mecánico.

Antes de poder abordar el diseño mecánico se eligieron todos los componentes electrónicos así como los motores.

Esto es un punto importante, ya que el tamaño de la plataforma depende en gran medida de los componentes que tiene que alojar.

Teniendo en cuenta el tamaño de los motores se llegó a la conclusión de que la plataforma debería tener un mínimo de 270 mm de diámetro para que pudiera alojar los motores con las ruedas acopladas.

Por último se determinó la altura teniendo en cuenta tanto la electrónica como el diámetro de las ruedas, llegando a una altura de 135 mm para poder asegurar que todos los componentes entrasen sin problema.

Ya con estas dos restricciones de tamaño se pasó al diseño mecánico de la plataforma.

## 5.3 Diseño CAD de la plataforma.

Una vez se tuvieron elegidos los componentes a utilizar y el tamaño de la plataforma, lo primero que se diseñó fue la base.

Para esta parte del diseño se plantearon los siguientes requisitos:

Sujeciones para los motores de corriente continua.

Aberturas para que las ruedas pudieran tocar el suelo.

Soporte trasero y delantero para estabilizar la plataforma horizontalmente a la altura de las ruedas.

Sujeciones para las protecciones laterales.

Con estas características en mente, se llegó a una primera solución teórica, la cual podemos ver en la figura 5-2

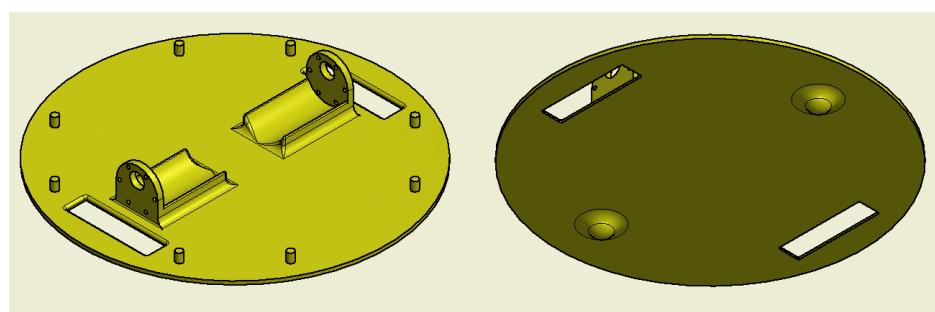


Figura 5-2 Diseño CAD de la base de la plataforma teórica.

Esta plataforma, aun cumpliendo todos los requisitos de diseño, no cumplía los requisitos de fabricación ya que no podía ser construida mediante impresión 3d.

Para que pudiera cumplir los requisitos de fabricación se realizó una segunda iteración en la que se incorporaron, además, los anclajes que se usarían para la carcasa superior.

### 5.3.1 Diseño de la base de la plataforma adaptada al proceso de fabricación.

Lo primero que se realizó fue dividir la plataforma en secciones de un máximo de 15cm x 15cm y que tuviera una superficie plana desde la que poder imprimir.

Todas estas consideraciones son debido a que el diseño se ha planteado para que pueda ser impreso por la gran mayoría de las impresoras 3D del mercado.

Tras la separación en secciones y el realizar un diseño capaz de ser ensamblado como una única pieza, se llegó al diseño que aparece en la figura 5-3.

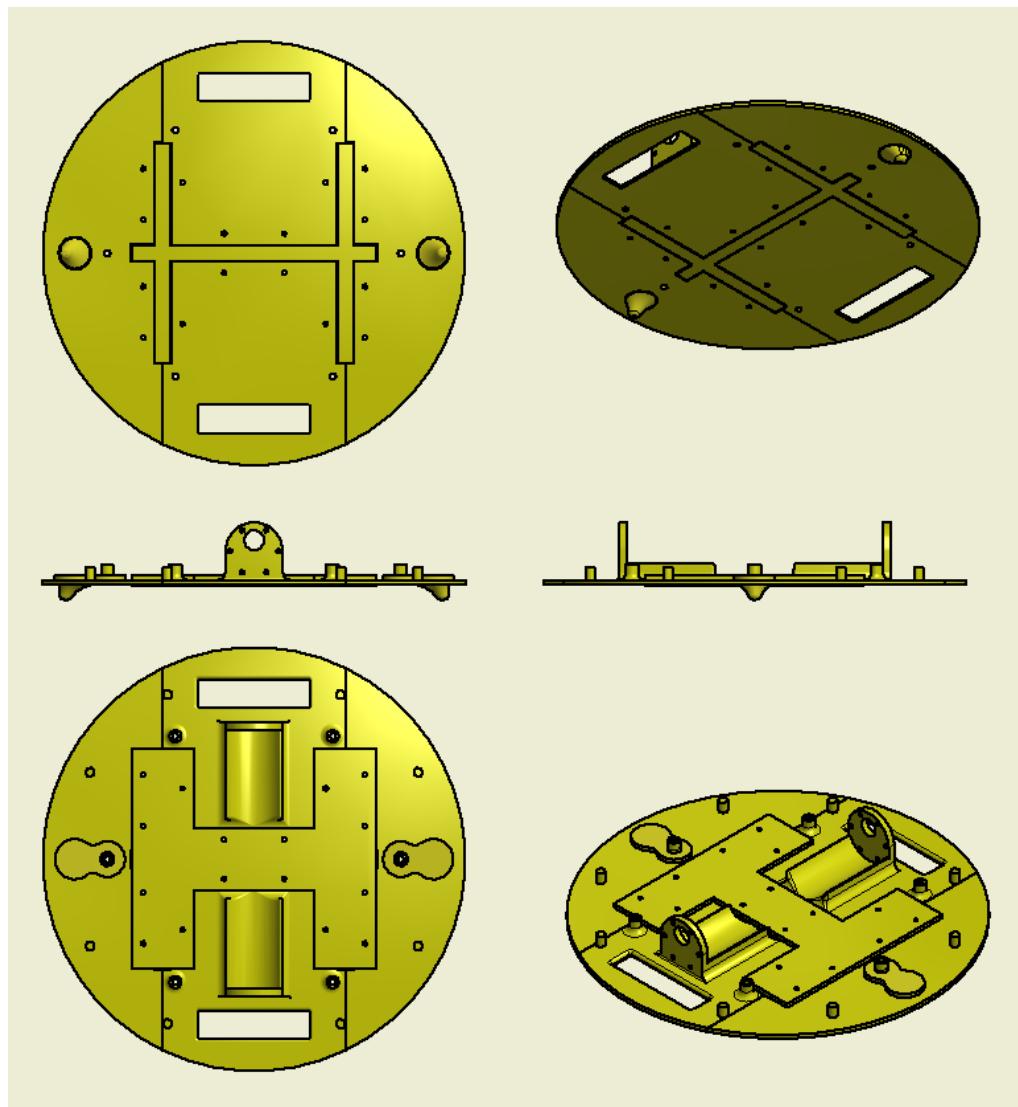


Figura 5-3 Diseño CAD de la base de la plataforma imprimible.

Para este modelo modular se diseñaron un total de cuatro componentes, una pieza con la sujeción para un motor (figura 5-4), una pieza con la pieza parte de la circunferencia frontal y trasera (figura 5-5), una pieza con los soportes para estabilizar la plataforma horizontalmente (figura 5-6) y una pieza central para el acoplamiento de todas las piezas (figura 5-7).

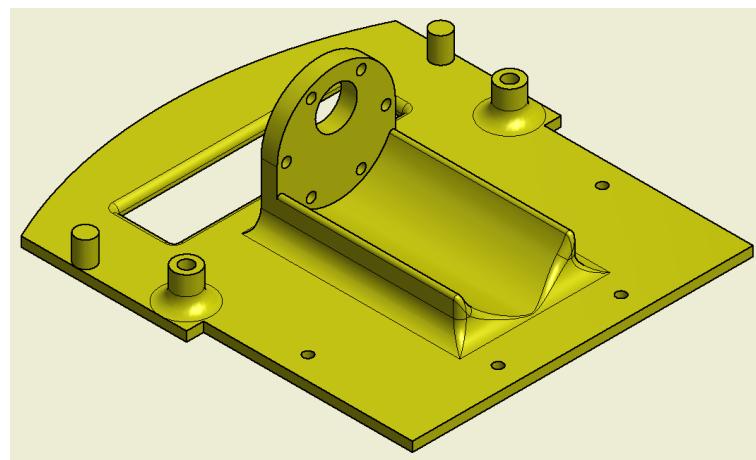


Figura 5-4 Diseño CAD del soporte del motor.

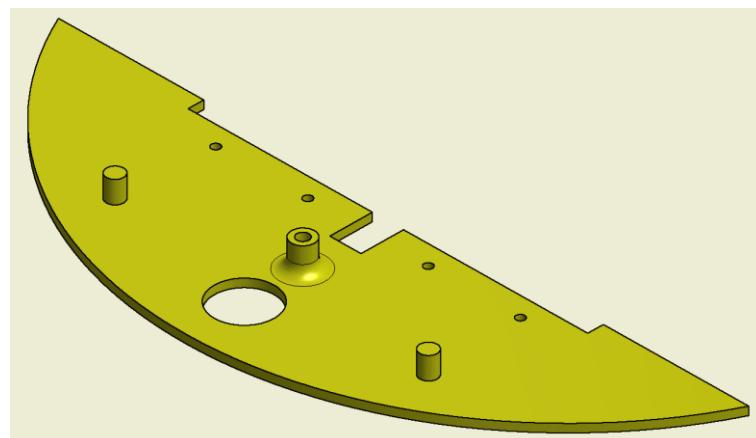


Figura 5-5 Diseño CAD de la parte posterior de la base.

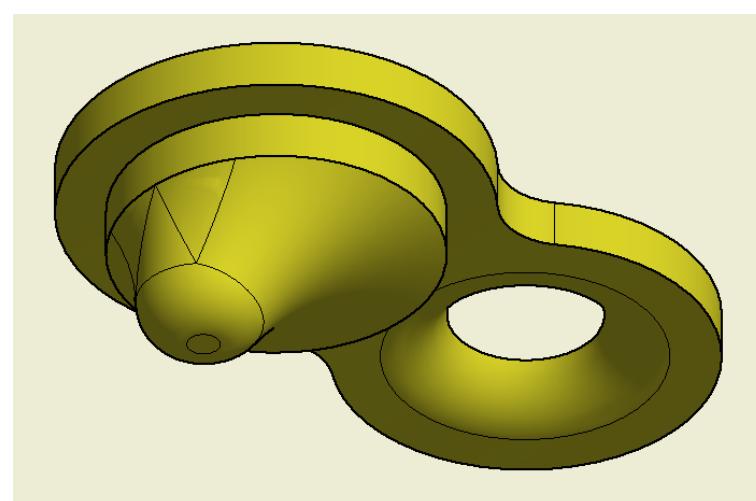


Figura 5-6 Diseño CAD del apoyo frontal contra el suelo de la base.

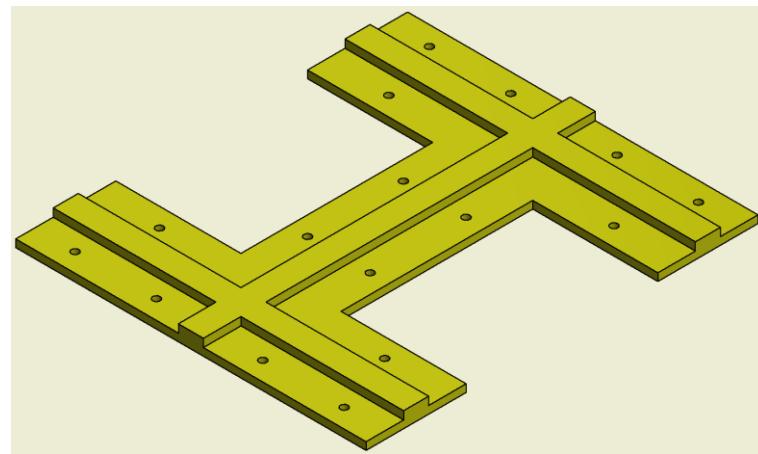


Figura 5-7 Diseño CAD de la piza que ensambla todas las partes de la base.

Al haber realizado un diseño modular simétrico, cada una de las piezas se requieren en dos unidades menos la pieza central que sirve para acoplarlas todas, de la cual solo se necesita una unidad.

Una vez se tienen todas las piezas, el proceso de ensamblaje se realiza mediante la utilización de tornillos de métrica M3 o por apriete. Este proceso se puede ver en la figura 5-8.

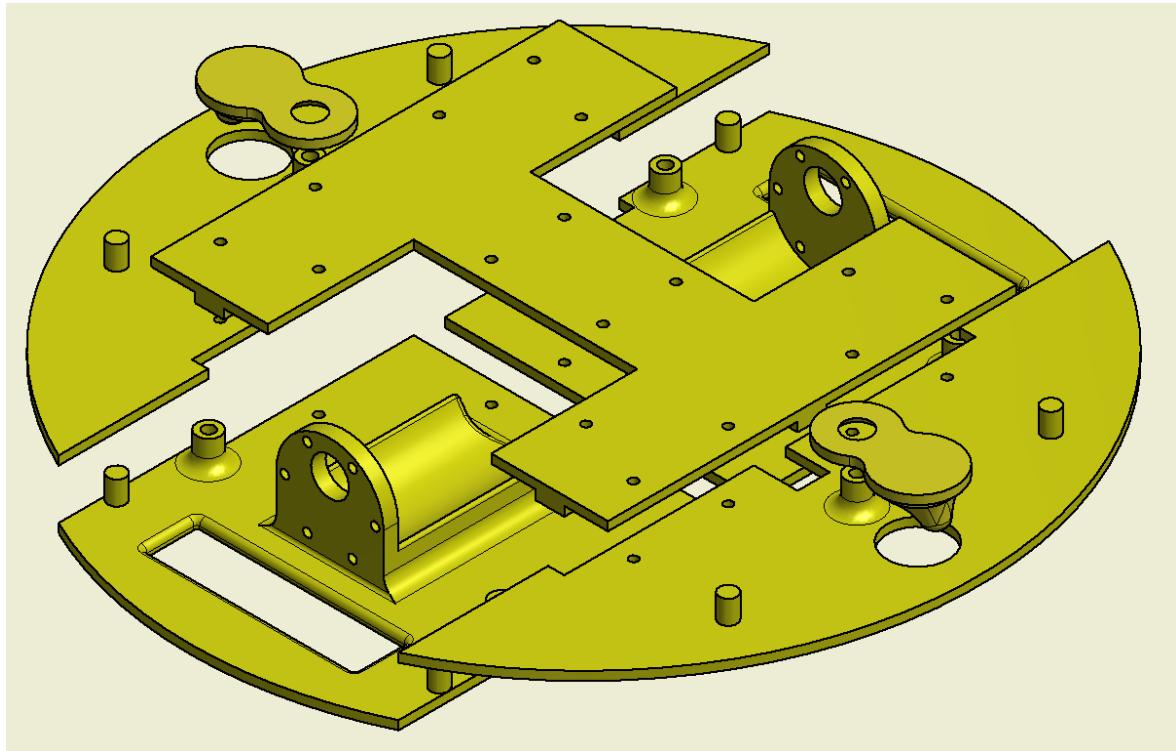


Figura 5-8 Diseño CAD del montaje de la base imprimible.

Ya con la plataforma terminada, se pasó a la incorporación de todos los elementos mecánicos y electrónicos para ver que todo encajaba en su ubicación final y detectar posibles fallos.

El resultado de la incorporación de los elementos fue todo un éxito y su resultado se puede ver en la figura 5-9, en la que podemos ver el modelo de la plataforma totalmente ensamblada junto con los motores, ruedas y electrónica de control acoplados.

Los únicos componentes que no aparece en la imagen son la electrónica de potencia y la batería, pero esto es debido a que la batería se tuvo en cuenta en el diseño de la cúpula superior y que la electrónica de potencia tiene un tamaño tan pequeño que no influye en esta etapa del diseño.

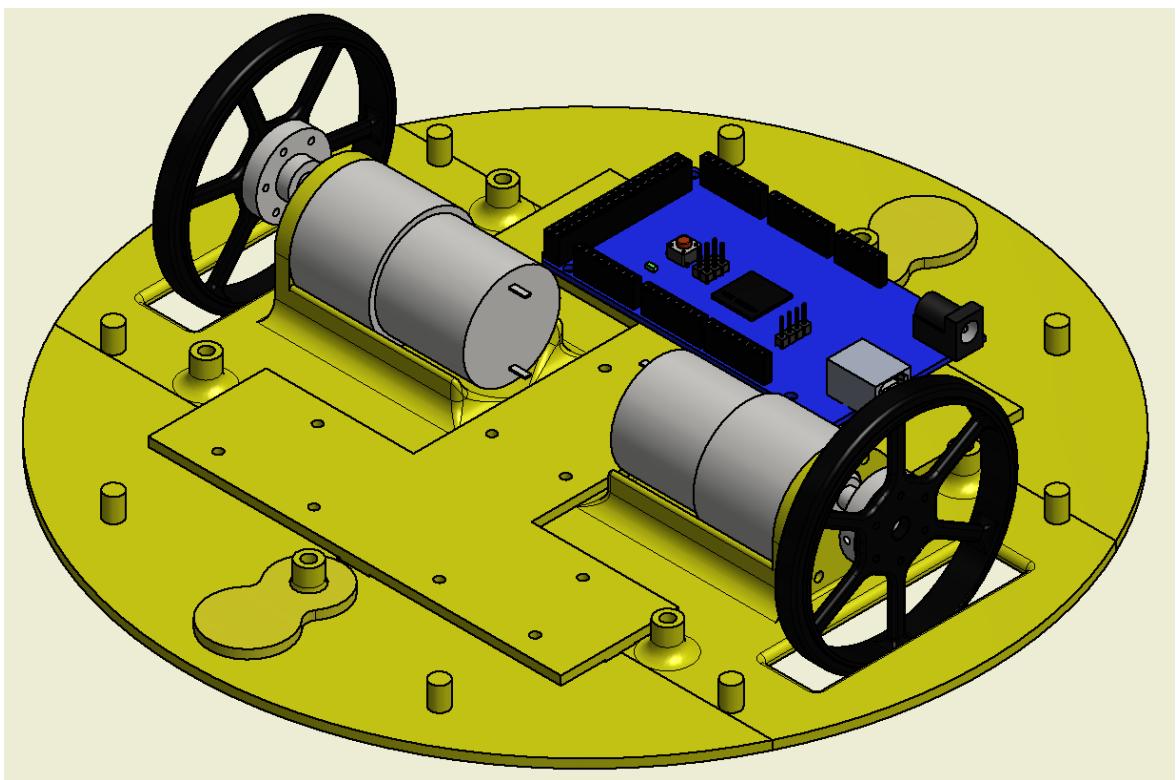


Figura 5-9 Diseño CAD de la base de la plataforma con los motores y la electrónica.

### 5.3.2 Diseño de las protecciones laterales flexibles.

Para dotar a la plataforma de medidas de seguridad suficientes para que se detuviese en el caso de que impactase con algún obstáculo, se planteó el diseño de unas protecciones laterales que fueran flexibles.

Gracias a la utilización de plástico para la construcción del mecanismo, se diseñó teniendo en cuenta que fuera resistente y lo suficiente flexible para la absorción de impactos, activarse unos actuadores que parasen el funcionamiento de la plataforma y fuera capaz de recuperar su posición inicial.

También se aprovechó para diseñar una de las protecciones de manera diferente para que asemejase la parte delantera del robot, haciendo que fuera más fácil identificarla y, además, que tuviera una forma agradable que haga que los niños se interesen por él y encuentren divertido el realizar los ejercicios con él.

Para las protecciones laterales y traseras se optó con un diseño semicircular que asemejase a un cinturón. El diseño se puede ver en la figura 5-10.

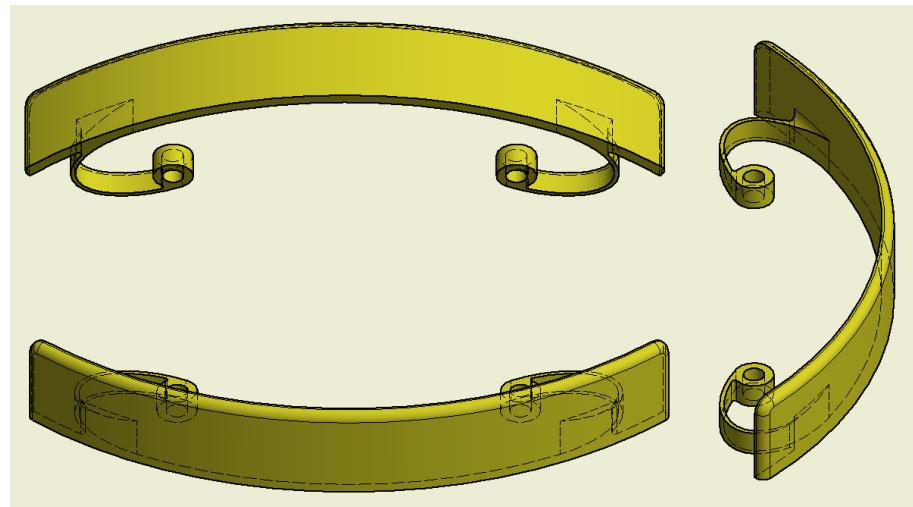


Figura 5-10 Diseño CAD de la protección lateral.

Para el caso de la protección delantera se decidió por una protección más alta que recordase a una boca en el modelo final, para que los niños puedan personalizar la plataforma y minimizar el rechazo que pudiera causarles.

El modelo para la protección delantera se puede ver en la figura 5-11, la cual es claramente más grande, identificándose fácilmente.

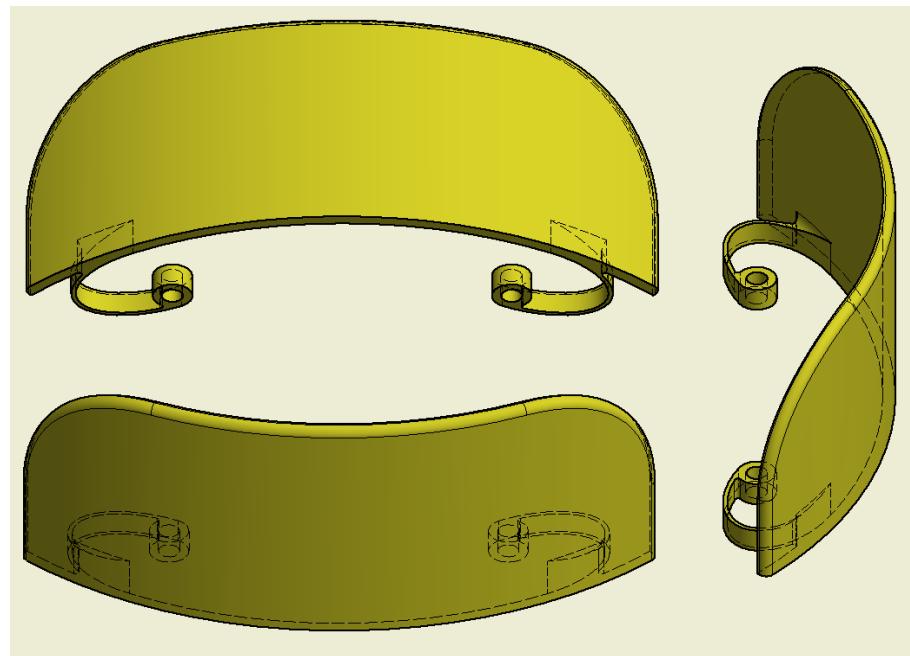


Figura 5-11 Diseño CAD de la protección frontal.

Una vez se tenían tanto las protecciones delanteras como laterales, se pasó a la comprobación de ensamblaje. Para ello, se colocaron los modelos de las protecciones sobre la base previamente diseñada para su comprobación. El correcto resultado de esta fase se puede ver en la figura 5-12.

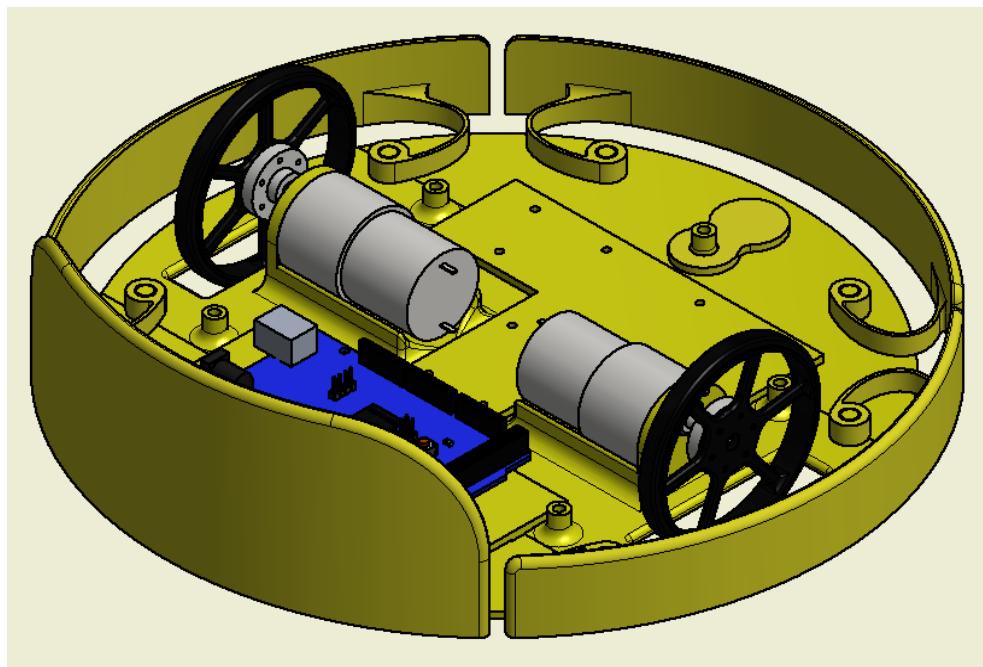


Figura 5-12 Diseño CAD de la base de la plataforma con las protecciones colocadas.

### 5.3.3 Diseño de la cúpula.

Una vez diseñado todas las partes mecánicas funcionales, se pasó al diseño de una cúpula que cumpliese dos funciones, proteger la electrónica y hacer a la plataforma estéticamente agradable.

Para esto se decidió por diseñar un cuerpo cilíndrico con dos ojos grandes en la parte frontal y una botonera de dirección en la parte superior a modo de interfaz de mando, siendo el resultado final como el que aparece en la figura 5-13.

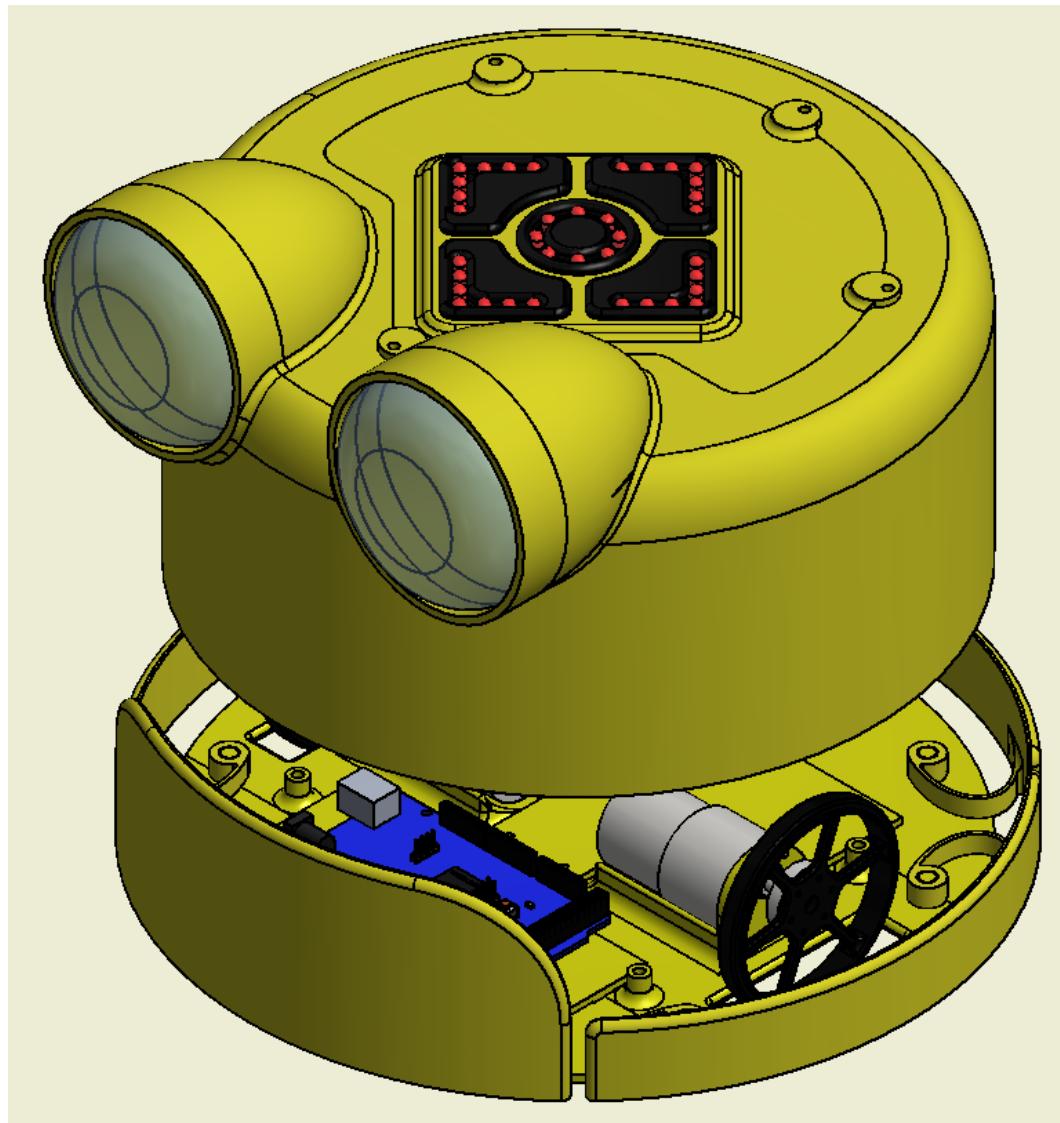


Figura 5-13 Diseño CAD de la plataforma con la cupula teórica.

Al igual que pasó con la base de la plataforma, aun cumpliendo todos los requisitos de diseño, no cumplía los requisitos de fabricación ya que no podía ser construida mediante impresión 3d.

Para que pudiera cumplir los requisitos de fabricación se realizó una segunda iteración en la que se incorporó material de soporte para poder ser impresa.

#### 5.3.4 Diseño de la cúpula adaptada al proceso de fabricación.

Lo primero que se realizó fue dividir la plataforma en secciones de un máximo de 15cm x 15cm y que tuviera una superficie plana desde la que poder imprimir.

Tras la separación en secciones y el realizar un diseño capaz de ser ensamblado como una única pieza, se llegó al diseño que aparece en la figura 5-14.

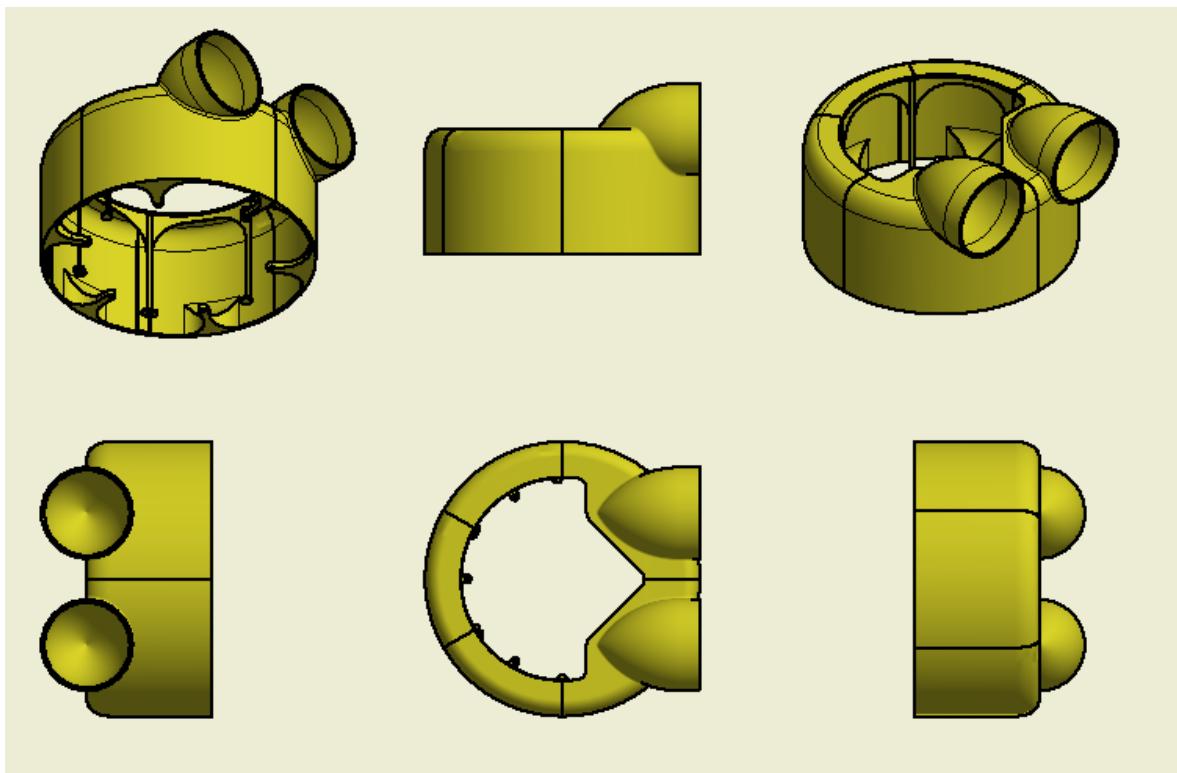


Figura 5-14 Diseño CAD de la cúpula imprimible.

Para este modelo modular se diseñaron un total de tres componentes, una pieza que, al unir tres, compusiese la mitad trasera de la cúpula (figura 5-15) y dos piezas simétricas que para la mitad frontal en la que se han dejado dos aberturas cónicas en forma de ojos para dotar de una estética más amigable a la plataforma (figuras 5-16 y 5-17).

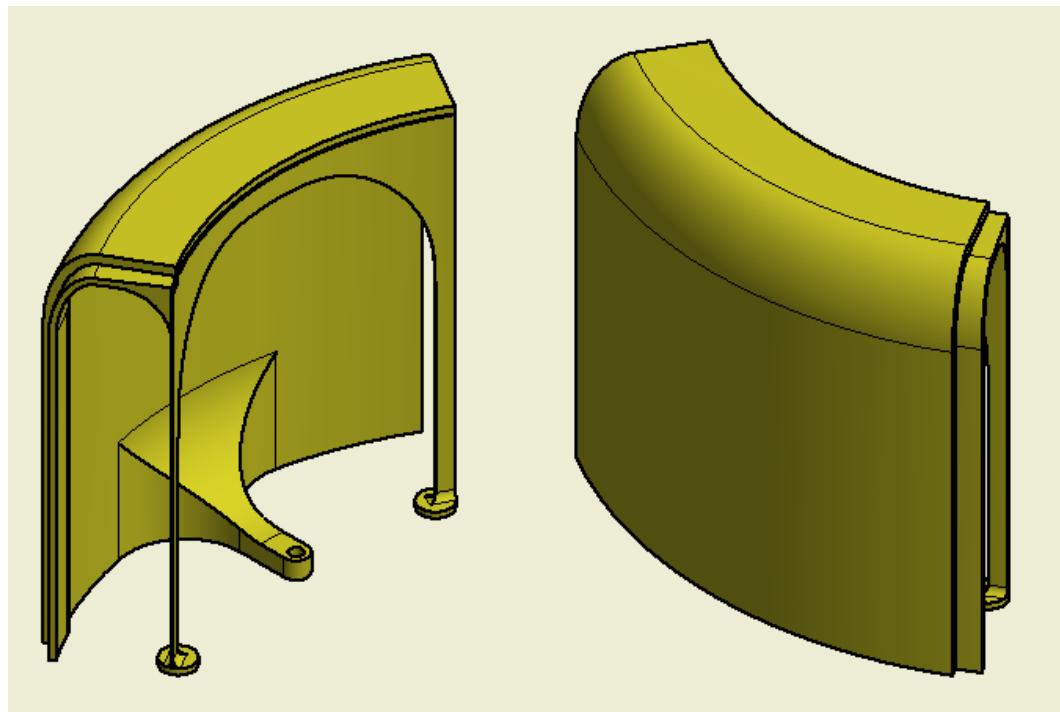


Figura 5-15 Diseño CAD de la parte trasera de la cúpula.

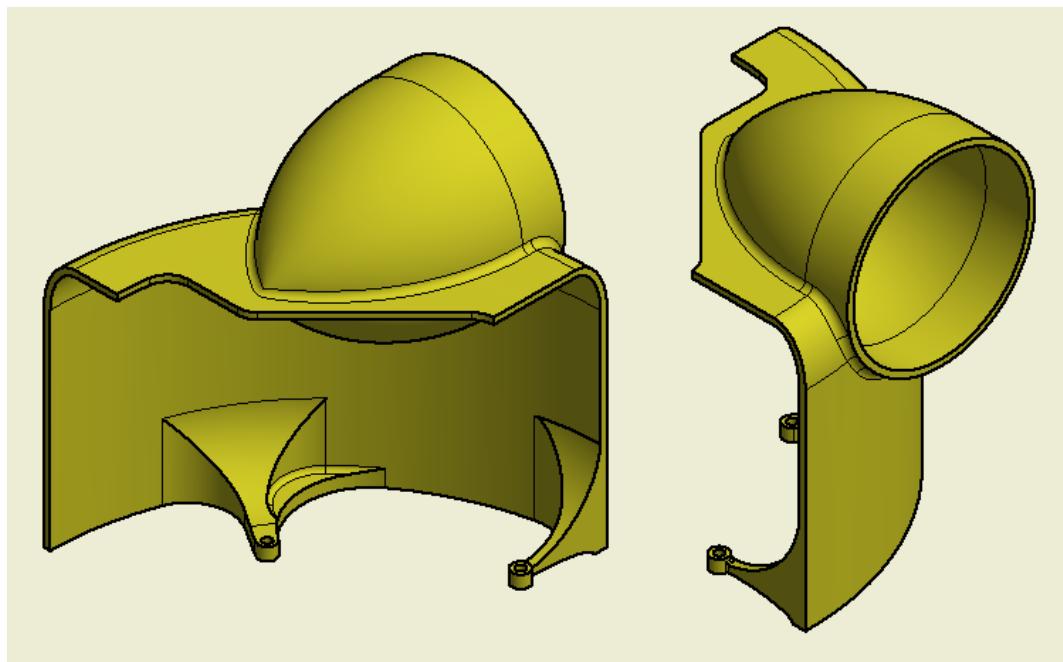


Figura 5-16 Diseño CAD de la parte frontal izquierda de la cúpula.

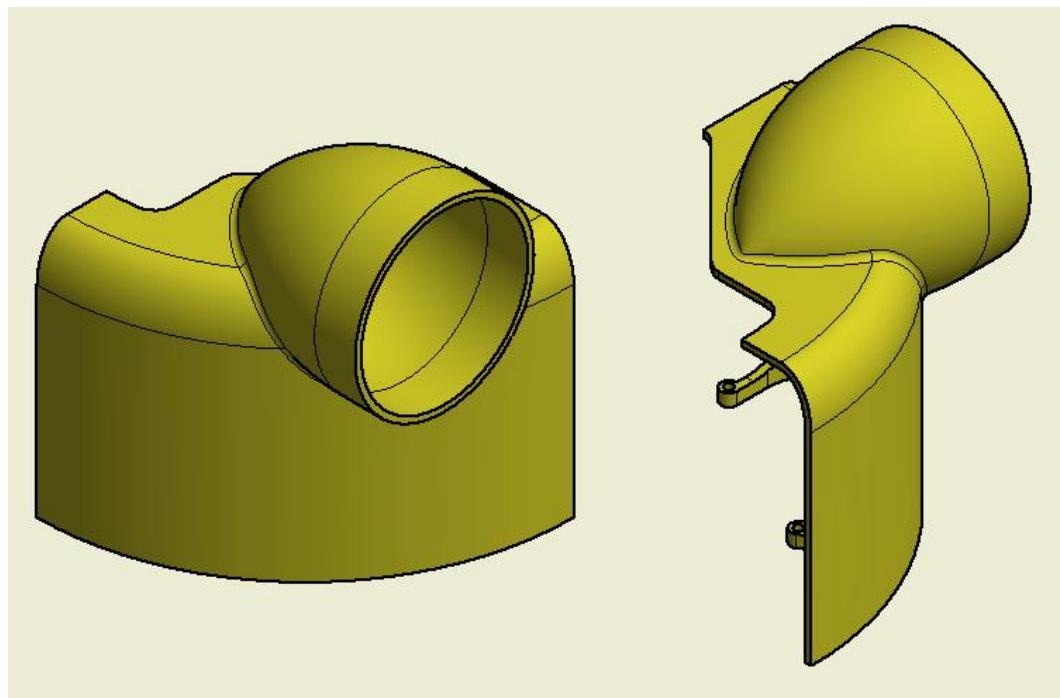


Figura 5-17 Diseño CAD de la parte frontal derecha de la cúpula.

Una vez se tienen todas las piezas, el proceso de ensamblaje se realiza mediante la utilización de tornillos de métrica M3.

Las dos piezas delanteras utilizan dos tornillos M3 cada una, por lo que quedan fijas por sí mismas.

Esto no pasa en el caso de las piezas traseras que solo cuentan con un tornillo por pieza. Para solventar esto, se les ha añadido una pestaña lateral que hace que, al tener todas las piezas juntas, solo exista una manera de encajar correctamente, por lo que quedan fijadas perfectamente en el conjunto.

Al ser un modelo igual que el original, pero modular, el ensamblaje con la base de la plataforma no cambia, quedando el conjunto en este punto como el que aparece en la imagen 5-18.

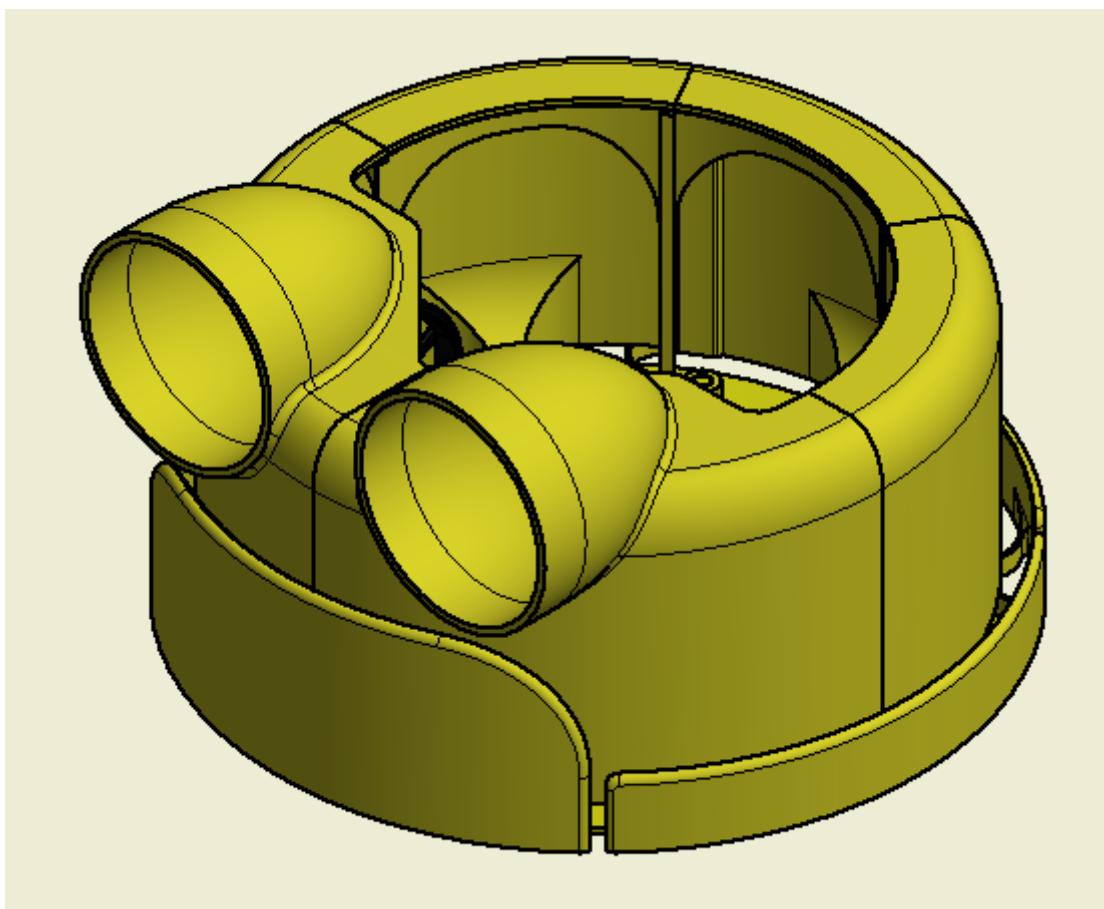


Figura 5-18 Diseño CAD de la plataforma con la cúpula imprimible colocada.

#### 5.4 Diseño final fabricado.

Tras la realización de estos diseños, se paso a su impresión para poder tener un modelo físico real con el que poder implementar el sistema de control real y validar el sistema de trazado de trayectorias.

Al finalizar el montaje, el primer prototipo de la plataforma se puede ver en las figuras 5-19, 5-20, 5-21, 5-22.

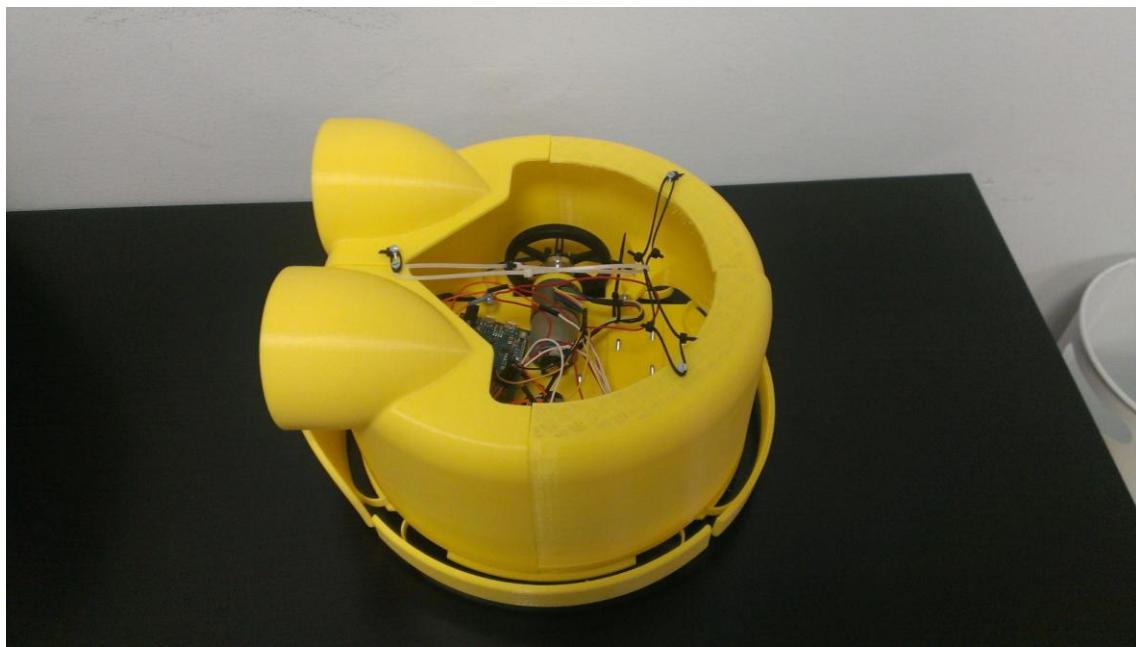


Figura 5-19 Imagen superior de la plataforma real.

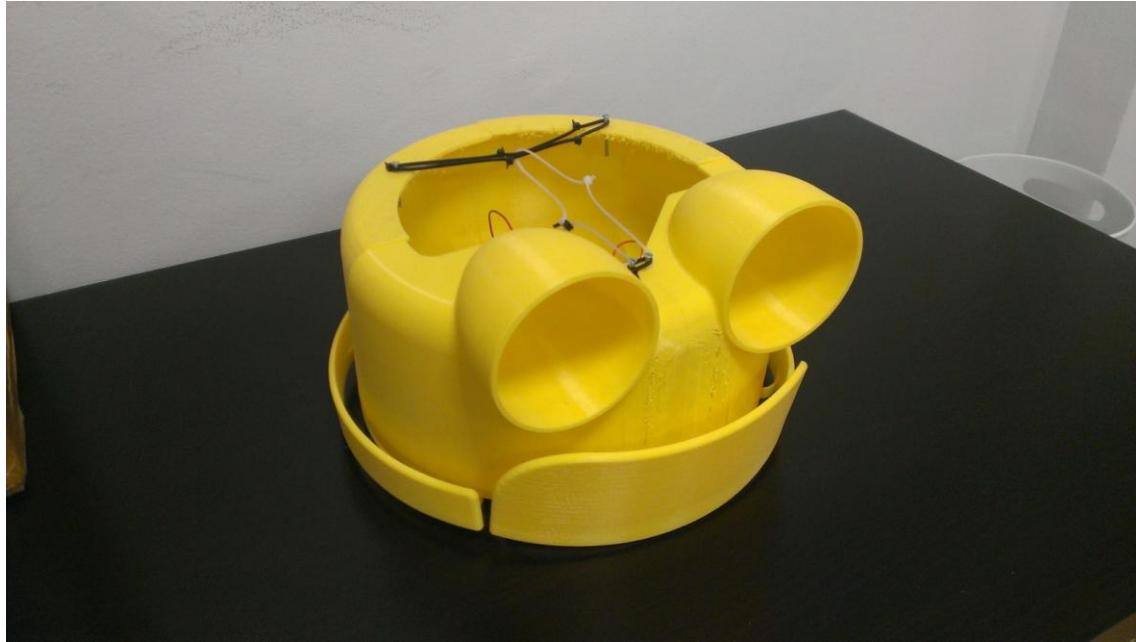


Figura 5-20 Imagen completa de la plataforma real.

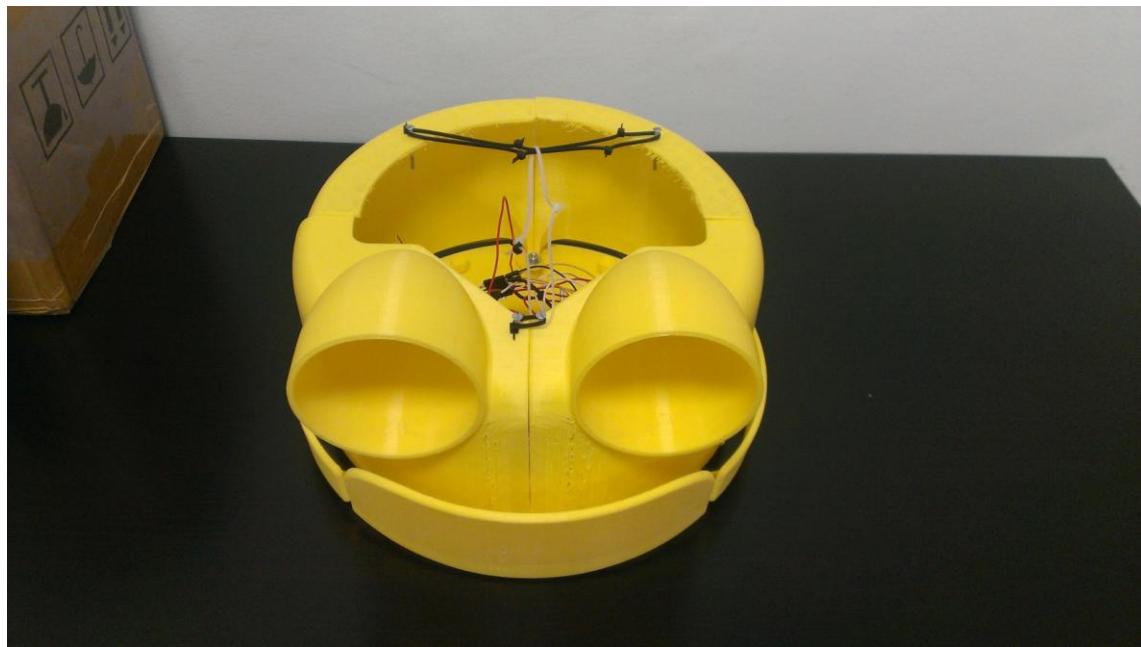


Figura 5-21 Imagen frontal de la plataforma real.



Figura 5-22 Imagen lateral de la plataforma real.

## CAPÍTULO 6.

### DISEÑO ELECTRÓNICO.

#### 6.1 Requisitos mínimos de diseño.

Lo primero que se estableció a la hora de realizar el diseño electrónico fueron los requisitos mínimos que requería la plataforma para su correcto funcionamiento.

Lo primero que se planteó fueron los motores a utilizar, como se pretendía desarrollar, como hemos visto en el capítulo anterior, un robot de dos ruedas, se buscaron dos motores fiables y realimentados.

Entre todos lo que se encontraron, destacaron los pololu 37D debido a su relación velocidad potencia, con 100 rpm de velocidad máxima y 16 kg-cm de par motor.

Otra de las ventajas que aportaba este motor es que existe un modelo, que es el utilizado en el presente proyecto, que cuenta con la realimentación por dos canales como el que aparece en la figura 6-1.



Figura 6-1 Imagen de los cables del encoder del motor.

Otra de sus múltiples ventajas era que contaba con una gran variedad de accesorios para la utilización de ruedas, que facilitan en gran medida tanto el montaje como la calibración de todo el diseño mecánico, además de dotar de robustez al conjunto motor-eje-rueda como se ve en la figura 6-2



Figura 6-2 Imagen del conjunto motor, adaptador y rueda.

Después de tener los motores elegidos, se busco la etapa de potencia necesaria para poder controlarlos de manera eficiente y correcta.

Al igual que pasara con los accesorios, pololu cuenta con una gran cantidad de electrónica de potencia diseñada para sus motores.

Como en este caso teníamos dos motores iguales, se opto por utilizar una dual mc33926 (figura 6-3), que es una etapa de potencia pensada para controlar dos motores de corriente continua como los elegidos.

Admite una alimentación de entrada de de 5v a 28v y corrientes continuas de 3A y 5A de pico, por lo que cumple por completo con las necesidades que van a requerir los motores, así como tener la posibilidad de usar la misma alimentación para el Arduino Due y la etapa de potencia.

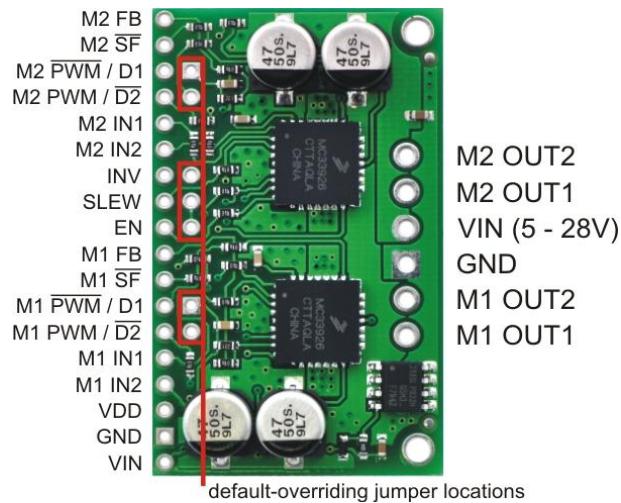


Figura 6-3 Imagen de conexionado de la etapa de potencia.

Gracias a este modelo, podemos controlar ambos motores de una manera cómoda, utilizando una señal PWM para la velocidad de cada motor y 2 señales digitales, por motor, para indicar la dirección.

Para terminar, faltaba por escoger la electrónica de control, para lo cual se optó por una Arduino Due (figura 6-4) debido a su gran cantidad de entradas/salidas de diferente tipo, además de contar con un microprocesador de 32bits

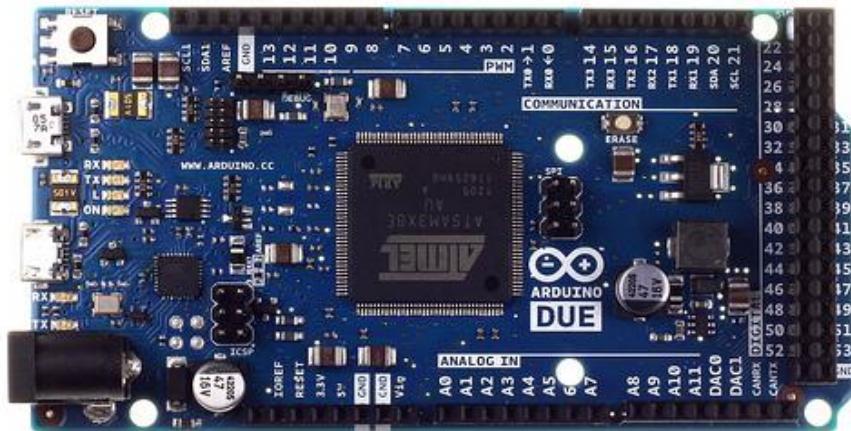


Figura 6-4 Imagen del Arduino DUE utilizado.

## 6.2 Diseño básico de conexionado y alimentación.

Una vez escogidos todos los componentes electrónicos principales, se pasó al cálculo y diseño de las necesidades de conexionado.

Lo primero que se calculo fueron las conexiones necesarios del bloque de los motores y la etapa de potencia.

En el caso de los motores, cada uno de ellos tiene 6 conexiones como se puede ver en la figura 6-5.

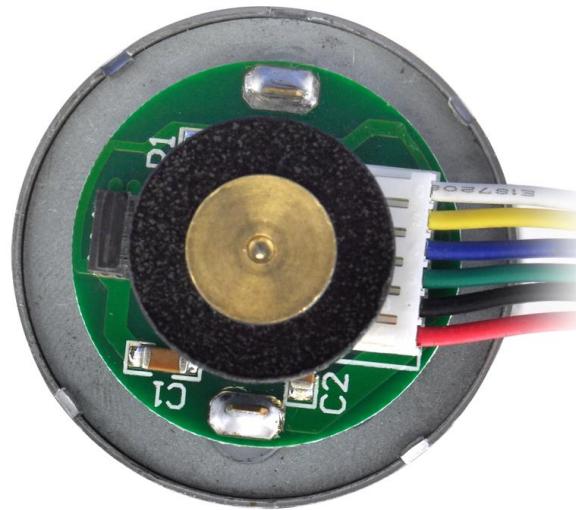


Figura 6-5 Vista en detalle del encoder del motor.

Estas conexiones corresponden a lo siguiente:

- ✓ Blanco: canal B del encoder.
- ✓ Amarillo: canal A del encoder.
- ✓ Azul: conexión para la alimentación del encoder a 5V.
- ✓ Verde: conexión a tierra para el encoder.
- ✓ Negro: alimentación del motor.
- ✓ Rojo: alimentación del motor.

Para el caso de la etapa de potencia requeríamos:

- ✓ Conexión a tierra.
- ✓ Conexión para la alimentación lógica a 5V.
- ✓ Conexión para la alimentación de potencia a 12V.
- ✓ 2 señales PWM para controlar la velocidad de cada motor.
- ✓ 4 señales digitales para la dirección de los dos motores.

Teniendo en cuenta estas necesidades, se llega a la conclusión de la necesidad de las siguientes líneas de conexión:

- ✓ Alimentación a 5V, suministrada por el Arduino Due.
- ✓ Alimentación a 12V, que servida para alimentar tanto a la etapa de potencia como al Arduino Due.
- ✓ Señal de tierra, que será la tierra que proporcione la alimentación de 12V externa.
- ✓ 4 salidas digitales por parte del Arduino Due.
- ✓ 2 señales de salida de PWM por parte del Arduino Due.
- ✓ 4 entradas digitales al Arduino Due para poder contar cada paso de los encoders y saber la dirección de giro.

## CAPÍTULO 7.

### DISEÑO DE CONTROL.

#### 7.1 Arquitectura básica de control: control PID

El PID está pensado para trabajar como compensador del error compuesto por 3 acciones a priori independientes.

La acción proporcional  $K$ , cuya función es dar consistencia al control y actúa sobre el valor presente del error.

La segunda es la acción integral, que es proporcional a la integral del error, tiene como función dotar de precisión al sistema tomando en cuenta los errores pasados.

La tercera acción es la acción derivativa, la cual es proporcional a la derivada del error, que tiene como función dar estabilidad dotar de mayor velocidad al sistema. Esta acción tiene un efecto predictivo del error en un cierto horizonte de tiempo.

Con esto se puede afirmar que el controlador PID calcula su señal de control combinando la información sobre el error actual, teniendo en cuenta los errores que se han ido arrastrando en el pasado y con una cierta capacidad de prevenir futuros errores para anticiparse a ellos.

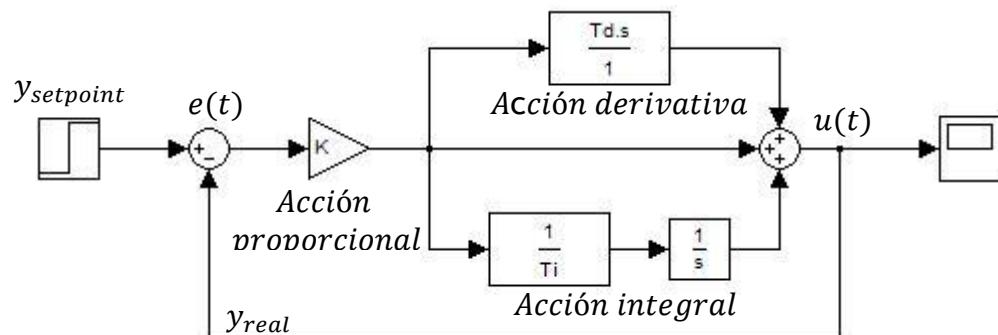


Figura 7-1 Estructura de un PID básico

La ley de control,  $u(t)$ , de un controlador PID como el de la figura 7-1 viene definida por la siguiente expresión:

$$u(t) = K \left( e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right)$$

Donde:

Error,  $e(t) = y_{setpoint} - y_{real}$ , es la diferencia entre la señal de entrada,  $y_{setpoint}$ , y la medición de la salida,  $y_{real}$ .

$K$  es la ganancia proporcional.

$T_i$  es la constante de tiempo de integración.

$T_d$  es la constante de tiempo de derivación.

Aunque muchos de los aspectos de un sistema de control se pueden entender a partir de la teoría de control lineal, algunos efectos no lineales deben ser tomados en cuenta a la hora de implementar un controlador.

Para un sistema de control con un amplio rango de condiciones de operación, puede suceder que la variable de control alcance los límites prefijados del actuador. Cuando esto pasa, el bucle realimentado permanece en su límite independientemente de la salida del proceso. Si se usa un controlador con acción integral, el error continuará siendo integrado, incrementando aún más su valor. Esto significa que el término integral puede volverse muy grande y producirse el efecto llamado “windup”.

Para solucionar esto se recurre a una técnica conocida como back-calculation & tracking. Cuando se detecta que la salida se ha saturado la integral se recalcula para que la salida del regulador coincida con la de saturación. Desde ese momento el regulador actualiza el valor de la acción integral de tal manera que se mantenga el valor de saturación, quedando un sistema final como el que aparecen la figura 7-2. Este nuevo sistema compara la salida solicitada por el PID y su valor en saturación, si son diferentes se recalcula la acción integral para no superar este valor, dejando el valor calculado si no entra en saturación.

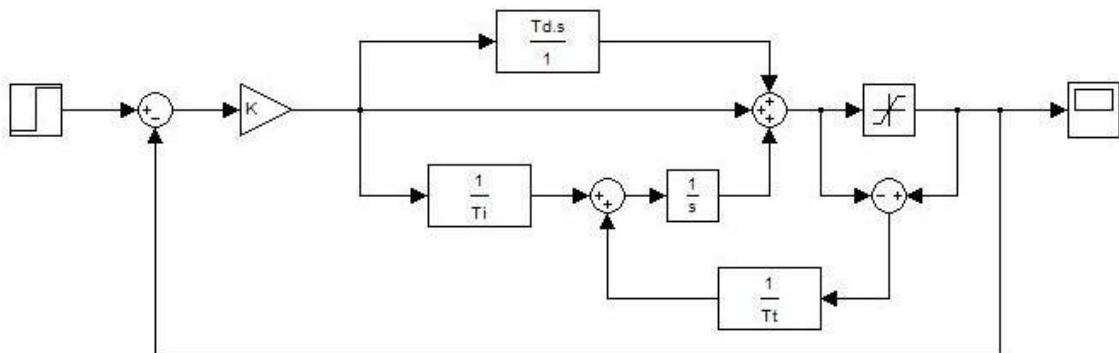


Figura 7-2 Estructura de un PID con back-calculation & tracking

Lo primero que se realizó para el control de los elementos motrices fue la obtención de una librería para PID que funcionase sobre Arduino con la característica de ser lo más robusta, completa y accesible posible. Para esto, se decidió por el desarrollo de una librería propia que garantizase todo lo anterior, con la ventaja de poder adaptarla a la perfección a las necesidades que surgiesen en el momento de usarse gracias a tener total control sobre el código.

El código que más adelante aparece se corresponde con la función más significativa del regulador PID que se ha utilizado en el control, la función Compute.

Esta función es el corazón del control, ya que tiene toda la matemática requerida para hacer un control PID con anti windup.

Esta función toma como parámetros la entrada, Input, y el punto deseado de funcionamiento, Setpoint.

Su manera de funcionar es la siguiente:

- ✓ Calcula el tiempo que ha transcurrido desde la última lectura.
- ✓ Calcula el error entre la señal de entrada y la esperada. Se calcula para todos sus componentes: P, I y D.
- ✓ Por último calcula la salida que tiene que pasársele al sistema para alcanzar el punto deseado, pasándolo previamente por el sistema anti-windup.

```
float Compute(float Setpoint, float Input)
//Función para el procesamiento del PID.
{
    _Setpoint = Setpoint;
    _Input = Input;
    if(!_inAuto) return 0.0;
//Cuanto tiempo a pasado desde el último cálculo.
    unsigned long now = millis();
    double timeChange = (double)(now - _lastTime);
    if (timeChange < 0.0) timeChange = 0;

//Calculamos todas las variables de error.
    double error = _Setpoint - _Input;
    _ITerm += (_ki * error * timeChange);

    double dErr = (error - _lastErr) / timeChange;

//Calculamos la función de salida del PID.
    _Output = _kp * error + _ITerm + _kd * dErr;

//anti-WindUp.
    if(_Output > _outMax){
        double ITermAux = _Output - _outMax;
        _ITerm = _ITerm - ITermAux;
    }

    else if(_Output < _outMin){
        double ITermAux = _outMin - _Output;
        _ITerm += ITermAux;
    }
}
```

```

    }

//Recalculamos la función de salida del PID con el nuevo _ITerm.
    _Output = _kp * (error + _ITerm + _kd * dErr);

//Guardamos el valor de algunas variables para el próximo ciclo de cálculo.
    _lastErr = error;
    _lastTime = now;

    return _Output;
}

```

## 7.2 Calculo y modelado de un motor de corriente continua.

En este apartado se pretende realizar un pequeño análisis de un motor de corriente continua. Este estudio se realizó ya que si se comprende su comportamiento y se sabe cómo va a actuar, se pueden interpretar mejor los datos recogidos por los sensores y calcular un mejor sistema de control.

Para este estudio se partió de un modelo básico para un motor de corriente continua como el que aparece en la figura 7-3

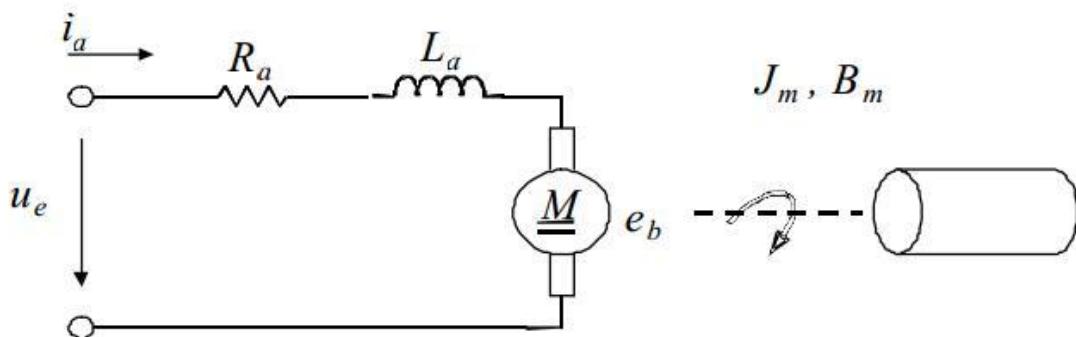


Figura 7-3 Modelo básico para un motor de corriente continua

Siendo el diagrama de bloques de la figura 7-4 el que define al modelo del motor anterior:

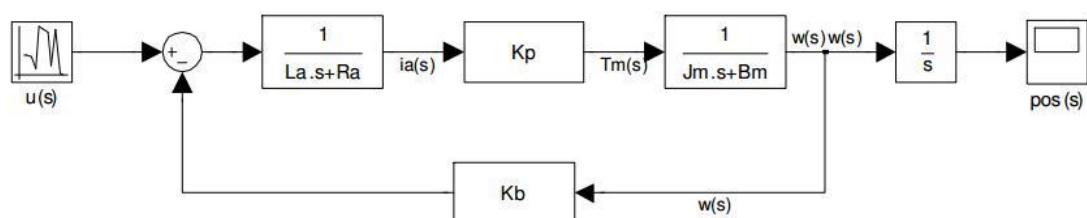


Figura 7-4 Modelado de un motor de corriente continua.

Si realizamos la transformación de diagramas e bloques la función matemática que define este sistema, la forma que toma es como la siguiente:

$$G(s) = \frac{K_1}{s^2 + s \cdot K_2 + K_3} \cdot \frac{1}{s}$$

Si analizamos el sistema resultante en función de su error en régimen permanente suponiendo que tengamos una realimentación unitaria, el cual se define por sus constantes de error, tendremos el siguiente resultado.

$$\begin{cases} K_p = \lim_{s \rightarrow 0} G(s) = 0 \\ K_v = \lim_{s \rightarrow 0} s \cdot G(s) = \frac{K_1}{K_3} \\ K_a = \lim_{s \rightarrow 0} s^2 \cdot G(s) = 0 \end{cases} \rightarrow \begin{cases} e_p = \frac{1}{1 + K_p} = 0 \\ e_v = \frac{1}{K_v} = \frac{K_3}{K_1} \\ e_a = \frac{1}{K_a} = \infty \end{cases}$$

Si tenemos en cuenta los resultados comparándolos con la tabla siguiente tabla:

Tabla 7-1

| Tipo de sistema | Constante de error |          |          | Error al escalón unitario | Error a la rampa unitaria | Error a la parábola |
|-----------------|--------------------|----------|----------|---------------------------|---------------------------|---------------------|
|                 | $K_p$              | $K_v$    | $K_a$    |                           |                           |                     |
| 0               | $K_p$              | 0        | 0        | $\frac{1}{1 + K_p}$       | $\infty$                  | $\infty$            |
| 1               | $\infty$           | $K_v$    | 0        | 0                         | $\frac{1}{K_v}$           | $\infty$            |
| 2               | $\infty$           | $\infty$ | $K_a$    | 0                         | 0                         | $\frac{1}{K_a}$     |
| 3               | $\infty$           | $\infty$ | $\infty$ | 0                         | 0                         | 0                   |

Podemos afirmar que el motor de corriente continua en un sistema de tipo 1. Esto quiere decir que su error cuando se le pide que alcance una posición específica va a ser siempre 0, mientras que si se le pide que alcance una velocidad constante siempre cometerá un cierto error, en el caso de que se le pide que mantenga una aceleración constante el sistema no será capaz de responder.

Gracias a saber cómo es el sistema y de qué manera va a responder a las diferentes señales de mando que se le apliquen, se partió con una base suficiente para poder interpretar los datos que se obtuvieron en el apartado siguiente y de cómo utilizarlos para obtener el regulador necesario.

### 7.3 Obtención del regulador PID para los motores de corriente continua.

Lo primero es comentar las herramientas a utilizar para poder entender mejor la elección de las mismas.

Simulink es un entorno de diagramas de bloque para la simulación multidominio y el diseño basado en modelos. Admite el diseño y la simulación a nivel de sistema, la generación automática de código y la prueba y verificación continuas de los sistemas embebidos.

Simulink ofrece un editor gráfico, bibliotecas de bloques personalizables y solvers para modelar y simular sistemas dinámicos. Se integra con Matlab, lo que permite incorporar algoritmos de Matlab en los modelos y exportar los resultados de la simulación a Matlab para llevar a cabo más análisis.

Esta herramienta se escogió ya que permitía analizar los datos reales del motor y su modelo. Gracias a la obtención de este modelo se pudo trabajar cómodamente con la rltool y probar diferentes combinaciones de controladores hasta obtener el más indicado para nuestro sistema real.

La siguiente herramienta que se utilizó una vez obtuvo el modelo fue la rltool.

La rltool es una herramienta de Matlab que proporciona una interfaz gráfica de usuario para el análisis del lugar de las raíces de sistemas SISO. Es solo una parte de la herramienta SISO Design Tool de Matlab.

Rltool proporciona una forma rápida, fácil y útil de diseñar controladores, y ver su influencia en el lugar de las raíces dibujado sobre el plano complejo. Con rltool podemos añadir, mover y eliminar los polos y los ceros del controlador de forma rápida, cambiar su ganancia y ver los resultados de forma inmediata en la nueva localización de los polos del sistema en bucle cerrado.

Además rltool puede dibujar la respuesta del sistema en relación al estímulo de entrada proporcionado, de manera que podemos verificar la bondad de nuestro sistema controlado.

Una vez eligieron las herramientas, la manera de obtener el regulador fue la que se detalla a continuación.

Lo primero fue la obtención del modelo del motor, para esto lo primero fue estudiar su respuesta real ante un escalón y mediante la captura de una serie de puntos de la respuesta, obtener el modelo del sistema.

Al ser un sistema de tipo 1, su respuesta es de tipo rampa, por lo que para modelar el sistema tenemos que fijarnos en dos características de la misma. Estas dos características son la pendiente y en punto en el que se corta el eje del tiempo si se dibuja una recta infinita tangente a la rampa del sistema real.

Para obtener este modelo seguimos la siguiente relación entre la respuesta real y el modelo:

$$G(s) = \frac{K_e}{s \cdot (\tau \cdot s + 1)}$$

Una vez obtenido el modelo (figura 7-5), se simuló para comprobar que era correcto y poder utilizarlo para la obtención del sistema de control. Esta comprobación se hizo comparando una nube de puntos de la grafica real con la del modelo calculado (figura 7-6).

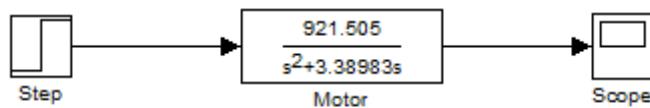


Figura 7-5 modelo del motor de corriente continua.

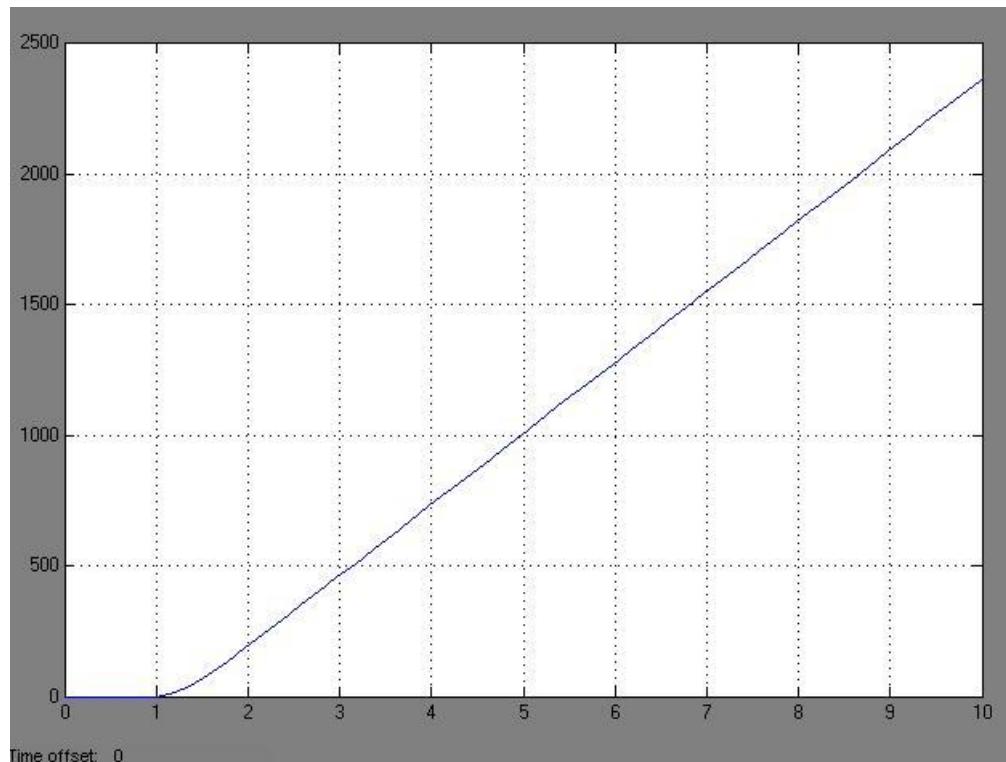
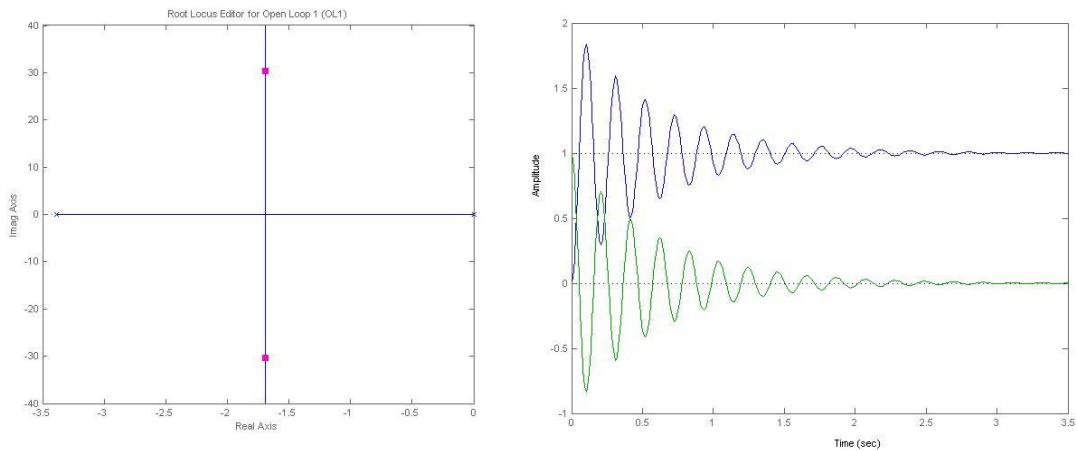


Figura 7-6 comparando una nube de puntos de la gráfica real con la del modelo calculado

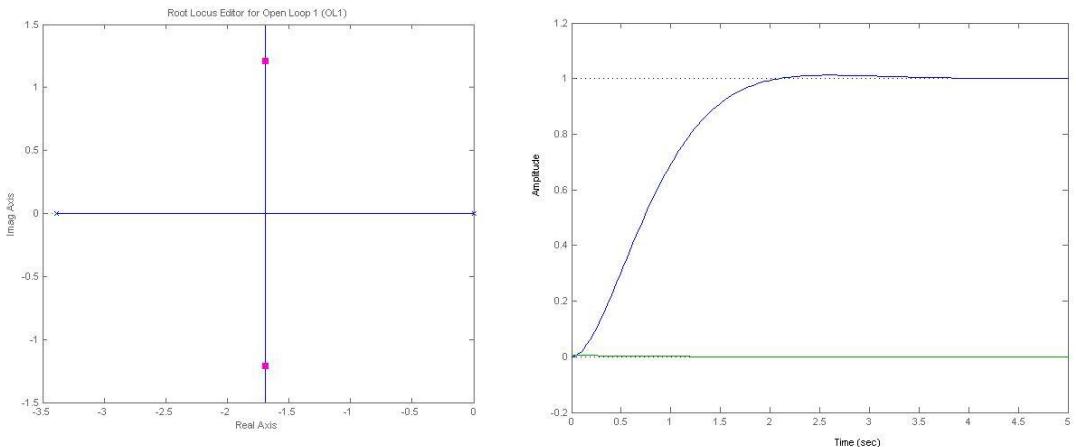
Una vez obtenido y validado el modelo, se utilizó la `rltool` para poder obtener el sistema de control y validarla al mismo tiempo gracias a la posibilidad de poder simularlo mientras se modela.

Debido a que el sistema era de tipo 1, el error en régimen permanente era 0, por lo que se optó por la implementación de un controlador proporcional.

Con una  $K = 1$ , se vio que el sistema era muy oscilatorio como se puede ver en la figura 7-7

Figura 7-7 Respuesta del sistema a un regulador  $K = 1$ 

Variando el valor de  $K$ , se llegó a la conclusión de que el valor con el que se comportaba mejor el sistema era de  $K = 0.0047$ , con el cual se obtenía un sistema rápido, preciso y que no oscilaba como se ve en la figura 7-8.

Figura 7-8 Respuesta del sistema a un regulador  $K = 0.0047$ 

Una vez se obtuvo este valor de  $K$  para el regulador, el siguiente paso fue hacer los ajustes sobre el sistema real.

## 7.4 Calculo de la posición de la plataforma en función de dos encoders.

Para poder realizar un control en posición, lo primero fue definir las ecuaciones que regían el comportamiento de la misma, así como las relaciones entre los datos de entrada al sistema y las posiciones a alcanzar.

### 7.4.1 Lectura y traducción de los encoders

Lo primero fue abordar la traducción de la lectura de los dos encoders en un punto bidimensional y un ángulo para determinar su posición y sentido.

Para ello, lo primero fue la composición de las dos señales de cada encoder para conocer no solo la distancia recorrida, sino también la dirección en la que se está avanzando.

Como se puede ver en la figura 7-9, los dos canales de cada encoder están desfasados, por lo que sabemos que cuando el canal A tiene un señal por flanco de subida, si el canal B esta en low, ha avanzado un paso en un sentido. De la misma manera, si el canal B se encuentra en high cuando hay un flanco de subida en el canal A, el motor estará girando en la dirección opuesta.

Con esta información se implementaron dos interrupciones en la placa Arduino Due por cada motor, una por cada canal, para poder medir con exactitud cada uno de los canales.

Se optó por la utilización de interrupciones ya que es la única manera de poder asegurar que no se van a perder pasos de los encoders, ya que paran por completo el ciclo de ejecución del software para atender esa petición.

La interrupción asociada a cada canal se implementó de manera diferente, en una se implementó la lectura del canal cuando hubiera un flanco de subida, para el caso del canal A, mientras que para la otra interrupción se implementó para que saltase cuando hubiera un cambio de valor en la señal de control.

En el siguiente fragmento de código podemos ver como es la inicialización de las interrupciones. Para este propósito, se necesita utilizar la función `attachInterrupt`, la cual recibe como parámetros el pin que va a controlar el lanzamiento de la interrupción, la función que se ha de ejecutar cuando se active la interrupción y, por último, como ha de cambiar la señal de control para que se lance la interrupción, que en nuestro caso usamos lanzamiento por cambio y lanzamiento por flanco de subida.



Figura 7-9 Lectura de los canales del encoder de los motores.

```
void setup_motors() {
    attachInterrupt(encoderPinB, CountB, RISING);
    attachInterrupt(encoderPinA, StateA, CHANGE);
}
```

Para el caso del canal B, el cual se controla por flanco de subida, lo que se realiza es una comprobación del estado del canal A, si el canal A esta en modo LOW quiere decir que el motor está moviéndose en la dirección que nosotros consideramos como positiva. Si por el contrario el canal A se encuentra en modo HIGH, el flanco de subida en el canal B nos indicara que se está avanzando en el sentido negativo según nuestro criterio.

```
void CountB()
{
    if (m == LOW) {
        encoderPos++;
    }
    else {
        encoderPos--;
    }
}
```

Para el caso del canal A, tan solo leemos el valor de la señal de dicho canal cuando esta cambia, capturando si es LOW o HIGH, para que cuando se produzca un flanco de subida en el canal B podamos saber en qué dirección está avanzando.

```
void StateA()
{
    m = digitalRead(encoderPinA);
}
```

Gracias a estas dos interrupciones ya tenemos la primera parte para localizar la plataforma, tener una referencia de cuanto se ha avanzado desde el momento en que se ha puesto en marcha y la dirección en la que lo ha hecho en tiempo real.

#### 7.4.2 Cálculo de la posición en función de los encoders.

Una vez obtenida la posición de los encoders, se pasó al cálculo de la posición de la plataforma, para ello se utilizan tres funciones básicas para calcular la distancia recorrida, el radio de giro y el ángulo de giro.

Lo primero fue realizar una conversión de pasos de encoders en milímetros recorridos, para hacer este cálculo, sabiendo que el encoder tiene 1600 pasos (n) por vuelta y que la rueda tiene un radio (r) de 40 milímetros, se puede calcular la distancia por paso (dp) recorrida como:

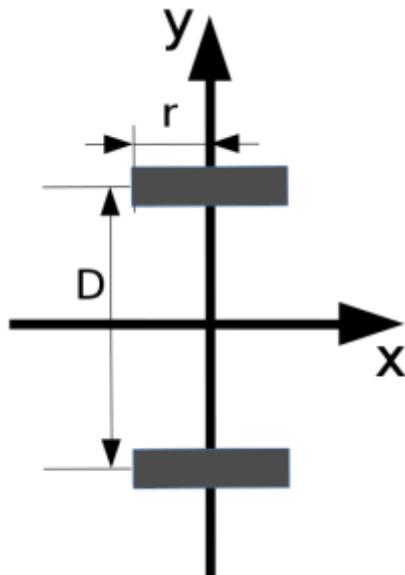


Figura 7-10 Esquema de las variables de la posición de las ruedas.

$$dp = \frac{\pi \cdot 2 \cdot r}{n}$$

A su vez, para un número de pasos (np), podemos calcular la distancia recorrida (dr) como:

$$dr = np \cdot dp$$

Para calcular la distancia recorrida ( $A_m$ ), se recurre al cálculo de la media de las distancias recorridas por cada rueda ( $A_i$ ,  $A_d$ ), ya que, sin ser la medida más exacta, es una aproximación muy fiable teniendo en cuenta que siempre va a haber errores en su cálculo derivado del deslizamiento de las ruedas.

$$A_m = \frac{A_d + A_i}{2}$$

Una vez calculada la distancia, se pasó al cálculo de la orientación, para el cual necesitábamos saber el ángulo girado y el radio de giro. Esto sólo se calcula si las distancias recorridas por las ruedas no son iguales o su diferencia se encuentra dentro de un margen de referencia.

Para el caso del cálculo del incremento del ángulo de giro ( $\Delta\theta$ ), se puede recurrir a la función que relaciona la distancia recorrida por cada rueda ( $A_i$ ,  $A_d$ ) y la separación entre ambas ( $D$ ) con el ángulo girado.

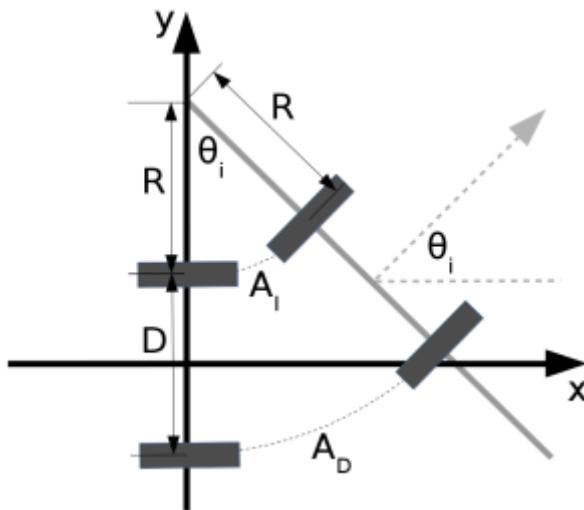


Figura 7-11 Esquema de las variables que intervienen en el cálculo del giro.

$$\Delta\theta = \frac{A_i - A_d}{D}$$

Este ángulo toma como referencia positiva el lado izquierdo de la plataforma y como un ángulo de giro negativo la parte derecha de la plataforma.

Por último, nos queda determinar el radio de giro ( $R$ ) para lo cual partimos de la distancia media ( $A_m$ ) que se recorre con un movimiento circular.

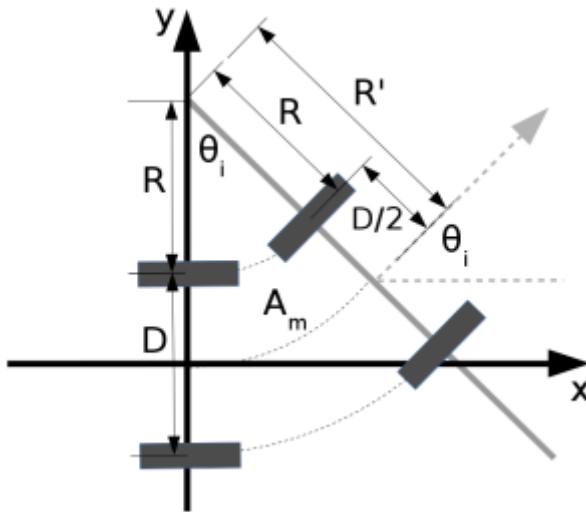


Figura 7-12 Esquema de las variables que intervienen en el cálculo del giro detalladas.

$$\begin{cases} A_m = R' \cdot \Delta\theta = \left(R \cdot \frac{D}{2}\right) \cdot \Delta\theta \\ A_m = \frac{A_d + A_i}{2} \end{cases}$$

$$R' = \frac{D}{2} \cdot \frac{A_d - A_i}{A_d + A_i}$$

Se toma el valor absoluto ya que la dirección de giro ya lo marca el signo del ángulo.

Ya con estas ecuaciones podemos definir cómo se va desplazando la plataforma cada cierto periodo de tiempo y calculando su trayectoria como la composición de lo ocurrido entre los periodos de tiempo de muestreo.

Si el movimiento en el periodo de tiempo desde el último cálculo da como resultado que las dos ruedas han avanzado lo mismo, se calculará la distancia como la media de ambas y el ángulo de giro como el ángulo de la etapa anterior.

En el caso de que el desplazamiento de la plataforma en el último ciclo no sea rectilíneo, se calculará tanto el ángulo como el radio de giro con las dos ecuaciones anteriores.

Todo esto nos permite realizar un cálculo espacial de las coordenadas cartesianas en las que se encuentra y poder calcular cuánto se ha desviado de las coordenadas reales para poder corregir su trayectoria.

Al ser el cálculo de posición anterior de carácter relativo, lo que se calcula en un primer momento son las coordenadas relativas que tiene la plataforma desde su anterior ciclo.

En el caso de que el movimiento sea rectilíneo, solo hace falta descomponer la distancia recorrida en sus componentes  $\Delta x$  e  $\Delta y$  utilizando el ángulo absoluto de la plataforma.

$$\begin{cases} \Delta x = A_m \cdot \sin(\theta) \\ \Delta y = A_m \cdot \cos(\theta) \end{cases}$$

Para el caso contrario, el cálculo de posición que se utiliza en el caso de movimiento no sea rectilíneo tenemos que tener en cuenta también el radio de giro.

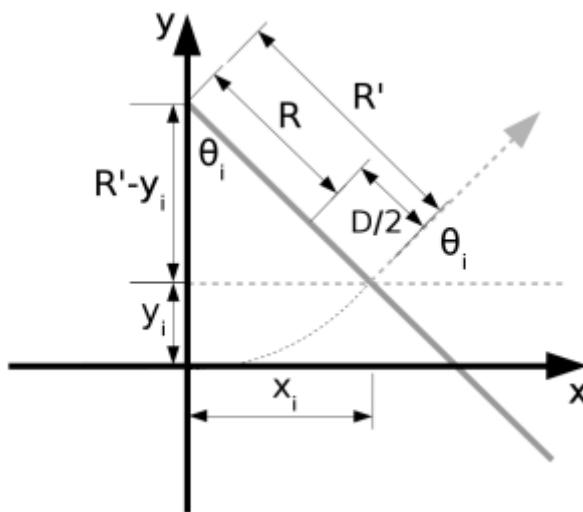


Figura 7-13 Esquema de las variables que intervienen en el incremento de posición.

$$\begin{cases} x_i = R' \cdot \sin(\Delta\theta) \\ R' - y_i = R' \cdot \cos(\Delta\theta) \end{cases}$$

Por lo que, para conocer la posición absoluta solo tendríamos que sumar este incremento a la posición actual.

$$\begin{cases} \Delta x = R' \cdot (\sin(\Delta\theta + \theta) - \sin(\theta)) \\ \Delta y = R' \cdot (\cos(\theta) - \cos(\Delta\theta + \theta)) \end{cases}$$

Aunque estos incrementos son una aproximación, debido a que se calcula por tramos, la ventana de tiempo entre adquisiciones es inferior al segundo por lo que, unido a que la

velocidad de la plataforma no es muy elevada, el error cometido es prácticamente despreciable.

Al final, lo que se consigue es un modelo de control que permite trazar trayectorias punto a punto desde el punto de origen como el que aparece en la figura 7-14.

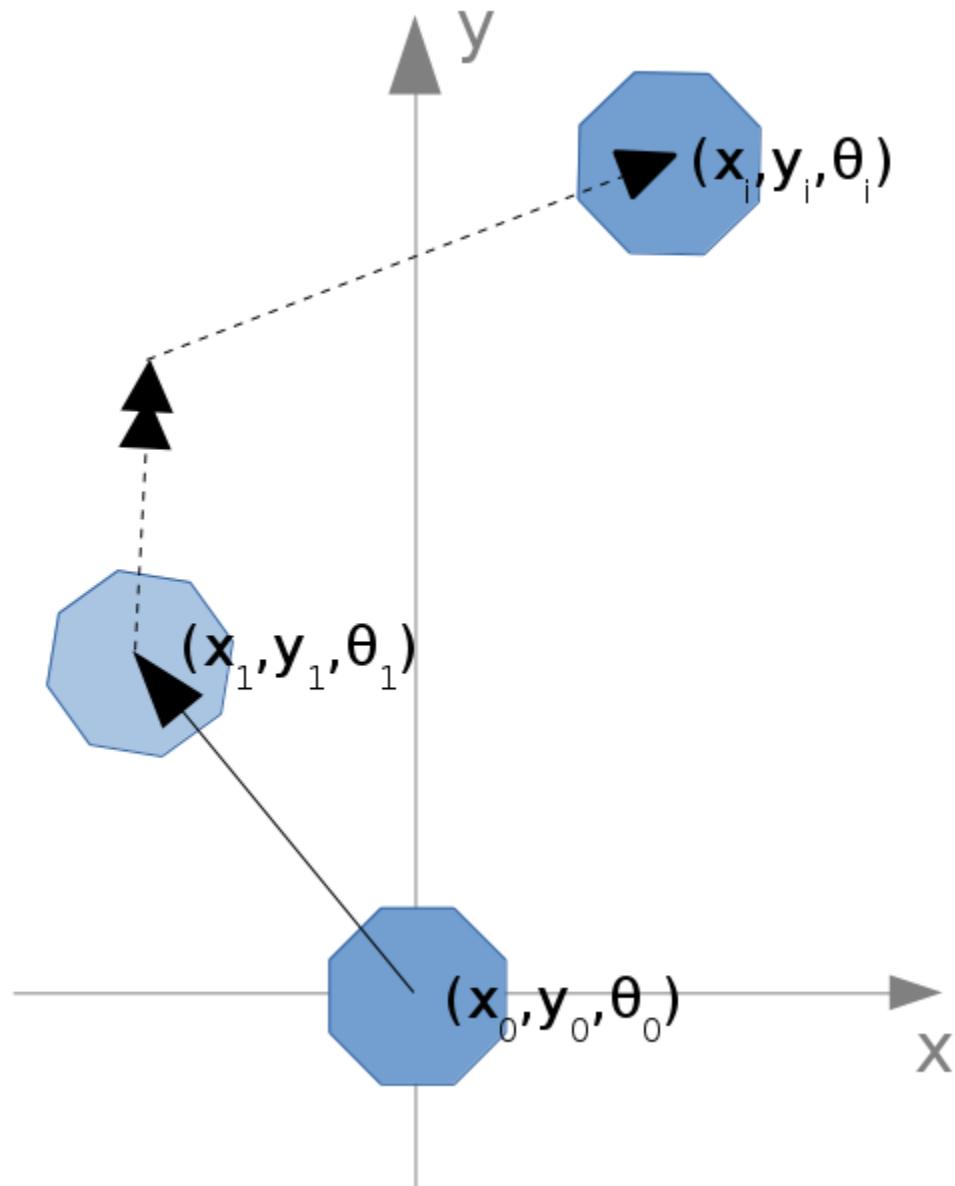


Figura 7-14 Esquema del trazado de una trayectoria punto a punto.

## CAPÍTULO 8.

### PRUEBAS EXPERIMENTALES.

---

Para validar el correcto funcionamiento de la plataforma se diseñaron cuatro pruebas para comprobarlo.

Estas pruebas fueron las siguientes:

- ✓ Movimiento de un metro de longitud en el eje X, punto (1m, 0).
- ✓ Movimiento de un metro de longitud en los ejes X e Y, punto (1m, 1m).
- ✓ Movimiento de dos pasos consecutivos al punto (1m, 0) y luego al (1m, 1m)

#### 8.1 Movimiento de un metro en el eje X.

Esta fue la primera prueba, ya que se pretendía conseguir el movimiento más simple posible.

En esta prueba solo se tenía en cuenta la distancia, ya que el ángulo de partida y el de llegada eran en mismo.

Teniendo en cuenta nuestra plataforma de pruebas, el movimiento deseado era como el que aparece en la figura 8-1.

En esta figura podemos ver como la plataforma solo ha de desplazarse una distancia igual a 1000mm en línea recta.

Esta prueba sirvió para validar tres cosas básicas.

- ✓ Se realizó una validación de la lectura de los encoders, ya que son el único punto de referencia actual y, de no funcionar correctamente no seguiría una trayectoria rectilínea.
- ✓ Se validó también la resolución de los encoders, ya que si el desplazamiento era mayor o menor de 1000mm, teniendo en cuenta los errores por deslizamiento, esto indicaría que no se han tomado correctamente los valores de la resolución.
- ✓ Por último se validó el sistema de control de trayectoria, ya que si funcionando bien los encoders la plataforma no terminaba en la posición adecuada, todo indicaría un fallo del sistema de control.

Teniendo todo esto en cuenta, esta primera prueba se superó sin muchas complicaciones, encontrándose en ella un error de implementación mínimo.

Como se puede ver en las figuras 8-2 y 8-3, la trayectoria trazada por la plataforma es casi perfecta, por lo que esta prueba se dio por satisfactoria.

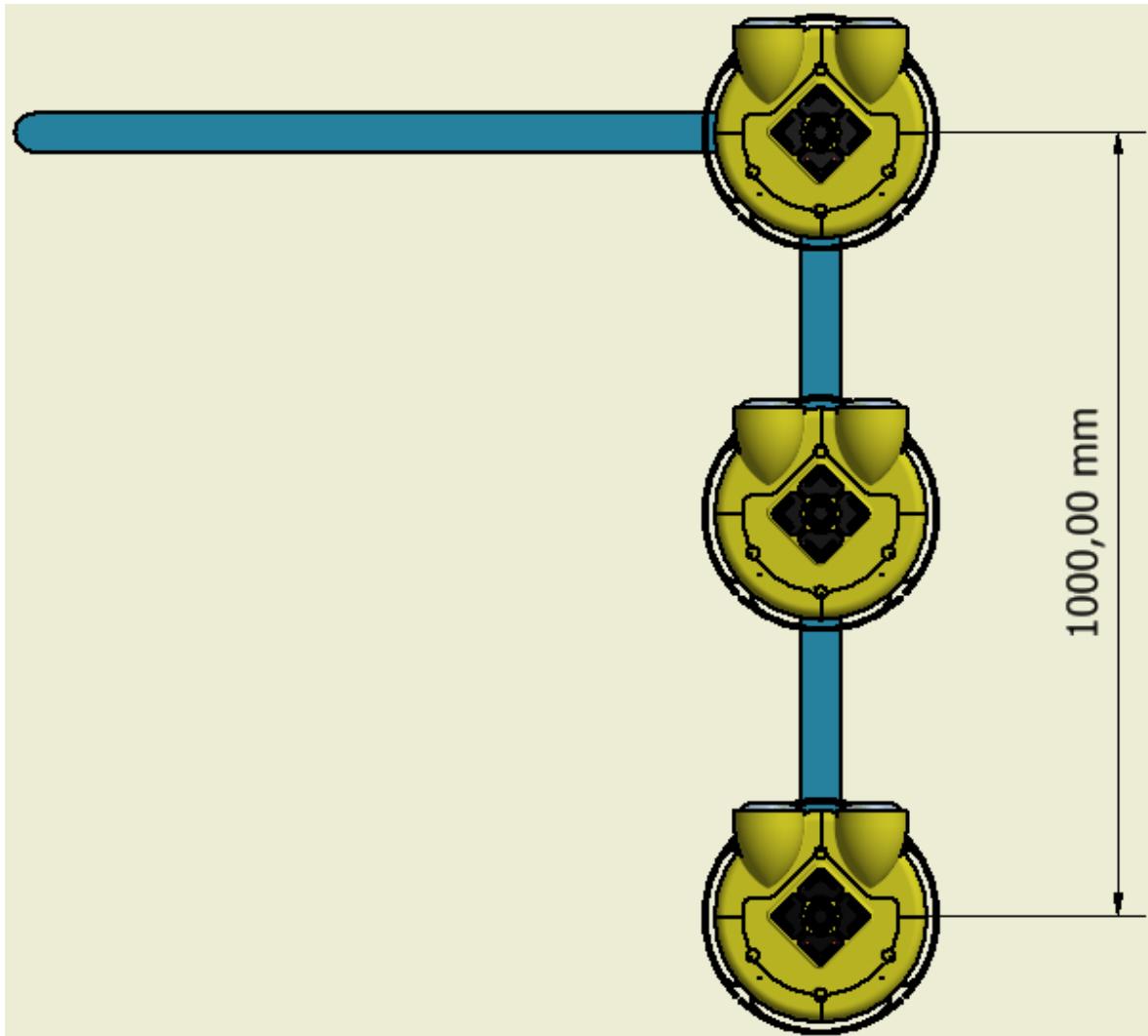


Figura 8-1 Recorrido teórico de la plataforma en la prueba 1.

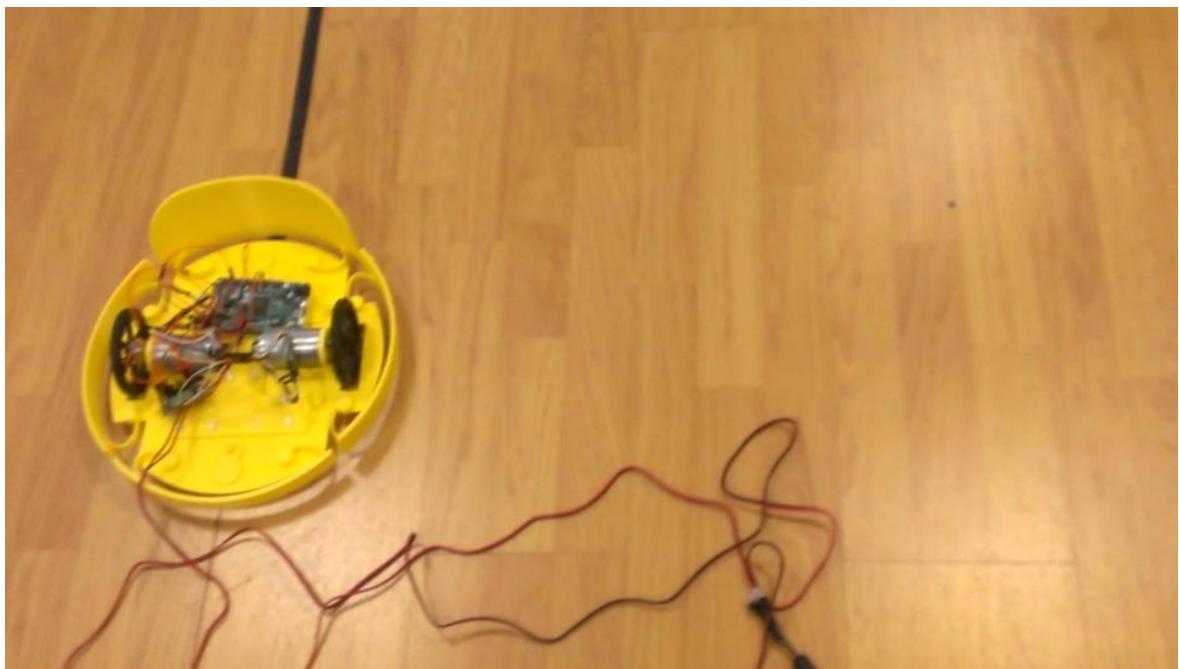


Figura 8-2 Imagen de la plataforma al inicio de la prueba 1.

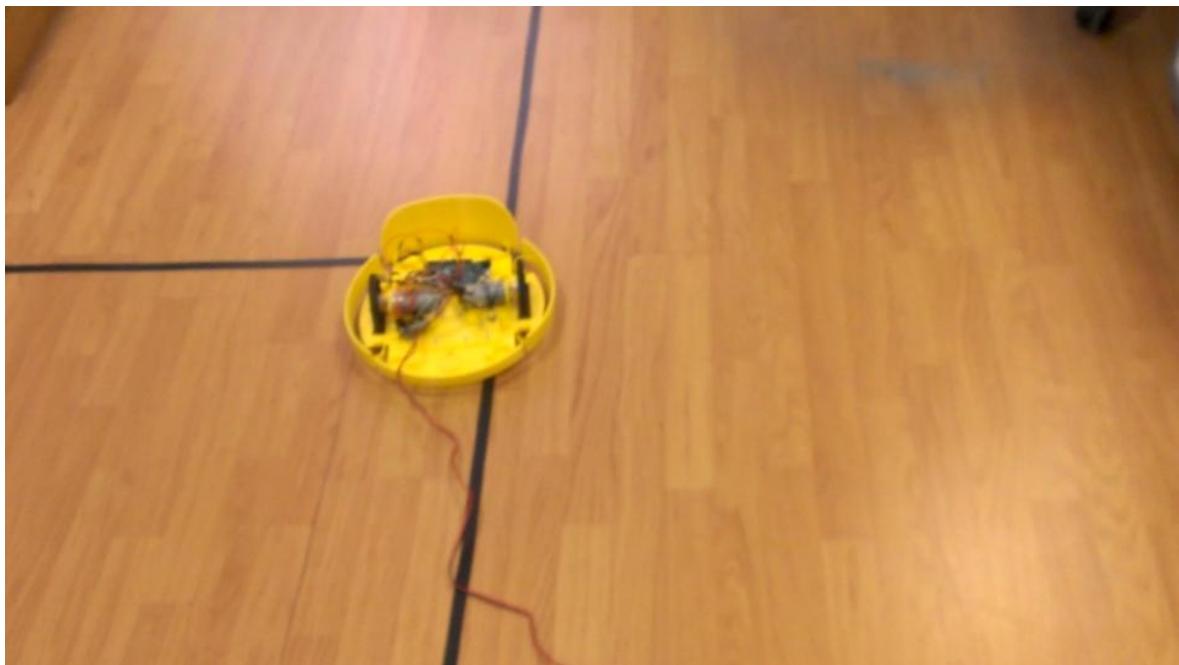


Figura 8-3 Imagen de la plataforma al final de la prueba 1.

## 8.2 Movimiento de un metro en los ejes X e Y

En esta prueba, lo que se pretendía era validar el movimiento con una trayectoria combinada en los dos ejes de coordenadas, para ello se planteó que la plataforma alcanzase la posición  $p(1m, 1m)$ .

Esta fue el primer experimento en el que se ponía a prueba el sistema de control al completo, ya que tenía que combinar los dos movimientos de las pruebas anteriores en uno y realizar la trayectoria más eficiente.

En la figura 8-4 podemos ver cuáles eran los resultados esperados para esta prueba.

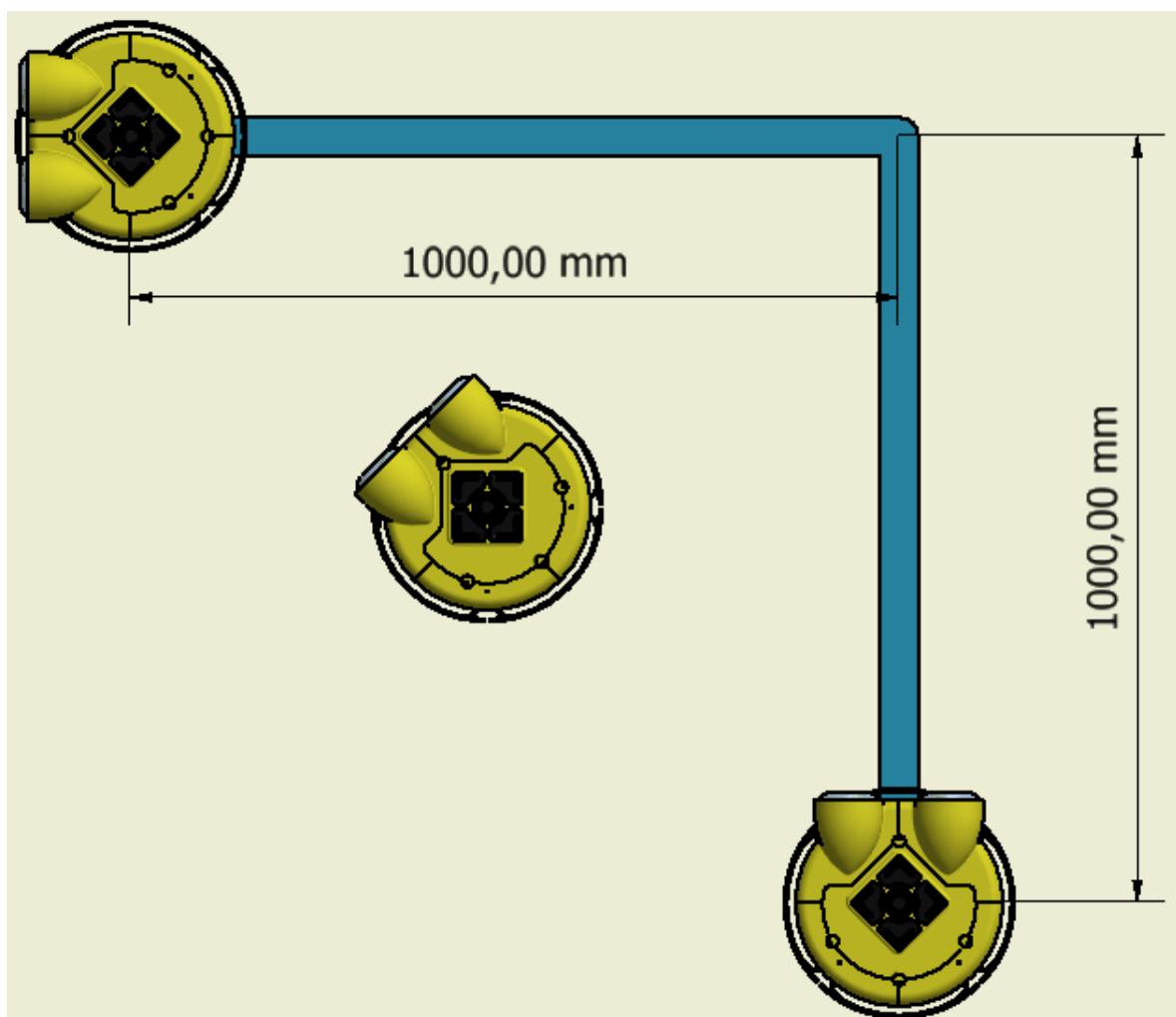


Figura 8-4 Recorrido teórico de la plataforma en la prueba 2.

Como se puede ver en las figuras 8-5, 8-6 y 8-7, la trayectoria trazada por la plataforma es casi perfecta muy similar a la teórica, por lo que el resultado de este experimento se consideró un éxito también.



Figura 8-5 Imagen de la plataforma al inicio de la prueba 2.

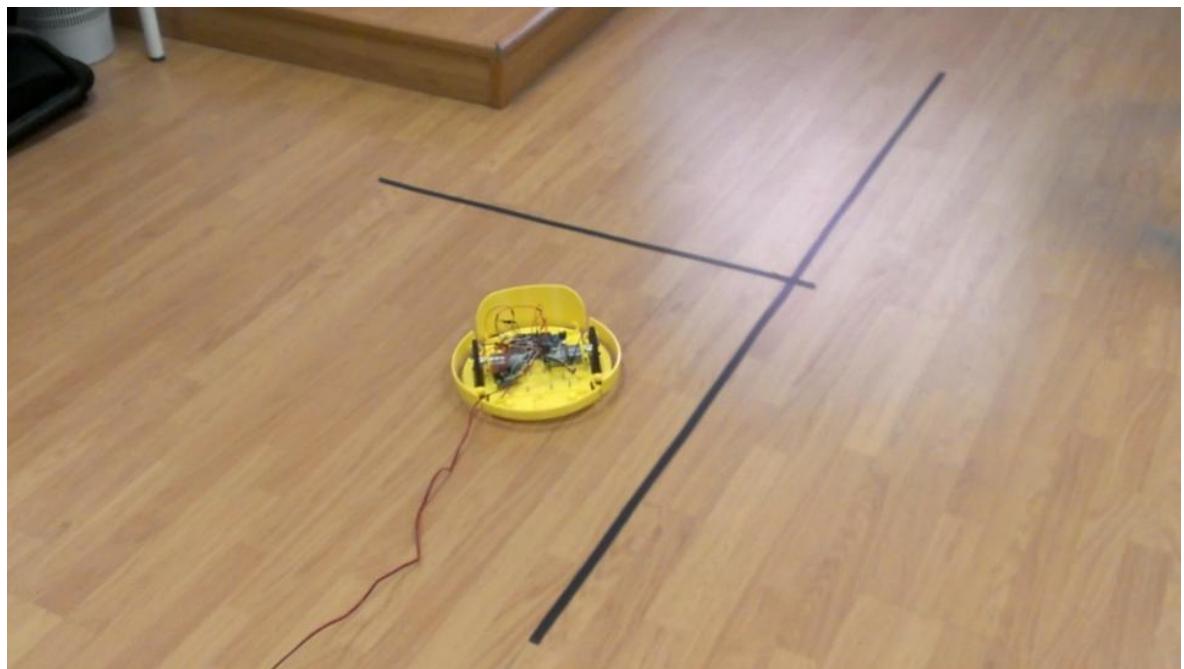


Figura 8-6 Imagen de la plataforma a la mitad de la prueba 2.



*Figura 8-7 Imagen de la plataforma al final de la prueba 2.*

### 8.3 Movimiento compuesto por dos pasos de 1 metro.

Como prueba final, se planteó un experimento para probar la capacidad de la plataforma para seguir trayectorias compuestas por varios pasos.

Para poder validar esta trayectoria se tomo como pasos  $p(1m, 0m)$  y  $p(0m, 1m)$ . Primer intentaría alcanzar la posición  $p(1m, 0m)$  y posteriormente la posición  $p(1m, 1m)$ .

El movimiento teórico deseado lo podemos ver en la figura 8-8.

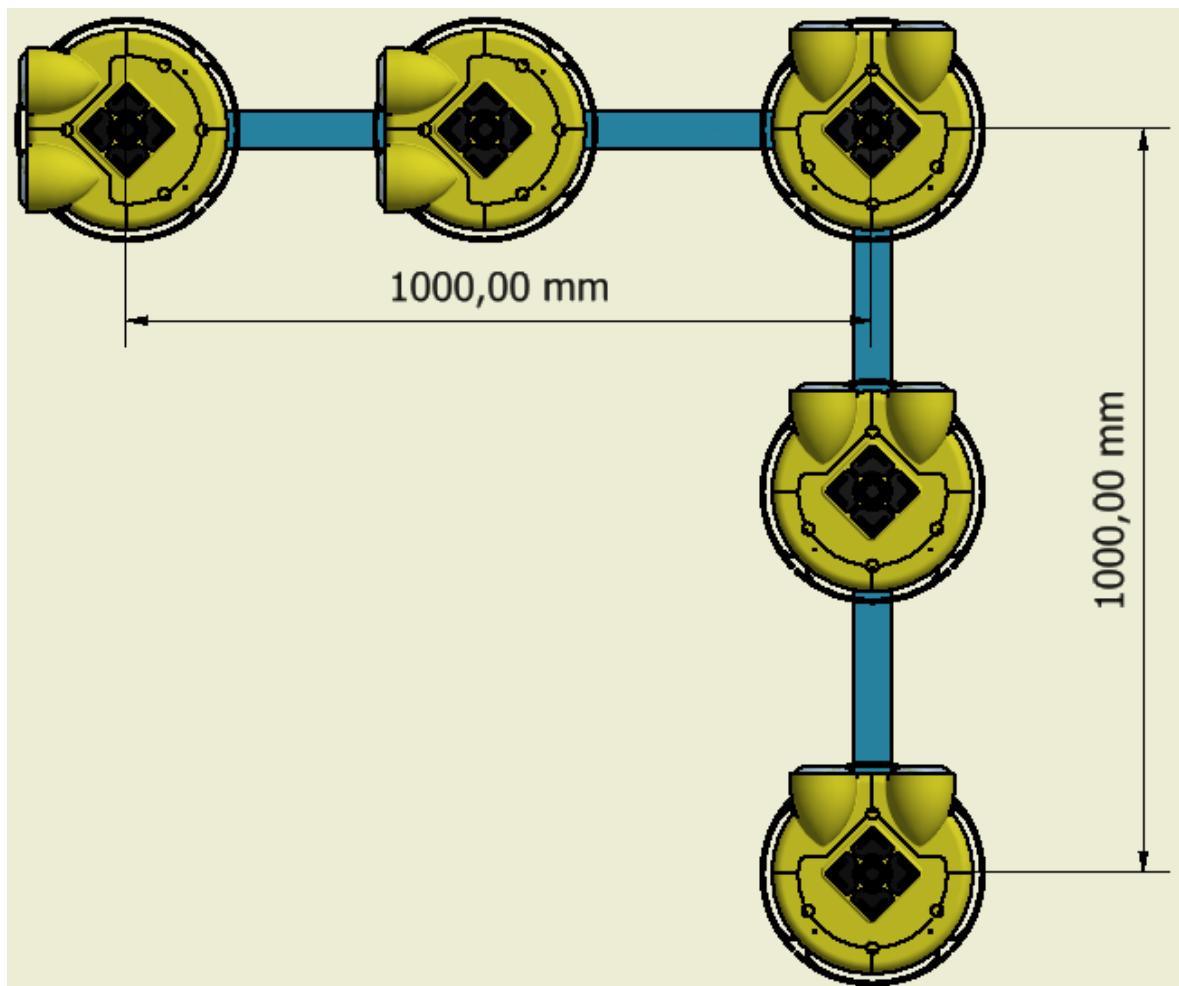
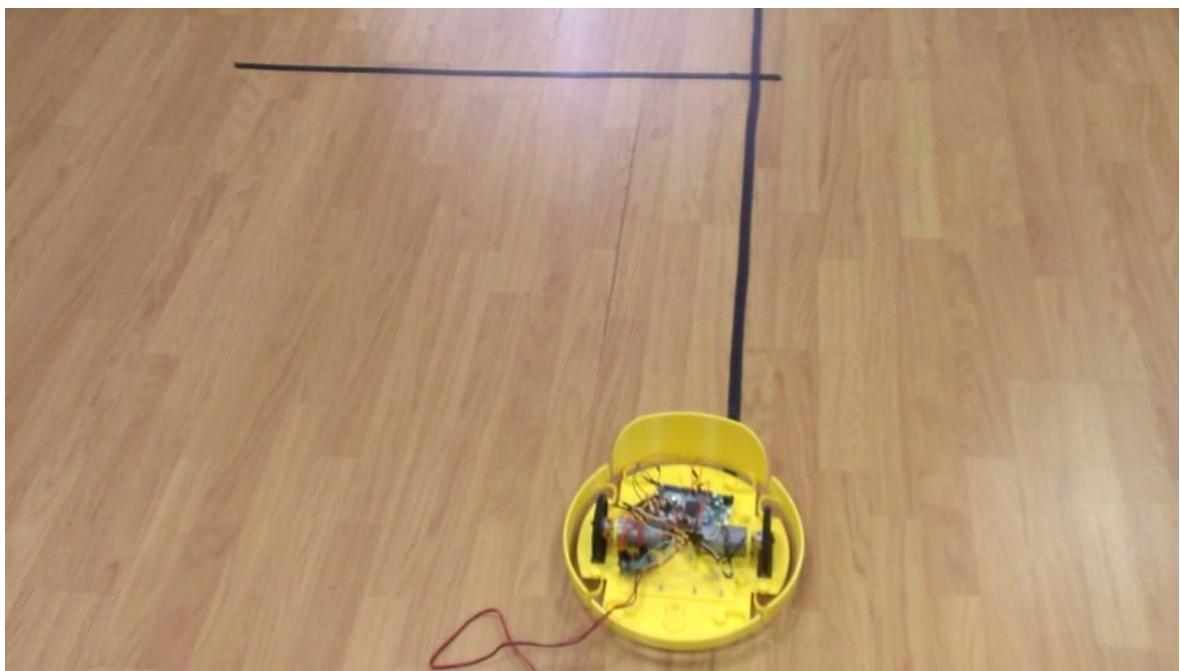
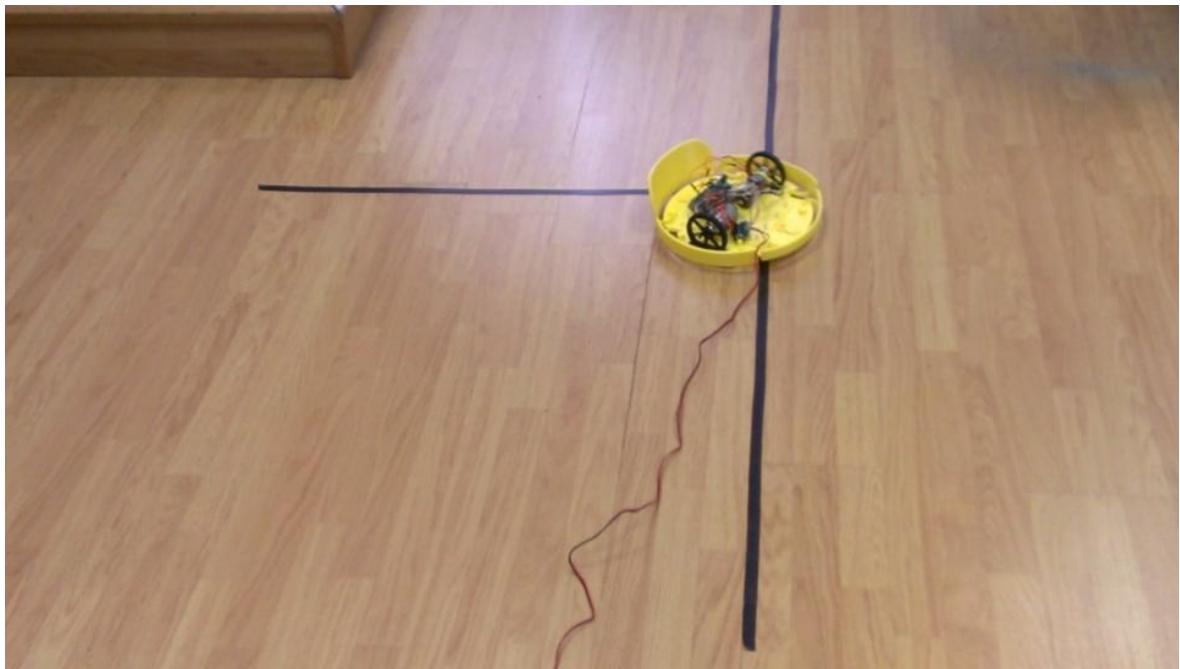


Figura 8-8 Recorrido teórico de la plataforma en la prueba 3.

Como hicimos en los anteriores experimentos, comparándolo con las imágenes tomadas en la prueba real (figuras 8-9, 8-10 y 8-11), se puede ver que el resultado es muy similar al teórico.



*Figura 8-9 Imagen de la plataforma al inicio de la prueba 3.*



*Figura 8-10 Imagen de la plataforma a la mitad de la prueba 3.*

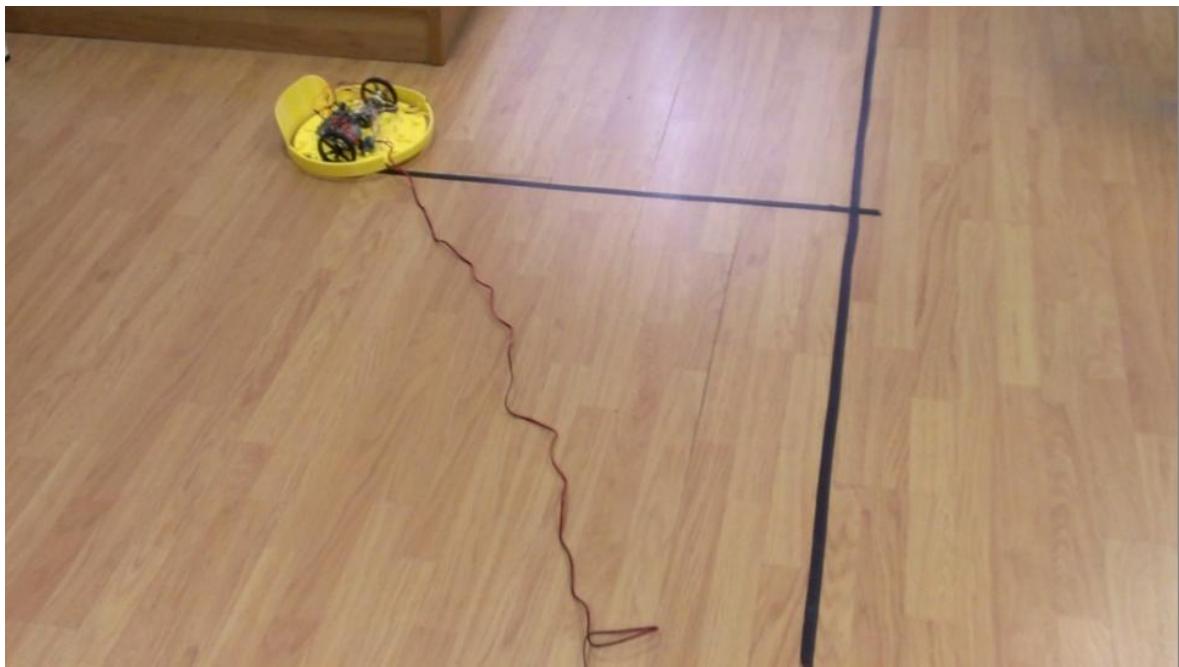


Figura 8-11 Imagen de la plataforma al final de la prueba 3.

En esta prueba, si que cabe destacar que, debido a la superficie en la que se han realizado las pruebas, el deslizamiento que se produce en el giro de 90 grados al intentar alcanzar la posición  $p(1m, 1m)$  hace que este punto no se alcance con total precisión.

Este problema es debido a que la posición de la plataforma solo se calcula mediante la posición de los encoders.

Una solución a este problema se expondrá en el siguiente capítulo, conclusiones, cuando se expongan las posibles futuras mejoras que debería sufrir la actual plataforma.

## CAPÍTULO 9.

### CONCLUSIONES.

---

La realización de este proyecto comenzó en febrero de 2015 y termina en julio de 2015.

El desarrollo se puede separar en dos partes que, en muchos casos, estuvieron solapadas. El periodo de aprendizaje ocupó la mayor parte del tiempo, llegando a ser más del 70% del tiempo dedicado. Esto no quiere decir que el tiempo de desarrollo solo fuese del 30%, ya que gran parte del tiempo de aprendizaje fue aplicado al propio diseño, llegando a ocupar más del 50% del tiempo dedicado en el desarrollo del proyecto.

Una vez terminado el proyecto y analizado los resultados, se ha podido comprobar que los objetivos propuestos a principio de éste se han cumplido de manera satisfactoria, ya que el fin último de este proyecto era el desarrollo de una plataforma robótica replicable, abierta y fácil de modificar y controlar.

Aunque los datos obtenidos con las pruebas de la plataforma han sido buenos, hay ciertas consideraciones y posibles mejoras que se podrían aplicar al diseño de ésta que se exponen a continuación.

- ✓ La primera mejora que se podría plantear es la mejora del diseño para minimizar las horas de impresión, ya que actualmente ascienden a más de 50h por plataforma.
- ✓ La segunda mejora que previsiblemente se abordara en los próximos meses a la consecución de este proyecto, es la incorporación de una unidad de medición inercial que cuente con acelerómetros en, al menos, dos ejes y un giróscopo. Esto permitiría minimizar el error de los deslizamientos ya que se podrían contrastar la información de fuerzas y giro realizado con la información obtenida de los encoders, obteniendo una medida mucho más precisa. Aunque es una mejora importante requiere también una inversión económica importante, que encarecería la plataforma en un 30%, por lo que para una primera aproximación a la robótica por parte de los niños no es algo fundamental.

Aun con estas mejoras propuestas, basándonos en las pruebas realizadas, el prototipo de la plataforma que se ha desarrollado es un magnífico punto de partida para la introducción de los niños en el mundo de la robótica, ya que es totalmente replicable y abierto, dándole todas las herramientas para que puedan modificarla y aprender cómo está diseñada y construida.

Como conclusión final solo queda añadir que se ha conseguido el objetivo principal que se exponía el primer punto de este proyecto:

*"abordar el diseño y desarrollo de una plataforma de robótica educativa, replicable y abierta."*

---

## CAPÍTULO 10.

### BLIBIOGRAFÍA.

---

- BARRIENTOS, Antonio. *Fundamentos de Robótica*. Luis Felipe Peñín, Carlos Balaguer, Rafael Aracil. Universidad Politécnica de Madrid. McGraw-Hill Aravaca (Madrid) 1997.
- RODRIGUEZ-LOSADA, Diego. HERNANDO, Miguel. *Diseño y Programación Orientados a Objetos. Una Visión Aplicada*. Universidad Politécnica de Madrid, ELAI. Madrid, 2009.
- HERNANDO GUTIÉRREZ, Miguel. *Apuntes de Informática Industrial: Programación c++*. Universidad Politécnica de Madrid, ELAI. Madrid, 2006.
- PLATERO DUEÑAS, Carlos. *Apuntes de Regulación Automática I*. Universidad Politécnica de Madrid, ELAI. Madrid, 2009.
- PLATERO DUEÑAS, Carlos. HERNANDO, Miguel. *Apuntes de Regulación Automática II*. Universidad Politécnica de Madrid, ELAI. Madrid, 2009.
- GARCÍA DE JALÓN, Javier. RODRÍGUEZ, José Ignacio. VIDAL Jesús. *Aprende Matlab 7.0 como si estuvieras en primero*. Universidad Politécnica de Madrid, ETSII. Madrid, 2005.
- MUÑOZ MENCÍA, Patricia. *Swarm-robots: sonorización y visión*. Universidad Politécnica de Madrid. Escuela Universitaria de Ingeniería Técnica Industrial (EUITI) Proyecto de fin de carrera. Madrid, 2012.
- HERRERO DE CAMPOS, Alejandro. *Desarrollo de un sistema de control para robots móviles usando información en dos y tres dimensiones*. Universidad Politécnica de Madrid. Escuela Universitaria de Ingeniería Técnica Industrial (EUITI) Proyecto de fin de carrera. Madrid, 2012.
- GARCÍA SAURA, Carlos. *Robots educativos libres, imprimibles y de bajo coste*. Universidad Autónoma de Madrid. Madrid, 2012