Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería Industrial

Proyecto Fin de Carrera

CONTROL REMOTO DE UN ROBOT IMPRIMIBLE DE EXTERIORES

Autor: Julián Marín Mato

Tutor: Alberto Valero Gómez

Director: Juan González Gómez

Leganés, Diciembre 2011

Título: Control remoto de un robot imprimible de exteriores

Asunto: Memoria del Proyecto Fin de Carrera de Ingeniería Industrial

Autor: Julián Marín Mato

Tutor: Alberto Valero Gómez

Director: Juan González Gómez

Universidad Carlos III de Madrid

Campus de Leganés

PROYECTO FIN DE CARRERA

Departamento de Ingeniería de Sistemas y Automática

Universidad Carlos III de Madrid

Título: Control remoto de un robot imprimible de exteriores

Autor: Julián Marín Mato

Tutor: Alberto Valero Gómez

Director: Juan González Gómez

La lectura y defensa del presente Proyecto Fin de Carrera se realizo el día 19 de Diciembre de 2011 bajo el tribunal:

- **Presidente:**
- **Secretario:**
- **Vocal:**

Habiendo obtenido la calificación de:

Presidente **Secretario** **Vocal**

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Preámbulo

Este proyecto forma parte de una serie de proyectos que comienzan con la idea de crear pequeños robots móviles cuyas piezas puedan ser impresas con la impresora 3D MakerBot Thing-O-Matic [18] del departamento de Sistemas y Automática de la Universidad Carlos III de Madrid.

Con dicho objetivo, un grupo de alumnos del grado de ingeniería bajo la supervisión de Alberto Valero (profesor titular de la UC3M) diseñan y construyen a lo largo de 2011 los robots móviles imprimibles F-Track y Orugator.

Llegados a este punto se tiene la estructura del robot junto a sus actuadores totalmente montados, pero se carece de un sistema de control que permita manejarlos. Es exactamente aquí donde el presente proyecto toma razón de ser, marcándose su objetivo como el de proporcionar a los robots dicho sistema de control, y diseñar y desarrollar una aplicación que permita teleoperarlos a través de un periférico sencillo e intuitivo como es el *gamepad*.

La elección del *gamepad* como medio de control va a ser de gran relevancia, y se justifica por el gran éxito que han tenido las videoconsolas en la última década, logrando que el manejo de dicho periférico pase a ser de dominio público. Esto implica que prácticamente a cualquier persona, independientemente de los conocimientos previos sobre la materia de estudio de esta tesis, le resulte familiar, fácil e intuitivo el control del robot.

Por tanto este proyecto va a constituir una parte fundamental en el desarrollo de robots imprimibles, por proporcionar una herramienta muy potente y visual para el estudio y evaluación de los PrintBots (Printable Robots, robots imprimibles), así como para la mejora de su diseño. También va a ser idóneo para futuras líneas de trabajo como podrían ser dotar al robot de autonomía, ya que proporciona al investigador una rápida forma de familiarizarse con sus movimientos y de visualizar las limitaciones del diseño.

1.2 Marco del Proyecto

Este proyecto ha sido desarrollado en el departamento de Ingeniería de Sistemas y Automática de la Escuela Politécnica Superior de la Universidad Carlos III de Madrid como proyecto final de carrera de la titulación de Ingeniería Industrial en la especialidad de Automática y Electrónica Industrial.

1.3 Principales objetivos

El objetivo fundamental de la tesis es lograr que el robot móvil imprimible F-Track pueda ser controlado remotamente mediante el empleo de un *gamepad*. En base a ese objetivo principal, se proponen los siguientes objetivos parciales:

- Dotar al robot móvil de los medios necesarios para que pueda llevar a cabo todos los movimientos para los que fue diseñado. Es decir, la estructura y diseño del robot es dada, pero carece de la electrónica y actuadores los cual deberán ser añadidos y configurados
- Proporcionar al robot móvil los medios necesarios para que pueda ser teleoperado. El robot carece de cualquier sistema de control, por lo que habrá que dotarle de alguna placa microcontroladora que gestione los movimientos de los actuadores y permita comunicarse con un PC.
- Diseñar, desarrollar y evaluar una aplicación capaz de teleoperar el robot móvil mediante el empleo de un *gamepad*.
- Diseñar de tal manera la aplicación objetivo de la tesis, que permita fácilmente teleoperar robots móviles basados en la misma arquitectura de control, sin la necesidad de diseñar una nueva aplicación específica para cada nueva máquina. Es decir, la aplicación desarrollada a lo largo de este documento, debe poder ser reutilizada para otros robots móviles imprimibles tales como el Orugator, gracias al desarrollo del software mediante el empleo de modulos de programación RDK.

1.4 Fases de desarrollo

En este apartado se explica brevemente los pasos seguidos para llevar a cabo todo el proyecto.

Investigación inicial: Previamente incluso a la planificación de este proyecto, se ha realizado un estudio sobre el estado actual de desarrollo de robots móviles y aplicaciones para el control de estos tales como el software modular OpenRDK.

Formación: Para llevar a cabo esta tesis ha sido necesario afianzar y ampliar los conocimientos, ya adquiridos a lo largo de los diferentes cursos de la titulación de Ingeniería Industrial, sobre lenguaje de programación C y sobre microcontroladores y su correcta programación.

También ha sido necesario iniciarse en programación orientada a objetos típica de C++ y adquirir unos conocimientos básicos sobre RDK.

Programación placa SkyMega1.0: En esta fase se procedió al estudio, desarrollo y evaluación de diferentes programas para el microcontrolador ATmega168/V de Atmel instalado en la placa SkyMega1.0 con el objetivo de familiarizarse y hacer lo más simple y eficiente el control de los servomotores y la comunicación a través del puerto serie.

Como resultado de esta fase se obtuvo el microcontrolador correctamente configurado para la recepción de datos a través del puerto serie, y en función de dichos valores realizar el control de los servomotores.

Selección periférico para el control del robot móvil: Se lleva a cabo la búsqueda del periférico más adecuado que permita un completo, intuitivo y fácil control del robot y de todos sus posibles movimientos.

Esta fase termina con la elección de un *gamepad* o periférico para videojuegos compuesto por dos *joysticks* y 12 botones.

Desarrollo software: En esta fase se realiza un breve diseño previo de la aplicación para un posterior desarrollo del software empleando C++ y programación modular RDK.

Pruebas: Pruebas de funcionamiento con el robot F-Track y el robot Orugator en medios favorables y adversos (con y sin obstáculos), para la posterior corrección, depuración y mejora del software a partir de los resultados arrojados por los ensayos.

Redacción de la memoria del proyecto: Tras conseguir los hitos anteriores y tras haber realizado las pruebas y analizar los resultados. Se obtuvieron las conclusiones, se

anotaron las posibles líneas futuras de desarrollo y se procedió a la redacción de la presente tesis.

1.5 Medios Empleados

Los medios con los que se ha contado para llevar a cabo este proyecto son:

- Placa SkyMega1.0 equipada con microcontrolador ATmega168V de Atmel.
- Cable de descarga FTDI TTL-232R-5V o cable USB-serie. Sirve tanto para descargar software a la placa SkyMega1.0 como para controlar los robots desde el PC.
- Adaptador AC/DC para alimentación de la placa SkyMega1.0 y de los actuadores del robot desde red eléctrica.
  Características: INPUT: 100-240V AC 0.4A 50-60Hz
  OUTPUT: 5V DC 1500M
- Servomotores FUTABA S3003.
- Robot móvil imprimible *F-Track* y robot móvil imprimible *Orugator*.
- *Gamepad* "Logitech Dual Action" compuesto por 12 botones digitales, dos *joysticks* analógicos y un *pad* octodireccional.
- Ordenador con sistema operativo Ubuntu.
- Software OpenRDK.

1.6 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo….

El primer capítulo, denominado "Introducción y objetivos" consta de un breve preámbulo como iniciación al proyecto seguido del marco del proyecto y de los objetivos principales que se buscan con la realización de este. También se explican las fases a seguir para su desarrollo y los medios empleados para tal fin. Dicho capítulo termina con el presente apartado, en el que se explica la estructura de la memoria.

En el segundo capítulo, "Estado del arte", se habla de la robótica y los robots imprimibles o PrintBots desarrollados en la Universidad Carlos III de Madrid. También se explica que es la filosofía OpenSource, sus ventajas, y por qué todo este proyecto ha sido realizado basándose en ella. Se abordara el tema de las impresoras 3D usadas en la impresión de los robots. Y se terminara por dar una visión del software OpenRDK usado para el desarrollo de gran parte de la aplicación objetivo de la tesis.

El tercer capítulo, "Hardware" tiene como meta tratar los medios materiales empleados. Así pues comenzara con una descripción general de los robots F-Track y Orugator, para después profundizar en las diferentes partes que lo componen, abordándose por tanto el tema de sus actuadores (servomotores) y del hardware de control (Placa SkyMega). También se presentara el periférico de control, el *Gamepad*, parte fundamental de este trabajo.

En el cuarto capítulo denominado "Diseño software" se describe, mediante el empleo de diagramas, la programación del microcontrolador y de la aplicación basada en OpenRDK que permiten gestionar los movimientos de los robots móviles.

Capitulo cinco, "Descripción de la aplicación". A lo largo de este capítulo se procederá a describir todas las funcionalidades y modos de la aplicación. Además se aportaran todos los conocimientos necesarios para que cualquier usuario de los robots imprimibles pueda manejarlos a través del *Gamepad*, es decir, se explicara la función de cada uno de los botones o *joystick* del periférico de control.

El capitulo seis, "Conclusiones y líneas futuras", servirá para abordar y analizar los resultados obtenidos durante el desarrollo de este proyecto. En función de estos se concretan posibles mejoras y líneas futuras.

Por último, el séptimo capítulo, "Presupuesto", recopilara los costes incurridos a lo largo de esta tesis. Con este fin, este capítulo incluirá un pequeño resumen del proyecto, una división en fases y subfases con el correspondiente diagrama de Gantt y un desglose de costes de personal, costes del material y costes totales.

Capítulo 2

Estado del arte

2.1 La robótica y los robots imprimibles, PrintBots.

La Robótica en los últimos años ha sufrido un gran auge empezando a tomar cada vez más importancia en los planes de estudio de diversas carreras universitarias de carácter técnico. Esto es debido a que la robótica está ganando terreno en la industria y en consecuencia muchas empresas están reclutando candidatos con experiencia en programación de robots. Por esta razón, muchas universidades están enseñando robótica en sus programas de grados y posgrados [15, 13].

Un enfoque común en la enseñanza de programación de robots es el uso de simulaciones, en el que el usuario puede crear diferentes configuraciones de robot con poco esfuerzo. Estos robots pueden ser compartidos con otras personas, multiplicando el número de plataformas [3, 5]. Además, el coste es cero, pudiendo tener tantos robots como quieras y además ellos nunca sufrirán averías. Pero esta solución tiene un inconveniente: los robots simulados no son como los robots reales. Que las cosas funcionen en la simulación no quiere decir que vayan a funcionar de la misma forma en una plataforma real. Además, siempre resulta más emocionante trabajar con robots reales que con simulados.

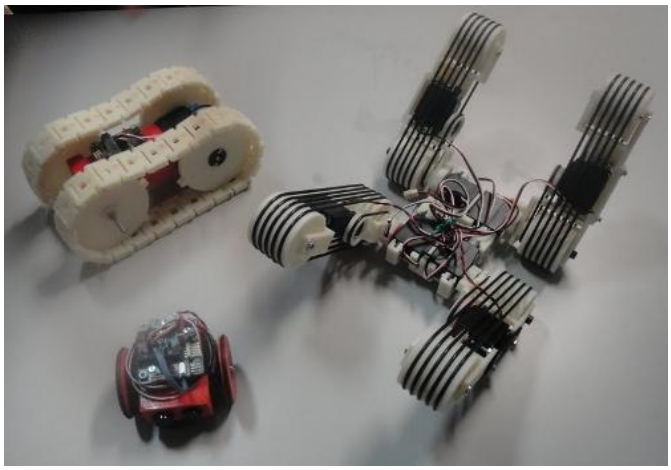


Figura 2. 1: Robots imprimibles o PrintBots desarrollados en la UC3M [24].

¿No sería fantástico, que los robots pudiesen ser compartidos de la misma forma en que se comparte el código de programación (como sucede con las simulaciones)? Si esto fuese posible los investigadores, profesores y estudiantes podrían compartir sus robots de código abierto a través de Internet. Intercambiando ideas con otros grupos de investigación, comparando prototipos, probando sus algoritmos en distintas

configuraciones, desarrollando las propuestas de otros.... esta idea es posible y asequible gracias al código abierto y las impresoras 3D de prototipado rápido y bajo coste como las RepRap [4].

Esto abre una nueva forma de enseñanza de la robótica con las siguientes ventajas:

- Prototipado rápido de plataformas robóticas.
- Bajo coste de impresión de piezas de robot.
- Fácil reconfiguración y adaptación de la plataforma (evolución).
- Fácil intercambio de modelos de robots entre las personas.

Debido a estas grandes ventajas y con el objetivo de adecuarse a este modelo educativo, se han desarrollado en el departamento de Ingeniería de Sistemas y Automática de la Escuela Politécnica Superior de la Universidad Carlos III de Madrid diversos robots imprimibles. Estos robots son totalmente OpenSource (tanto en la mecánica como en la electrónica) y diseñados exclusivamente con herramientas de código abierto (Openscad. FreeCAD e Ivicad). Además sus piezas son impresas en la impresora 3D MakerBot Thing-o-Matic.

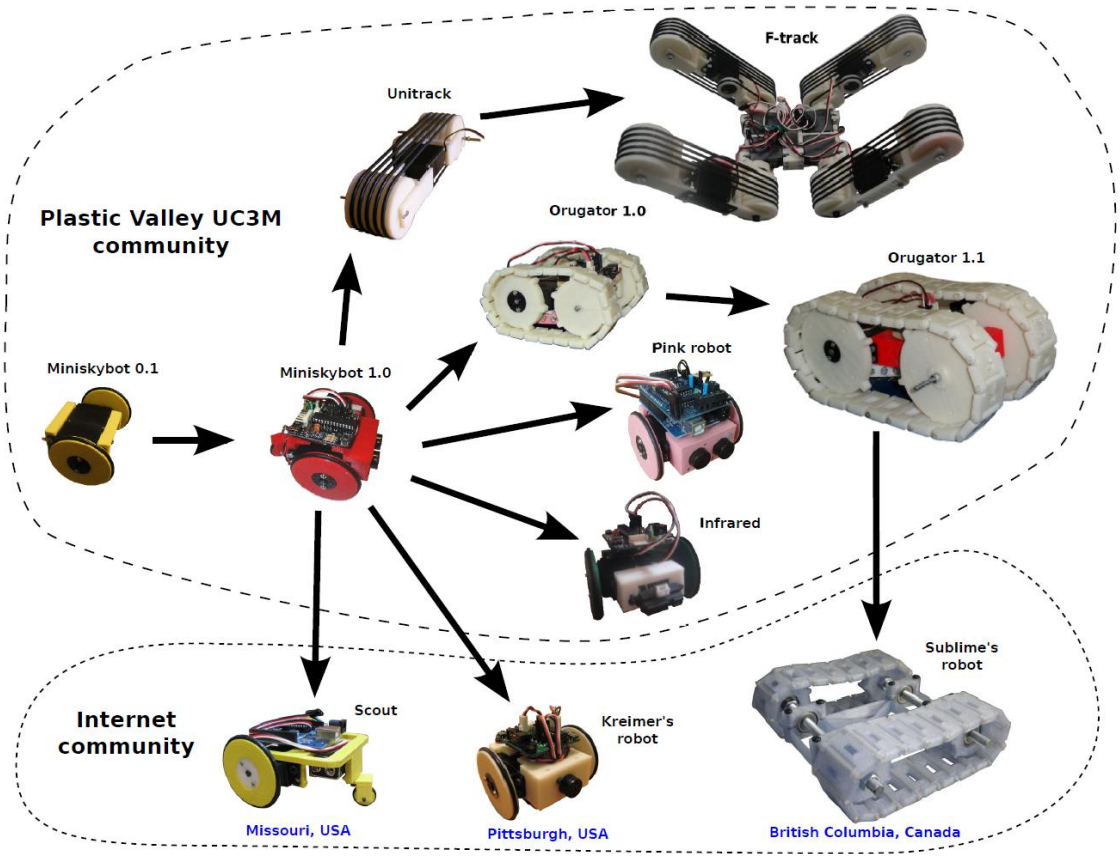


Figura 2. 2: Esquema evolutivo robots imprimibles en la UC3M [17]. A partir de un diseño inicial (Miniskybot 0.1) el diseño de los robots evoluciona impulsado por la creatividad de los estudiantes. Dichos diseños son posteriormente compartidos, lo que provoca que otras personas de la comunidad internacional desarrollen nuevos robots basados en ellos.

2.1.1 Robot móvil Miniskybot

Tal y como se puede observar en la Figura 2.2 el robot Miniskybot es el origen o punto de partida de los posteriores diseños de robots imprimibles. Este robot, Miniskybot, es totalmente open source: todos los diseños mecánicos y electrónicos han sido llevados a cabo bajo licencia copy-left. Además, únicamente han sido empleadas en su diseño herramientas de software de código abierto. Esto es importante porque, al hacerlo, se garantiza que cualquier persona pueda leer, estudiar y modificar los archivos de diseño, sin problemas de licencia y empleando su plataforma informática preferida (Linux, Mac, BSD, Windows...).

Figura 2. 3: Grupo de robots Miniskybot [24]

El Miniskybot es un robot de accionamiento diferencial compuesto de diferentes partes imprimibles y de dos servos modificados. Además ha sido diseñado de modo que se pueda imprimir en una impresora 3D de código abierto como las RepRap.

2.1.2 Diseños derivados del robot Miniskybot

El robot Miniskybot ha sido una referencia y un punto de partida para el diseño de nuevos robots por parte de estudiantes. Así pues a lo largo de 2011 y en tiempo record se desarrollan diferentes robots como el Orugator y el F-Track motivados principalmente por: el pleno acceso al "código fuente" del robot Miniskybot, y el acceso libre a la impresora 3D MakerBot Thing-o-Matic que permite convertir los nuevos diseños en objetos físicos de una forma rápida y sencilla.

En los próximos dos puntos se recogen los robots diseñados por estudiantes de segundo año del grado de ingeniería.

A) Robot imprimible Orugator

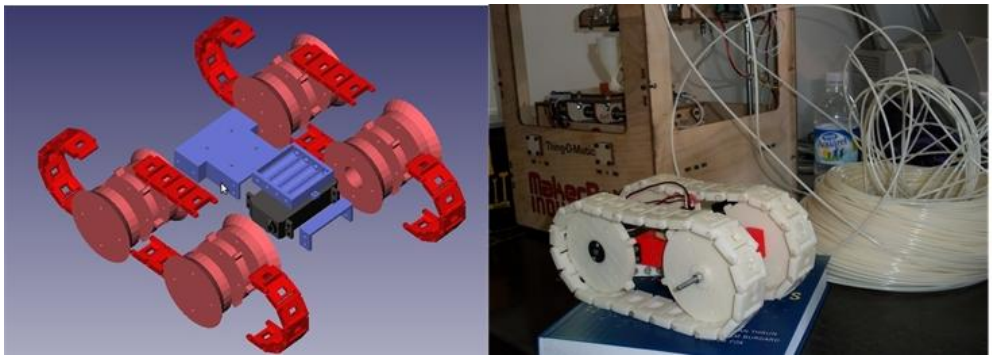


Figura 2. 4: Robot Orugator [25]

El robot Orugator fue diseñado por Olalla Bravo y Daniel Gómez, estudiantes de grado de ingeniería. Este diseño toma como referencia el Miniskybot, pero cambiando sus dos ruedas motrices por dos ruedas de oruga. Convirtiendo al robot Orugator en apto para superar pequeños obstáculos y moverse en superficies irregulares.

B) Robots imprimibles UniTrack y F-Track

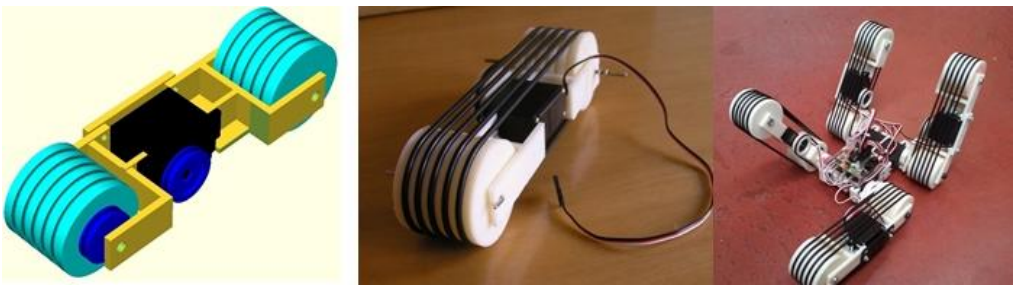


Figura 2. 5: Robot UniTrack y F-Track [19]

El robot UniTrack y F-Track también parte del modelo Miniskybot pero con un enfoque diferente. Dicho robot fue diseñado por Jon Goitia (estudiante de grado de ingeniería) que se centró en el diseño de robots articulados con ruedas de oruga. El primer diseño fue UniTrack, que se muestra en la Figura 2.5 (a la izquierda y en el centro). Se trata de

una rueda de oruga autónoma impulsado por un servo modificado Futaba S3003 (mismo servo que emplea el Miniskybot). Consta de dos ruedas conectadas mediante cinco gomas o juntas tóricas que en conjunto forman la rueda de oruga, mas otra junta tórica que se emplea como sistema de transmisión y une el servo con una de las dos ruedas. Una vez UniTrack fue completamente funcional, se creó el robot F-Track [26], que se muestra en la Figura 2.5 (a la derecha). Este consta de cuatro orugas independientes y articuladas (cuatro UniTracks) unidas a un cuerpo central.

2.2 Filosofía OpenSource

El modelo Open Source presenta como única característica que el código fuente está disponible para todos. La concesión de licencias open source permite a los usuarios la libertad de usar dicho código, estudiarlo, copiarlo, modificarlo y compartirlo. Esto provoca algunas consecuencias importantes. La primera es que potencia la creación de comunidades (desarrolladores, traductores, usuarios...) que evolucionan y mejoran el proyecto de diversas formas.

Una segunda característica es que motiva la creatividad y las nuevas ideas. De acuerdo con [8] "La colaboración e interacción con otros individuos es fundamental para la creatividad. La actividad creativa surge de la relación entre el individuo y su trabajo, y de las interacciones entre el individuo y otros seres humanos". En estas comunidades los miembros interactúan a través de lista de correo (mailing list) y foros de discusión, intercambiando sus opiniones y conocimientos.

La tercera característica es el aprendizaje. Estas es una de las razones más importantes por la que los desarrolladores participan en proyectos de código abierto (Open Source) [10]. Además, en contraste con el modelo clásico de empresa, la motivación no es el dinero, sino el entretenimiento y la pasión que surge del proyecto en el que el desarrollador está participando y se siente involucrado [12].

Las ideas del modelo de código abierto también se han extendido a otros ámbitos, tales como hardware, fotografía, arte, etc. En estas comunidades, la gente está compartiendo los archivos del diseño, en lugar del código fuente. Uno de los mayores éxitos de la comunidad hardware es el proyecto Arduino, donde las placas electrónicas se comparten bajo licencia Open Source Hardware [11]. En esta comunidad la gente comparte esquemas eléctricos y firmware. Los usuarios no sólo poseen permiso para usar, modificar, copiar y estudiar los diseños, sino también para construirlos y venderlos. Es un gran paso adelante, ya que estas comunidades no sólo comparten objetos digitales, sino que han dado el salto a compartir objetos físicos.

Las comunidades que abordan el tema de objetos físicos Open Source son muy jóvenes (alrededor de seis años de edad). Los miembros comparten los archivos de diseño bajo licencias Open Source (por lo general Creative Commons o Open Source Hardware). Sin embargo, podrían aparecer algunas restricciones al compartir los diseños, si el formato no es estándar o Open Source. Por ejemplo, si un diseño mecánico open source ha sido creado usando un software propietario (software no libre o de código cerrado) con su formato de archivo correspondiente, únicamente los usuarios que hayan pagado la licencia de dicho software podrán ejercer la libertad de modificar el diseño. Por lo tanto, dicha comunidad no estaría abierta a todo el mundo, sino únicamente a los usuarios de dicho programa. La solución a esto es crear una comunidad en la que todas las herramientas de software empleadas para el diseño sean OpenSource. Dicha comunidad debe crear diseños de código abierto sólo con herramientas de código abierto, o por lo menos, que permitan guardar el código fuente en un formato de código abierto que pueda ser abierto y modificado por cualquiera sin restricciones [18].

Por tanto podemos decir que el proyecto que en esta memoria se recoge va a ser totalmente Open Source ya que tanto los robots como la electrónica y el software empleado están bajo licencia open source.

2.3 Impresoras 3D de bajo coste.

Bradshaw y otros [2] han realizado recientemente un estudio sobre las impresoras 3D de bajo coste, llevando a cabo un breve repaso de la historia de la impresión 3D desde finales de 1970. En estos más de treinta años, los precios de las impresoras 3D han sufrido un gran descenso, alcanzándose precios asequibles para el público en general y no solamente para grandes empresas[1], permitiendo imprimir piezas complejas de ingeniería de una forma automática gracias al empleo de archivos de diseño que son fácilmente compartidos a través de Internet.

Mientras que el desarrollo de software de código abierto (Open Source) se ha estudiado exhaustivamente, se sabe relativamente poco acerca de la viabilidad del mismo modelo de desarrollo de un diseño de objetos físicos. Las impresoras 3D ofrecen nuevas posibilidades de intercambio de objetos físicos. A medida que se pueden definir objetos mediante código, los investigadores pueden compartir sus propias piezas, modificarlas y construirlas de una forma fácil y sencilla mediante el empleo de impresoras 3D. Esto permite la aparición de comunidades independientes y descentralizadas cuyo fin es producir piezas físicas a partir de diseños digitales y compartir dichos diseños a través de la red. Es decir, al igual que sucede con el software desde hace muchos años,

actualmente existen repositorios en línea de piezas donde la gente puede descargar o subir diseños contribuyendo al desarrollo de esta tecnología.

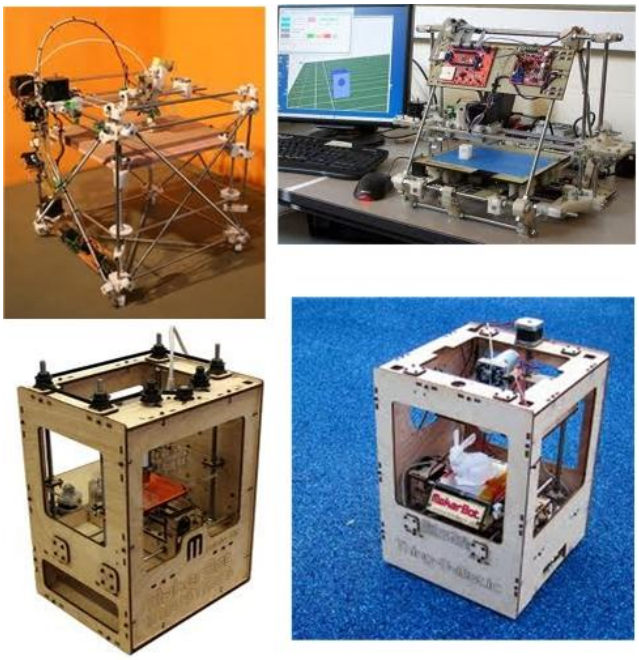


Figura 2. 6: Impresoras 3D Open Source. De izquierda a derecha y de arriba abajo: RepRap Darwin, primera generacion (Mayo, 2007); Reprap Mendel (Sep, 2009) segunda generacion; MakerBot Cupcake (Abril, 2009), primera impresora 3D comercial; Makerbot Thing-o-Matic (Sep, 2010), segunda versión.

En la Figura 2.6 se muestran cuatro de las impresoras 3D Open Source mas importantes. El origen de este tipo de impresoras es el proyecto RepRap [9] iniciado por Adrian Bowyer en el año 2004 con el objetivo de desarrollar una maquina autoreplicante de código abierto. En mayo de 2007 el primer prototipo, llamado Darwin fue terminado y algunos días más tarde, el 29 de mayo se logro la primera replicación. Desde entonces, la comunidad RepRap (máquinas originales reprap y diseños derivados) ha crecido de manera exponencial [4] con una población estimada actual de alrededor de 4.500 máquinas. La segunda generación RepRap, llamada Mendel, fue terminada en septiembre de 2009. Algunas de las principales ventajas de las impresoras de Mendel sobre Darwin son: mayor área de impresión, mayor eficiencia de los ejes, montaje más simple y sencillo, más barata y ligera y mayor portabilidad.

Inicialmente, Darwin y Mendel no fueron diseñados para el público en general, sino para las personas con algunos conocimientos técnicos. A medida que el proyecto RepRap fue adquiriendo carácter Open Source, pequeñas empresas fueron apareciendo para comenzar a comercializar estas impresoras 3D, así como sus diseños derivados. La primera empresa fue MakerBot Industries, fabricando un primer lote de la Cupcake CNC en abril de 2009. A finales de 2009 se habían comercializado cerca de 500 kits

completos. Después de operar durante un año se habían vendido unos 1.000 kits en abril de 2010. Su último diseño es la impresora Thing-o-Matic, anunciada en septiembre de 2010 cuyas principales ventajas son su fácil construcción y empleo, y su bajo coste, alrededor de 950 € [16].

En la actualidad, el Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid dispone de una MakerBot Thing-o-Matic para uso de los estudiantes, esta se muestra en la figura 2.6. Dicha impresora fue totalmente montada por los alumnos y a día de hoy cualquier persona tiene libre acceso a ella. Su principal objetivo se define como el de estimular la imaginación y mejorar la creatividad de sus usuarios [17].

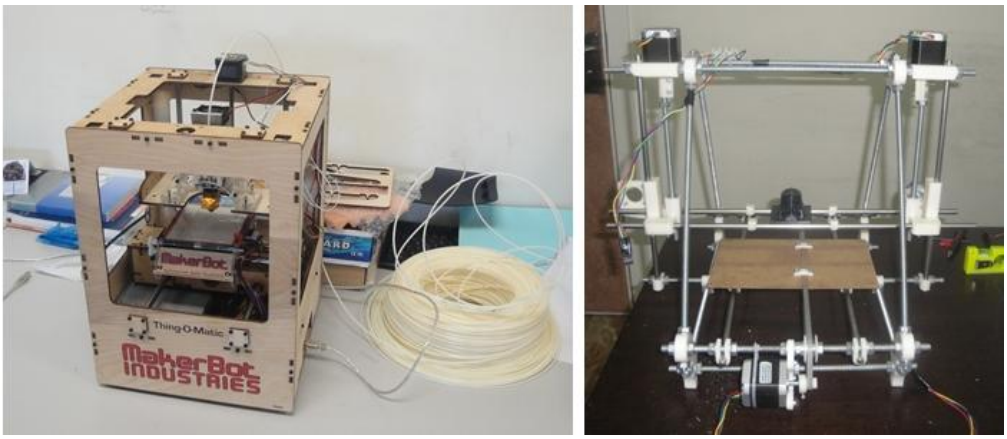


Figura 2. 7: Impresoras 3D Open Source de la UC3M. A la izquierda la MakerBot Thing-o-Matic. A la derecha prototipo de Prusa Mendel en construcción por estudiantes.

Además se ha comenzado un proyecto, llamado "The Clone Wars", en el que un grupo de estudiantes están construyendo sus propias impresoras RepRap desde el principio. Todas las piezas se están imprimiendo en la impresora 3D Thing-o-Matic de la UC3M y se ha elegido como modelo a replicar la Prusa Mendel, debido a su buena documentación y a su fácil montaje. En la figura 2.7 (a la derecha) se muestra el primer prototipo [16].

2.4 OpenRDK

En este apartado vamos a proceder a dar una visión general de OpenRDK (Open Robot Development Kit), el sistema software utilizado por los robots. Este software, es libre y ha sido desarrollado conjuntamente por la universidad de La Sapienza de Roma y por el grupo de control inteligente de la ETSII (Escuela Tecnica Superior de Ingenieros Industriales) de la UPM (Universidad Politécnica de Madrid), ambos grupos de investigación comparten interés por la robótica móvil, aunque sus líneas de investigación son sustancialmente diferentes. Este software lo podemos encontrar en Source Forge [7].

OpenRDK ha sido desarrollado siguiendo el asesoramiento de los usuarios y es un software modular centrado en el desarrollo rápido de sistemas robóticos distribuidos [27]. Este software viene siendo utilizado desde hace varios años y ha sido aplicado con éxito en diversas aplicaciones con robots heterogéneos, tales como:

- Plataformas móviles sobre ruedas: Pioneer 2-A, i-Robot PatrolBot, Pioneer 2-DX, ActivMedia Robotics P2AT y P2DX

Figura 2. 8: Plataforma móvil sobre ruedas P2AT y P2DX

- Vehículos de orugas: Tarantula, Kenaf.

Figura 2. 9: Vehículo de orugas Tarántula y Kenaf

- Vehículos aéreos no tripulados: AscTec Quad-rotor, Parrot AR.Drone.

Figura 2. 10: Vehículo aéreo AR.Drone

- Robots con piernas: Sony QRIO, Sony Aibo, Aldebaran Nao.

Figura 2. 11: Legged robots QRIO, Aibo y Nao

- Simulación de robots en Player/Stage, USARSim, Webbots.

Figura 2. 12: Simulación de robots con Player/Stage y USARSim

Algunas de las características de OpenRDK son:

- El agente, que viene a ser la entidad principal del software OpenRDK y puede definirse como una lista de módulos que son instanciados, junto con los valores de sus parámetros e interconexión. Este agente queda definido en un archivo de configuración.
- Los módulos, que se comunican entre sí mediante un repositorio, (Figura 2.13), en el que almacenan algunas de sus variables internas (parámetros, entradas y salidas) llamadas "propiedades". Un módulo define sus propiedades durante la inicialización. Después podrá acceder a las suyas y a las de otros módulos del mismo o diferentes agentes a través de un esquema de URL global. El acceso a las propiedades de un módulo remoto es transparente, lo que permite reducir el uso de memoria compartida.

Cada módulo normalmente implementa una tarea o comportamiento, como localización, mapeado, planificación de trayectorias y así sucesivamente. Todos estos módulos son implementados en el núcleo de OpenRDK. El conjunto de todos los módulos necesarios constituye el agente [27].

Todos los agentes por lo general tienen:

- Un módulo de comunicación con el robot: simulados o reales.
- Un conjunto de módulos para procesar los datos del sensor.
- Un conjunto de módulos que utiliza los datos procesados al mando del robot.

En la siguiente figura se muestra un ejemplo de un agente que se comunica con un robot. El proceso es el siguiente: Primero el agente recupera los datos del láser lo siguiente que hace el módulo *scan-matcher* es localizar el robot, después el módulo de mapeado construye el mapa, y a continuación el módulo de exploración mueve el robot de acuerdo con el mapa y la posición del robot.

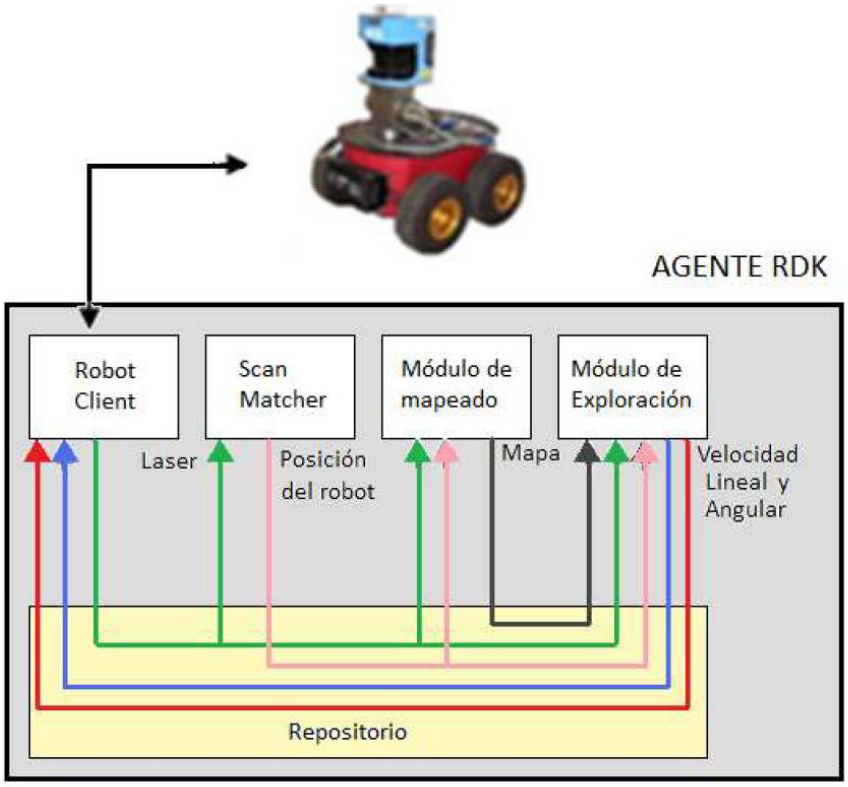


Figura 2. 13: Ejemplo de un agente RDK

Se puede concluir que la elección de OpenRDK para la realización de esta tesis va a ser la más adecuada, ya que además de ser software de código abierto fácilmente descargable a través de su web y por tanto sin requerimientos de comprar caras licencias, va a permitir crear código versátil y reutilizable gracias a la programación de módulos fácilmente intercambiables y validos para futuras aplicaciones.

Capítulo 3

Hardware

Tal y como ya se vio en el *Capítulo 1* en el apartado *1.3 Principales Objetivos*, la misión de esta tesis es lograr que el robot móvil imprimible F-Track sea totalmente controlable por teleoperación mediante el empleo de un *Gamepad*. Además también se definían otros subobjetivos tales como conseguir que la aplicación desarrollada a lo largo de esta memoria fuese válida para controlar otros robots imprimibles ya diseñados y construidos por el departamento de Sistemas y Automática como es el Orugator.

Pues bien, a lo largo de este capítulo se dará una visión de los elementos materiales para lograr tal fin, es decir, se va a explicar el diseño y estructura de los robots móviles imprimibles F-Track y Orugator, los actuadores de los que hace uso, su placa de control y el periférico empleado para su manejo, el *Gamepad*.

3.1 Estructura del robot

A continuación se da una visión del robot F-Track objetivo principal del proyecto, y del robot Orugator .

3.1.1 Estructura robot F-Track

El robot móvil imprimible F-Track cuyo nombre deriva de las cuatro orugas que emplea para poderse desplazar *Four Tracks*, fue diseñado y construido a lo largo de 2011 por Jon Goitia alumno de grado de Ingeniería Industrial. Dicho diseño fue realizado usando el software OpenSCAD para después ser impreso por la impresora 3D MakerBot Thing-o-Matic [14] del departamento de Sistemas y Automática de la Universidad Carlos III de Madrid.

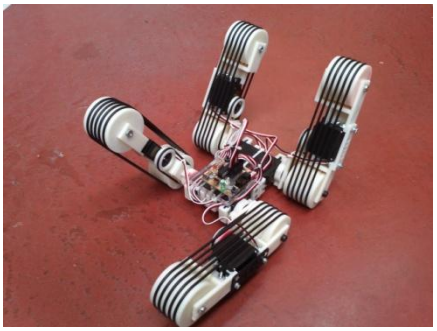


Figura 3. 1: Robot F-Track

El robot consta de cuatro ruedas de oruga [19], cada una con un servomotor Futaba S3003 modificado que permite su desplazamiento hacia delante y atrás. Dichas orugas están unidas a un cuerpo central a través de cuatro servos Futaba S3003 que permiten el

levantamiento y posicionamiento de las orugas en un rango de 90º a -90º independientemente de la posición de las demás.

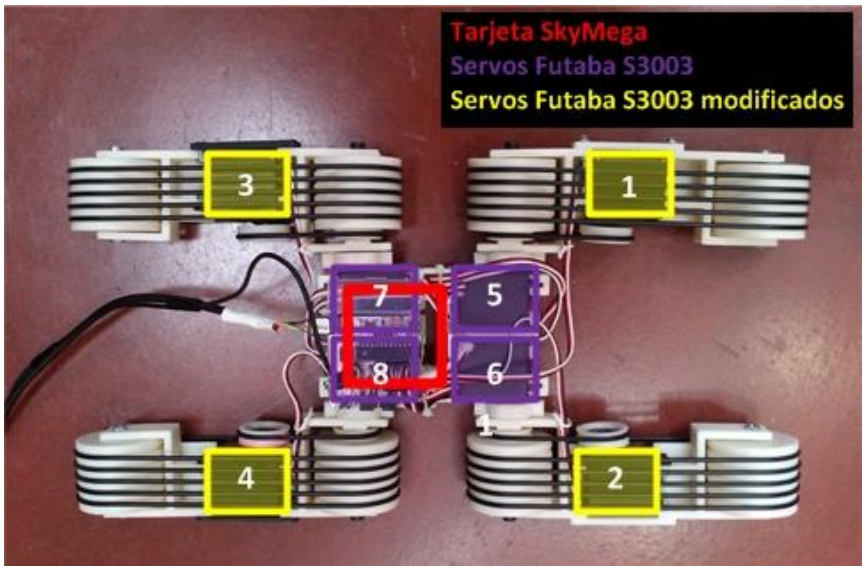


Figura 3. 2: Descripción robot F-Track

Además el F-Track contara con una tarjeta microcontroladora, dicha tarjeta será la SkyMega de la cual se hablara en este capítulo más adelante, con el objetivo de poder controlar el movimiento de los ochos servos y permitir que el robot se pueda comunicar con un PC.

Dicho robot gracias a su diseño y al uso de ocho servomotores independientes permiten un amplio abanico de movimientos, convirtiéndole en apto para superar toda clase de pequeños obstáculos incluido subir y bajar escaleras.

3.1.2 Estructura robot Orugator

Viene a ser el robot anterior en el tiempo al F-Track y al igual que este ha sido diseñado usando OpenSCAD e impreso en una impresora 3D MakerBot Thing-o-Matic.

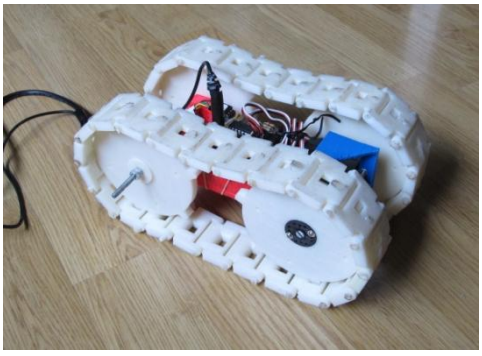


Figura 3. 3: Robot Orugator

Consta de dos orugas accionadas cada una por un servomotor Futaba S3003 (modificado para rotación continua) y gobernadas por la tarjeta microcontroladora SkyMega [25].

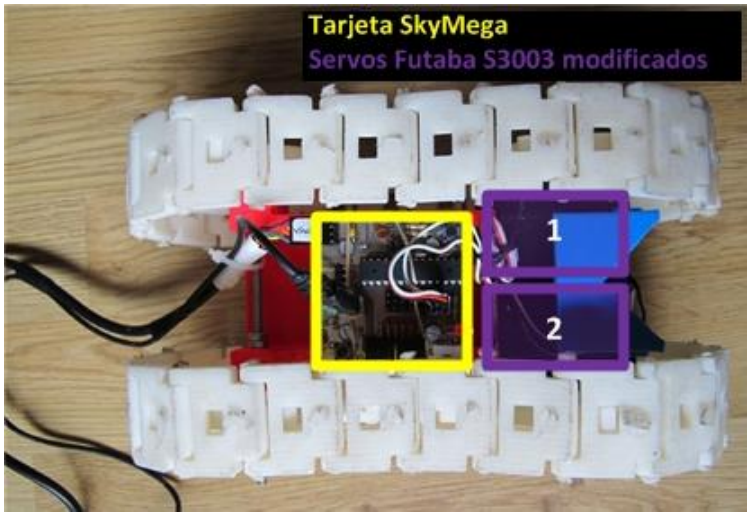


Figura 3. 4: Descripción robot Orugator

Este robot presenta muy pocas posibilidades de movimiento, ya que estos se reducen únicamente a girar oruga en una dirección y su contraria. Por los que los movimientos que se podrán realizar son:

- Si las dos orugas se mueven conjuntamente en la misma dirección el robot se desplazara hacia delante o atrás dependiendo del sentido de giro de estas.
- Si las orugas se mueven en sentido contrario la una respecto a la otra el robot girara a izquierdas o derechas.

Pese a la gran sencillez de este robot y su gran limitación de movimientos, se aprecia que su diseño de las ruedas como orugas lo hace apto para sortear pequeños obstáculos y desplazarse en terrenos rocosos e inclinados.

3.2 Actuadores

Los actuadores tienen como misión generar el movimiento de los elementos del robot según las órdenes dadas por la unidad de control. Los actuadores utilizados en robótica pueden emplear energía neumática, hidráulica o eléctrica. Cada uno de estos sistemas presenta características diferentes, siendo preciso evaluarlas a la hora de seleccionar el tipo de actuador más conveniente.

Se clasifican en tres grandes grupos, según la energía que utilizan:

- **Neumáticos:** Estos usan el aire comprimido como fuente de energía, siendo muy indicados para movimientos rápidos, pero su precisión es limitada.
- **Hidráulicos:** Son recomendables en los manipuladores que tienen gran capacidad de carga, junto a una precisa regulación de la velocidad.
- **Eléctricos:** Son los más usados, por su fácil y preciso control, así como por otras propiedades ventajosas que establece su funcionamiento como consecuencia del empleo de la energía eléctrica.

En nuestro caso se opto por actuadores de tipo eléctrico, ya que presentan varias características que nos convienen, tales como: precisión, facilidad de control y sencillez de instalación. La desventaja que presenta este tipo de actuadores es su potencia limitada, aunque en este caso no nos afecta ya que no se necesita que desplacen grandes cargas [20].

El sistema de accionamiento de F-Track está compuesto por ocho servomotores. Cuatro de ellos son empleados para el levantamiento y posicionamiento de las orugas respecto al cuerpo central y los otros cuatro restantes se encargan del movimiento de desplazamiento de las orugas. Para tal objetivo, los últimos cuatro servos han sido modificados para giro continuo, es decir, para poder ser controlados como motores de corriente continua y por tanto sin limitación de ángulo de giro.

Mientras tanto el Orugator solamente hace uso de dos servomotores modificados cuyo objetivo es el de movimiento de avance de cada una de las orugas.

3.2.1 Servomotores.

Un servomotor (también llamado servo) es un dispositivo similar a un motor de corriente continua, que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación y mantenerse estable en dicha posición. Está conformado por un motor, una caja reductora y un circuito de control. Los servos se utilizan frecuentemente

en sistemas de radiocontrol y en robótica, pero su uso no está limitado a estos. Es posible modificar un servomotor para obtener un motor de corriente continua que, si bien ya no tiene la capacidad de control del servo, conserva la fuerza, velocidad y baja inercia que caracteriza a estos dispositivos [21].

Figura 3. 5: Circuito interno de un servomotor

Los servomotores poseen tres terminales de entrada:

- Terminal positivo: Recibe la alimentación del motor (4 a 8 voltios).
- Terminal negativo: Referencia tierra del motor (0 voltios).
- Entrada de señal: Recibe la señal de control del motor.

Los colores del cable de cada terminal varían con cada fabricante: el cable del terminal positivo siempre es rojo; el del terminal negativo puede ser marrón o negro; y el del terminal de entrada de señal suele ser de color blanco, naranja o amarillo.

El principio de funcionamiento es el mismo, independientemente del modelo, e incluso del fabricante. Para posicionar el servomotor con un ángulo determinado, se enviara una señal encriptada que recibirá su circuito de control. Este la procesara y si la posición a la que equivale esta señal, coincide con la que tiene el servo, mantendrá la posición. De lo contrario girara hasta encontrar la posición solicitada.

La señal encriptada es una señal cuadrada de voltaje. El ángulo de ubicación del motor depende de la duración del nivel alto de la señal (Ton). A este tipo de control se le denomina PWM (Pulse Width Modulation, modulación por ancho de pulso). Hay que tener en cuenta que para bloquear el servomotor en una posición, es necesario enviarle continuamente una señal con la posición deseada. De esta forma el servo conservara su posición y se resistirá a fuerzas externas que intenten cambiarlo de posición. Si los pulsos no se envían, el servomotor queda liberado, y cualquier fuerza externa puede cambiarlo de posición fácilmente [21].

Cada servo motor dependiendo de la marca y modelo utilizado, tiene sus propios márgenes de operación. Los servomotores elegidos, tanto para el robot F-Track como para el Orugator, son de la marca FUTABA y el modelo es el S3003 Standard. Estos se

ajustan perfectamente al peso, tamaño y voltaje de alimentación requeridos para poder ser controlados por la placa SkyMega.

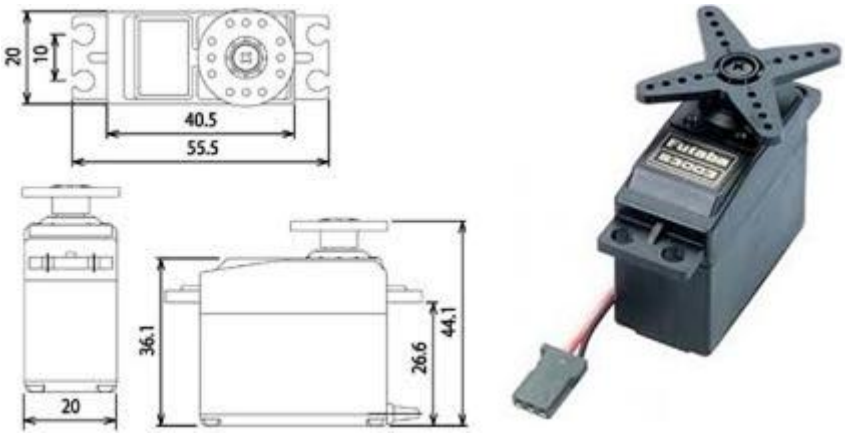


Figura 3. 6: Dimensiones servo Futaba S3003

Estos servos se controlan aplicando una señal PWM por su cable de control. Las señales PWM (Pulse Width Modulation, Modulación por anchura de pulso) son digitales (pueden valer 0 ó 1) y permiten que usando un único pin de un microcontrolador podamos posicionar el servo. Por lo que haciendo uso de la placa SkyMega podremos llegar a controlar 8 servos a la vez, que es el número requerido por el robot F-Track.

Los Futaba S3003 se alimentan a 5V, y para posicionar el servo hay que aplicar una señal periódica, de 50Hz (20ms de periodo). La anchura del pulso determina la posición del servo. Si la anchura es de 2.3ms, el servo se sitúa en un extremo y si la anchura es de 0.3ms se sitúa en el opuesto (90º y -90º respectivamente). Cualquier otra anchura entre 0.3 y 2.3 sitúa el servo en una posición comprendida entre un extremo y otro. Por ejemplo, si queremos que se sitúe exactamente en el centro (0º), aplicamos una anchura de 1.3ms.

La relación entre posición en grados (Θ) del eje del servomotor y el ancho de pulso de la señal PWM en milisegundos (t) viene dada por la siguiente expresión:

$$\theta = 90 \cdot (t - 1.3)$$

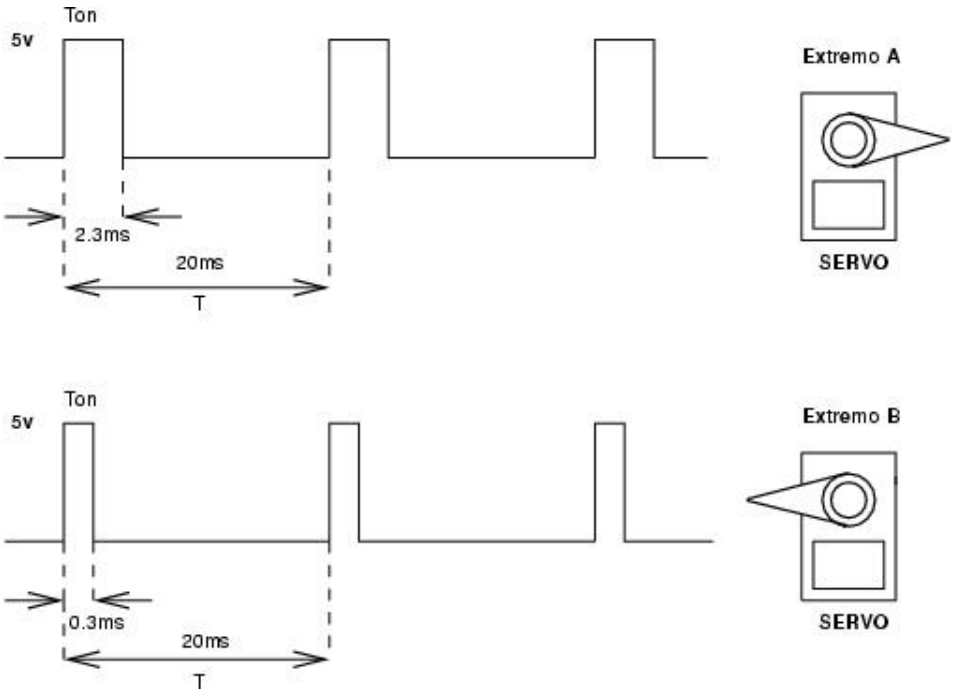


Figura 3. 7: Pulsos de control Futaba S3003 [22]

Para este proyecto se han usado cuatro servomotores S3003 sin modificar y controlados tal y como se ha descrito en este apartado. Pero también se han requerido de otros seis (4 para el F-Track y 2 para el Orugator) servomotores S3003 modificados para giro continuo, es decir para que su comportamiento sea como el de un motor de corriente continúa.

3.2.2 Servomotores modificados para giro continuo.

Los servos Futaba S3003 o compatibles son un tipo especial de motores que se usan para construir articulaciones y como tal, poseen un rango de giro limitado a 180 grados. Esta característica va a ser idónea para el levantamiento y posicionamiento de los brazos (orugas) del F-Track, pero para mover las ruedas de la oruga que permiten el desplazamiento del robot no van a ser validos. Ya que la limitación de giro de los servos impide que estos giren continuamente en una misma dirección. Para solucionar este problema se procederá a la modificación de los servomotores tal y como se explica a lo largo de este apartado.

Se procederá a desmontar el servo, eliminar el tope físico que presentan sus engranajes para limitar el giro, y manipular su electrónica para que al recibir una señal periódica de 50Hz PWM su comportamiento sea el siguiente en función del ancho de pulso:

- Ancho de pulso igual 1.3ms: el servo permanece inmóvil.

- Ancho de pulso comprendido en el rango (1.3ms , 2.3ms]: El servo gira en dirección de las agujas del reloj (sentido positivo) con una velocidad constante mayor, cuanto mayor es el ancho de pulso.
- Ancho de pulso comprendido en el rango [0.3ms , 1.3ms): El servo gira en dirección contraria a las agujas del reloj con una velocidad constante mayor, cuanto menor es el ancho de pulso.

Haciendo uso de la expresión que relaciona la posición en grados del eje del servo (Θ) con el ancho de pulso en milisegundos (t):

$$\theta = 90 \cdot (t - 1.3)$$

Podríamos resumir lo anterior, diciendo que si al servo modificado se le manda la orden de situarse en la posición 0º el servo no se mueve o se para en caso de que estuviese girando. Mientras que si se le da la orden de ir a uno mayor que 0º girara en sentido positivo y si se le ordena un valor negativo girara en sentido contrario.

De forma experimental se ha obtenido la velocidad en r.p.m. (revoluciones por minuto) en función del ángulo de posicionamiento en grados para el servo Futaba S3003 modificado.

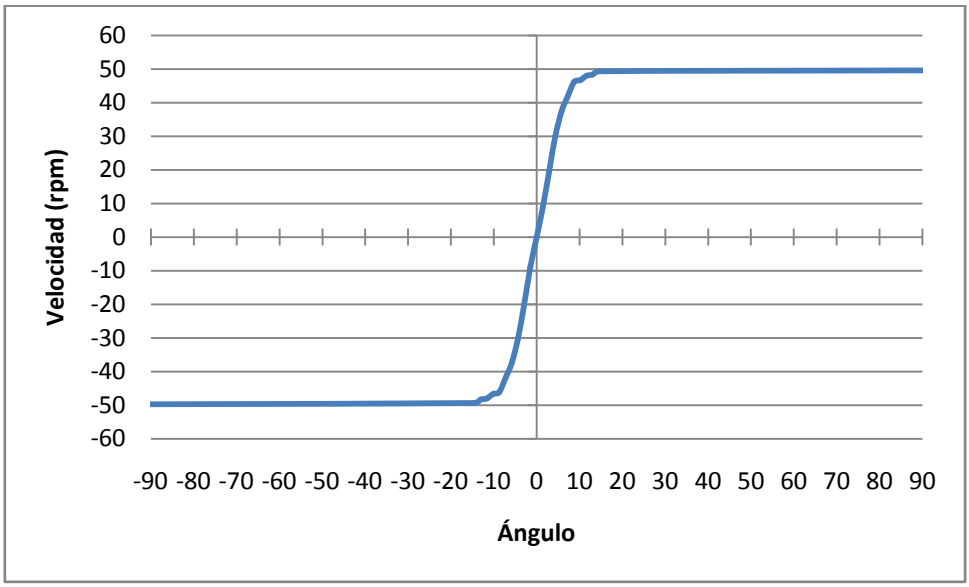


Figura 3. 8: Velocidad servo "Futaba S3003 modificado" en función del ángulo

Tal y como se puede observar en la gráfica el servomotor modificado va a permitir controlar el sentido de giro de su eje y la velocidad de este en un rango comprendido entre 0 rpm y 50 rpm. Para llevar a cabo tal control de la velocidad se deberá ordenar al servo motor valores de posición comprendidos entre los -12º y los +12º.

Esta característica de los servos modificados va a permitir que podamos controlar la velocidad de desplazamiento de los robots F-Track y Orugator.

3.3 Placa de control. Placa SkyMega

El control va a ser una de las partes fundamentales del proyecto, y para tal objetivo se va a emplear la tarjeta microcontroladora SkyMega cuyas características principales se enumeran a continuación [23]:

- Hardware libre.
- Compatible con Arduino.
- Microprocesador: ATMEGA a 16Mhz. Modelos: 88/168/328.
- Conexión de hasta 8 servos.
- Los conectores de los servos se pueden poner por ambas caras de la placa.
- Comunicación por bus I2C entre tarjetas skymega .
- Hasta 2 conectores de I2C, que se pueden soldar por ambas caras.
- Conector de alimentación doble, tipo molex, uno por cada cara.
- Conector de grabación ICSP.
- Led de pruebas.
- Pulsador de pruebas.
- Micro-interruptor de on/off.
- Led de power-on.
- Slot de expansión para conectar sensores.

Cabe destacar de dichas características que sea Hardware libre y por tanto su diseño esté disponible públicamente y que permita la conexión de hasta 8 servos, que son exactamente los requeridos por el robot F-Track.

En cuanto al microcontrolador que va a llevar equipado para la realización de esta tesis será de todos los compatibles, el ATMEGA 168/V de ATMEL, que será programado en lenguaje C haciendo uso del software AVR Libc.

En la siguiente imagen se puede apreciar la tarjeta SkyMega con sus diferentes partes señaladas:

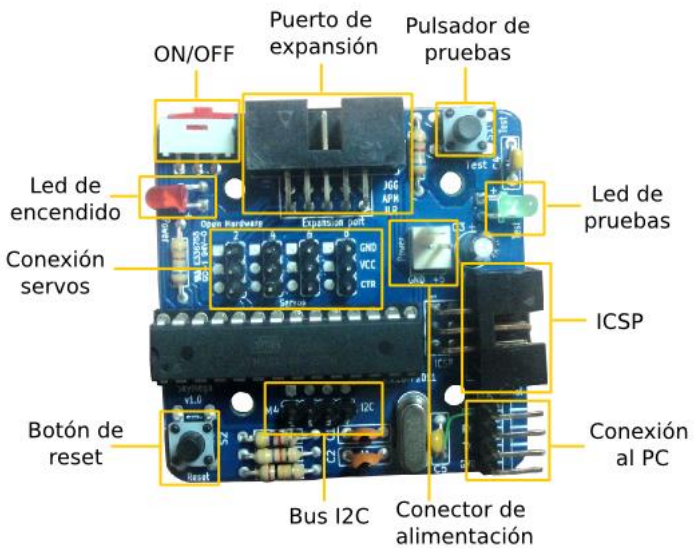


Figura 3. 9: Descripción de la SkyMega 1.0 [23]

3.3.1 Conexión servos

Los servos son conectados a la placa SkyMega a través de los terminales reservado para tal fin, tal y como se observa en la Figura 3.9. Dichos terminales están numerados del 1 al 8, siendo el 1 el que está más cerca del borde de la placa.

En cuanto al orden de conexión de los diferentes servos a la placa se puede apreciar en la Figura 3.2 y en la Figura 3.4 que los servos han sido numerados. Dicha numeración nos indica que el servo numero 1 será conectado al terminal de servo 1, que el servo numero 2 al terminal 2 y así sucesivamente hasta el servo numero 8 y el terminal 8.

3.3.2 Conexión al PC

La skymega se conecta al PC a través de un cable USB-serie de FTDI (modelo TTL-232R-5V). Este cable tiene un conector de 6 pines. Para usarlo con la SkyMega es necesario modifica este conector y sustituirlo por uno de 4 pines como se muestra en la siguiente imagen [23].

Figura 3. 10: Conector cable USB-serie de FTDI (modelo TTL-232R-5V) modificado para uso con SkyMega [23]

Este cable sirve para descargar firmware en la SkyMega, así como comunicarse vía puerto serie con ella.

3.3.3 Alimentación

La SkyMega se alimenta entre 4.5 y 6 voltios a través de un conector molex de 2 vías. Dicha alimentación se puede realizar de diferentes maneras tales como mediante el empleo de baterías, de un portapilas de 4 pilas AAA o de una fuente de alimentación.

De todas estas opciones se opto por la de ser alimentada a través de un adaptador AC/DC conectado a la red eléctrica, cuyas características son las siguientes:

INPUT: 100-240V AC 0.4A 50-60Hz
OUTPUT: 5V DC 1500M

Los motivos de dicha elección son que durante la fase de pruebas del robot F-Track se detecto un alto consumo energético debido al empleo de ocho servomotores. Lo que se traducía en una corta duración de las baterías o pilas. Además como el robot era teleoperado a través del cable USB-serie de FTDI, es decir, en todo momento existe un cable que conecta el PC con el vehículo no suponía un problema que existiese otro cable que lo conectase con una fuente de alimentación.

3.4 Periférico para control del robot. El Gamepad

Uno de los objetivos principales de este proyecto era lograr un completo, eficiente y fácil control del robot móvil imprimible F-Track por cualquier persona sin conocimientos previos en la materia.

Para alcanzar tal meta el periférico de control debía ser lo más intuitivo posible además de resultarle familiar a cualquier tipo de usuario. Por tanto, para dar solución a dicho problema se decidió emplear el periférico usado por excelencia en los videojuegos, el *Gamepad*.

Figura 3. 11: Gamepad "Logitech Dual Action"

El *Gamepad* elegido ha sido el "Logitech Dual Action", y el motivo de su elección es el gran parecido con los gamepad que emplean videoconsolas de gran éxito mundial tales como PlayStation y Xbox.

Dicho dispositivo se conecta al PC a través de cable USB y está compuesto por 12 botones digitales, dos *joysticks* analógicos y un *pad* octodireccional.

Figura 3. 12: Descripción del Gamepad

Esta gran cantidad de botones combinados con movimientos de los joysticks permite infinidad de posibles modos de control para cualquier tipo de robot.

Los controles y modos de uso asociados al gamepad tanto del robot F-Track como del Orugator son explicados detenidamente más adelante en el *Capítulo 5: Descripción de la aplicación*.

Capítulo 4

Diseño Software

En este cuarto capítulo denominado “Diseño software” se describe, mediante el empleo de diagramas, la programación del microcontrolador y de la aplicación basada en OpenRDK que permiten gestionar los movimientos de los robots móviles.

4.1 Programación del microcontrolador

La programación del microcontrolador embebido en la placa SkyMega [23] es una de las partes fundamentales de este proyecto. Ya que va a ser este el que se encargue del control de velocidad y posición de los servomotores que gobiernan los movimientos de los robots móviles imprimibles, así como de gestionar las ordenes enviadas por la aplicación que se ejecuta en el PC.

El microcontrolador a programar es un ATMEL Atmega 168/V y entre la posibilidad de ser programado en lenguaje ensamblador o lenguaje de programación C se ha optado por la segunda. Dicha elección responde a que resulta más rápido y cómodo crear código en un lenguaje de programación de alto nivel como es C, que en un leguaje de bajo nivel como es ensamblador.

A continuación se describe la estructura y funcionamiento del programa que ejecuta el microcontrolador. Dicho programa se divide en el programa principal cuya ejecución es continua y que produce llamadas a una interrupción, y en la rutina de atención a dicha interrupción (ISR, Interrupt Service Routine).

4.1.1 Programa principal

Los objetivos del programa principal que ejecuta el microcontrolador son los siguientes:

- Configuración de los puertos E/S y de los diferentes registros que gobiernan el comportamiento del microcontrolador Atmega 168. Es decir, configuración de interrupciones, de la velocidad de transmisión del puerto serie, etc.
- Recepción de datos a través del puerto serie. Lo que permite que se establezca comunicación PC – robot.
- Asignación de los valores de velocidad y posición que deberán tomar los actuadores (servomotores) del robot.

Todo esto se puede apreciar en el siguiente diagrama de flujo.

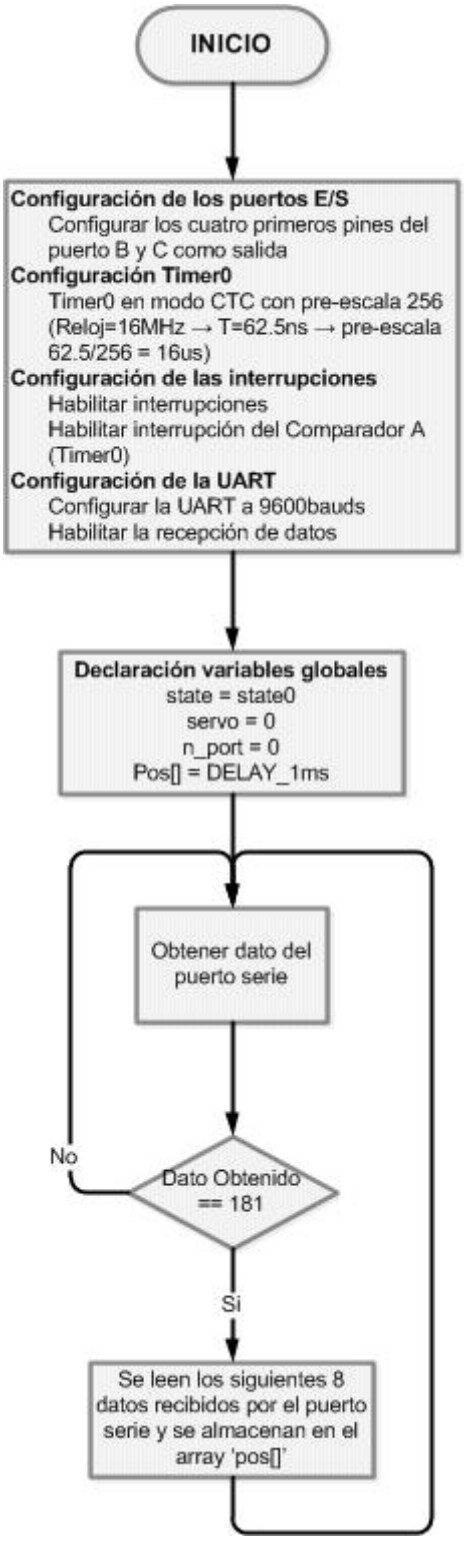


Figura 4. 1: Diagrama de flujo programación del microcontrolador. Programa principal.

Tal y como se puede observar, tras configurar los diferentes puertos y registros y declarar las variables globales de las que hará uso la ISR, el programa se introduce en

un bucle infinito en el que constantemente se están leyendo datos del puerto serie (caracteres), es decir, datos enviados por el PC al robot. Y en función de dichos valores se realizara lo siguiente: si este es igual a un valor de sincronización (181) quiere decir que los siguientes 8 datos (caracteres) van a ser los valores de posición de los 8 servos, realizándose las consecuentes operaciones de asignación. En caso contrario, se siguen leyendo los valores del puerto serie hasta que se encuentre el valor de sincronización.

Cabe destacar que esta rutina principal asigna lo valores de posición o velocidad que han de adquirir los actuadores del robot móvil, pero no crea la señal PWM encargada de tal fin. De esto se encarga la rutina de atención a la interrupción (ISR).

4.1.2 Rutina de atención a la interrupción

La rutina de atención a la interrupción o ISR (Interrupt Service Routine) es la encargada de crear una señal de control para cada uno de los ocho servos.

Esta señal de control será una señal PWM (Pulse Width Modulation, modulación por ancho de pulso), es decir, una señal periódica de 50Hz (20ms de periodo) cuadrada, cuya anchura de pulso determina la posición del servo tal y como se explico en el capítulo 3, *Hardware*.

La ISR se ejecuta cada vez que se dispara el comparador A, y dicho comparador se dispara cada vez que el Timer0 alcanza el mismo valor que posee el registro del comparador A (registro OCR0A). El Timer0 se incrementa una unidad cada 16µs, por lo que se producirá una llamada a interrupción cada vez que transcurra un lapso de tiempo que viene dado en función del valor del registro OCR0A. A continuación se pueden observar a modo de ejemplo los tiempos que tardara en lanzarse una llamada de interrupción en función del valor con que se cargue el registro del comparador A (OCR0A).

- Si OCR0A = 19 $\Rightarrow$ $19 \cdot 16\mu s = 304\mu s \Rightarrow$ ISR se ejecuta cada 0.3 ms.
- Si OCR0A = 12 $\Rightarrow$ $12 \cdot 16\mu s = 192\mu s \Rightarrow$ ISR se ejecuta cada 0.2 ms.
- Si OCR0A = 62 $\Rightarrow$ $62 \cdot 16\mu s = 992\mu s \Rightarrow$ ISR se ejecuta cada 1 ms.
- Si OCR0A = 125 $\Rightarrow$ $125 \cdot 16\mu s = 2000\mu s \Rightarrow$ ISR se ejecuta cada 2 ms.

Haciendo uso de esta propiedad se programa la ISR como una máquina de estados finitos cuya transición de un estado a otro viene marcado por el transcurso de un cierto intervalo de tiempo. Se definen cuatro estados:

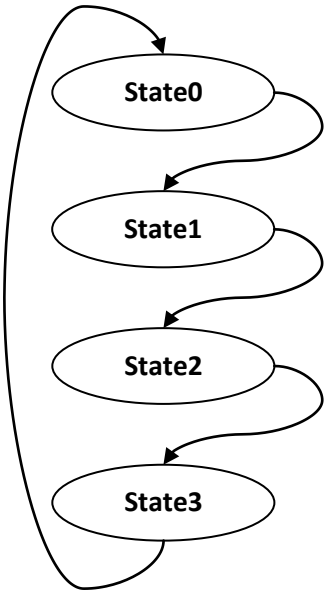


Figura 4. 2: Maquina de estados finitos.

- **State0:** Señal PWM a '1' durante un intervalo de tiempo de 0.3 ms.
- **State1:** Señal PWM a '1' durante un tiempo que viene dado por la variable 'pos[servo]'. Es decir, en el estado 1 se mantiene la señal PWM en alto (a 5V) durante un intervalo de tiempo que depende del valor de posición que queremos que adopte el servo. Si dicho intervalo de tiempo es de 2ms el servo se sitúa en un extremo, mientras que si es 0ms se posiciona en el opuesto (90º y -90º respectivamente). Cualquier otra anchura entre 0.3 y 2.3 sitúa el servo en una posición comprendida entre un extremo y otro. Por ejemplo, si queremos que se sitúe exactamente en el centro (0º), el estado 1 debe mantener la señal PWM a '1' durante 1ms.
- **State2:** Señal PWM a '0' durante un tiempo igual a 2ms – 'pos[servo]'. Este estado es el encargado de crear la parte complementaria de la señal PWM de 50Hz
- **State3:** Señal PWM a '0' durante 0.2 ms

Tal y como se observa cada uno de los estados crea una parte de la señal PWM, siendo los dos primeros los encargados de la parte en que la señal esta en alto o a 5V y los dos segundos los de la señal en bajo o 0V.

El diagrama de flujo de la ISR que implementa esta máquina de estados se presenta a continuación:

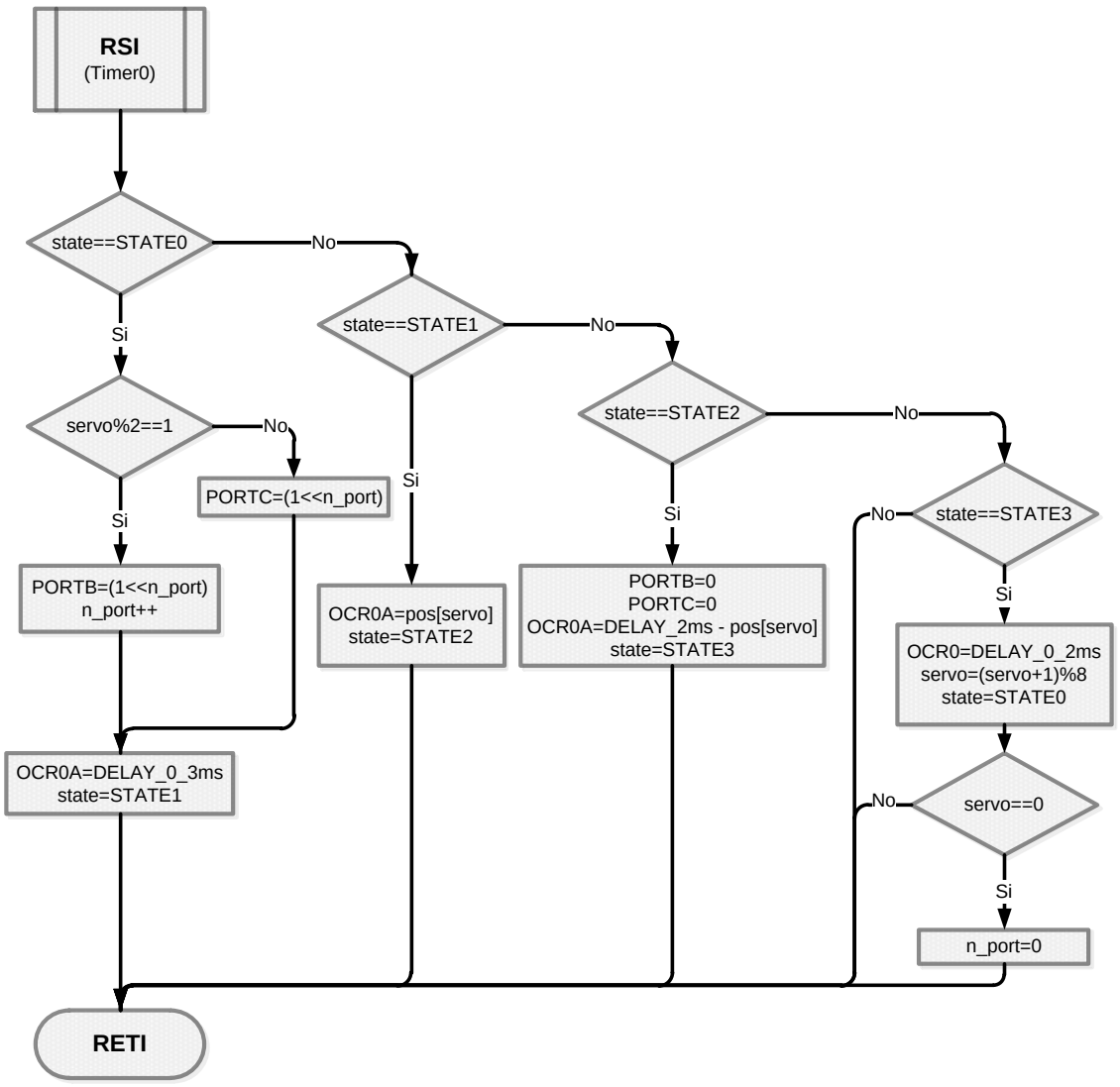


Figura 4. 3: Diagrama de flujo programación del microcontrolador. Subrutina de atención a la interrupción.

Como complemento a este apartado se añade al final del presente documento a nota de *ANEXO I* el código de programación del microcontrolador.

4.2 Diseño de la aplicación. Robot F-Track.

Ahora que el microcontrolador ha sido correctamente programado para recibir órdenes del PC y en función de ellas gestionar el movimiento de las diferentes partes móviles del robot, se va a proceder a explicar el diseño y la estructura de la aplicación que se ejecuta en el PC.

Esta aplicación tiene como misión leer y procesar los valores del gamepad y en función de estos mandar una serie de órdenes al robot para que realice ciertos movimientos. Es decir, la estructura de la aplicación se podría resumir como: leer valores del gamepad, procesar dichos datos y enviar órdenes al robot.

Figura 4. 4: Esquema sistema completo

El diseño y desarrollo de esta aplicación se va a realizar empleando el software de programación modular OpenRDK del cual ya se hablo en el Capitulo 2.

Mediante el empleo de OpenRDK se van a crear varios módulos que se comunican entre sí mediante un repositorio, cada uno de los cuales implementara una tarea o comportamiento. Para nuestro caso de estudio se crearan tres módulos cuyas tareas son:

- **Leer valores del gamepad o periférico de control.** Este modulo al que nos referiremos de aquí en adelante como *GamepadReaderModule* va a ser el encargado de comunicarse con el gamepad, leer sus valores y guardarlos en el repositorio para que puedan ser accedidos por otros módulos.
- **Determinar la velocidad o posición de los servos en función de los valores del gamepad.** Responsable de tal objetivo será el modulo llamado *Gamepad2FTrackModule*. Dicho modulo leerá del repositorio los valores de los diferentes botones y ejes del periférico de control, y en función de estos llevara a cabo una serie de operaciones que darán como resultado los valores de posición

o velocidad que deben adquirir los servos. Dichos valores serán escritos en el repositorio.

- **Comunicarse con el robot, es decir, comunicarse con la placa de control SkyMega con la que este va equipado.** Este modulo recibirá el nombre de *FTrackClientModule* y será el encargado de tomar del repositorio los valores de posición y velocidad de cada uno de los ochos servos, y de enviárselos al robot a través del puerto serie.

En la siguiente figura (Figura 4.5) se muestra la estructura del agente RDK que emplea los módulos descritos. El proceso es el siguiente: Primero el agente recupera los datos del gamepad, lo siguiente que hace el módulo *Gamepad2FTrackModule* es determinar los valores de control de los servos y por último se envían las órdenes de movimiento al robot.

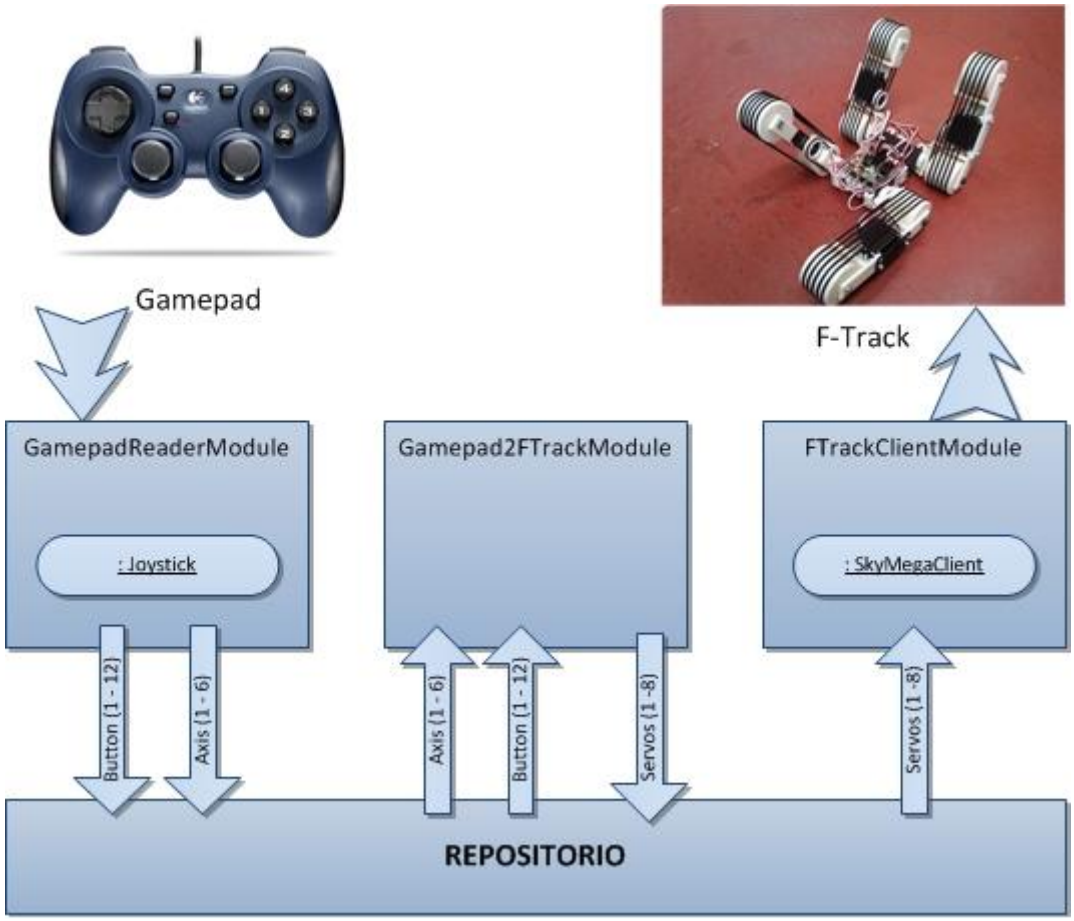


Figura 4. 5: Estructura del agente RDK para el robot F-Track

Cabe destacar de la Figura 4.5 el empleo por parte del modulo *GamepadReaderModule* de la clase *Joystick* para facilitar las operaciones con el gamepad, así como el empleo de la clase *SkyMegaClient* por parte del modulo *FTrackClientModule*.

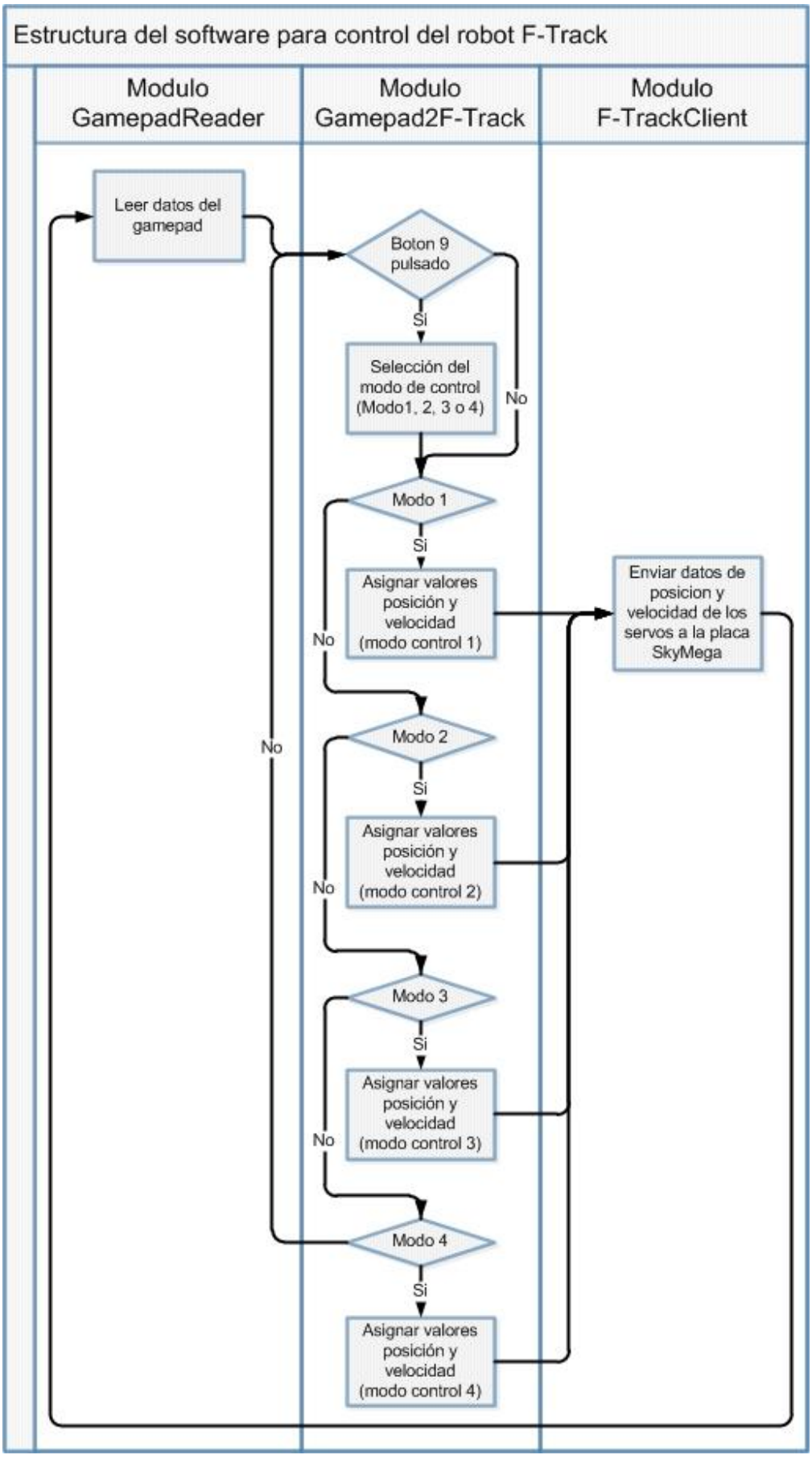


Figura 4. 6: Estructura del software para control del robot F-Track

La Figura 4.6 da una visión más precisa de las operaciones llevadas a cabo por cada modulo y la comunicación e interacción entre estos. Mención especial requiere el modulo *Gamepad2FTrack* en el cual se observa que dependiendo del modo de control seleccionado (modo 1, 2, 3 o 4) la asignación de valores de los servos se realizara de distinta forma. Todo lo relacionado a los modos de control se estudiara más profundamente en el capítulo 5 *Descripción de la aplicación*.

4.2.1 Diagramas de flujo

Ahora en este apartado se van a presentar los diagramas de flujo de cada uno de los 3 módulos.

- ***GamepadReaderModule***

Figura 4. 7: Diagrama de flujo modulo *GamepadReader*.

- ***Gamepad2FTrackModule***

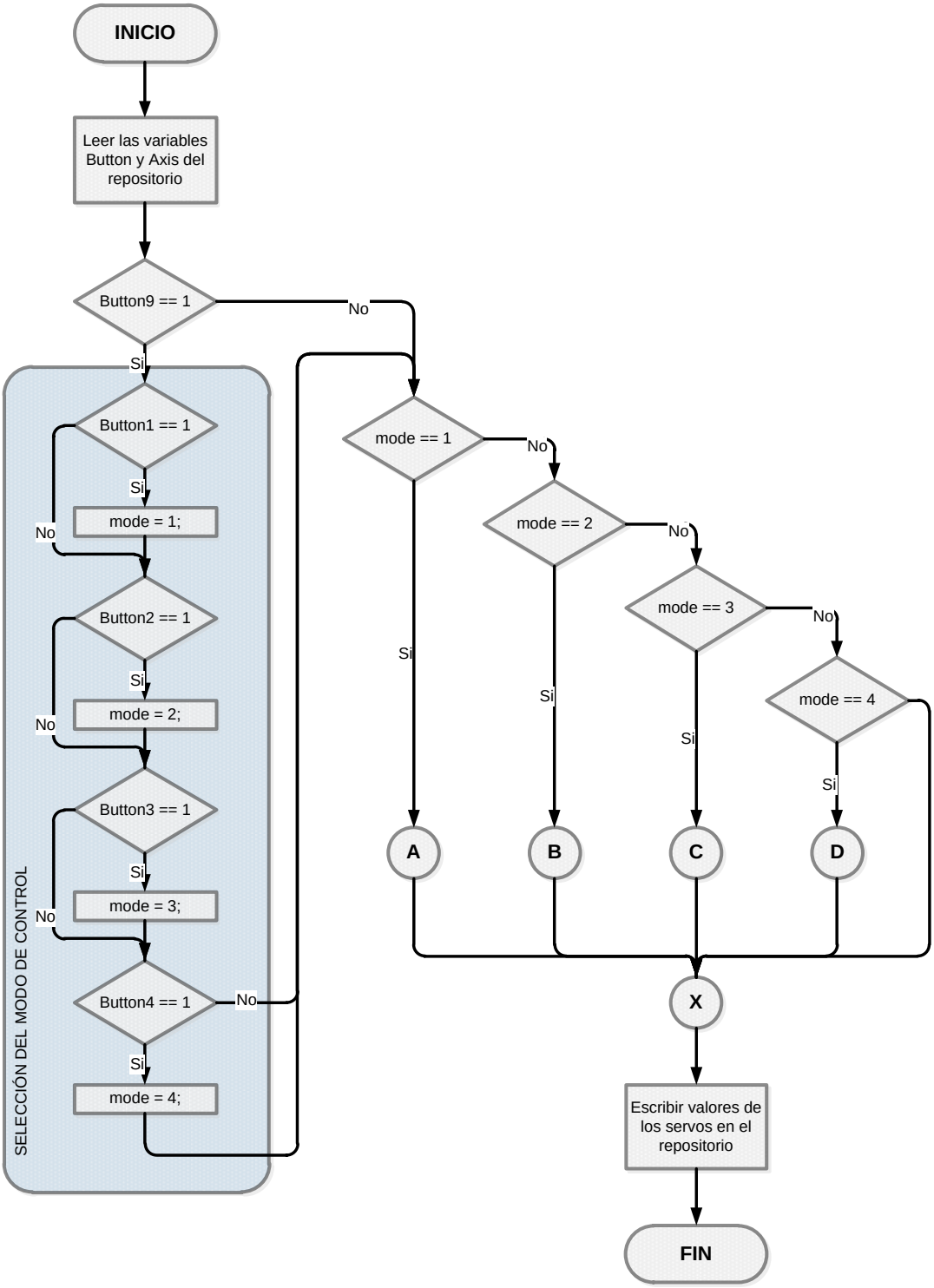


Figura 4. 8: Diagrama de flujo modulo *Gamepad2FTrack*. Esquema general.

El modulo *Gamepad2Ftrack* es una de las partes fundamentales de la aplicación, y también una de las más complejas. Ya que va a ser la que marque el carácter y funcionalidad del programa.

Tal y como se puede observar en la Figura 4.8 este modulo va a resultar mucho más complejo en su estructura que el resto. Además, debido a su gran extensión se ha optado por ser presentado en diversos diagramas de flujo. Es decir, en la Figura 4.8 se presenta el diagrama general de dicho modulo, pero tal y como se aprecia ciertas partes de este han sido reducidas a una referencia con el objetivo de ser desarrolladas en las siguientes hojas. Estas referencias vienen dadas por las siguientes letras: A, B, C, D y X que se corresponden respectivamente con las Figuras 4.9, 4.10, 4.11, 4.12 y 4.13.

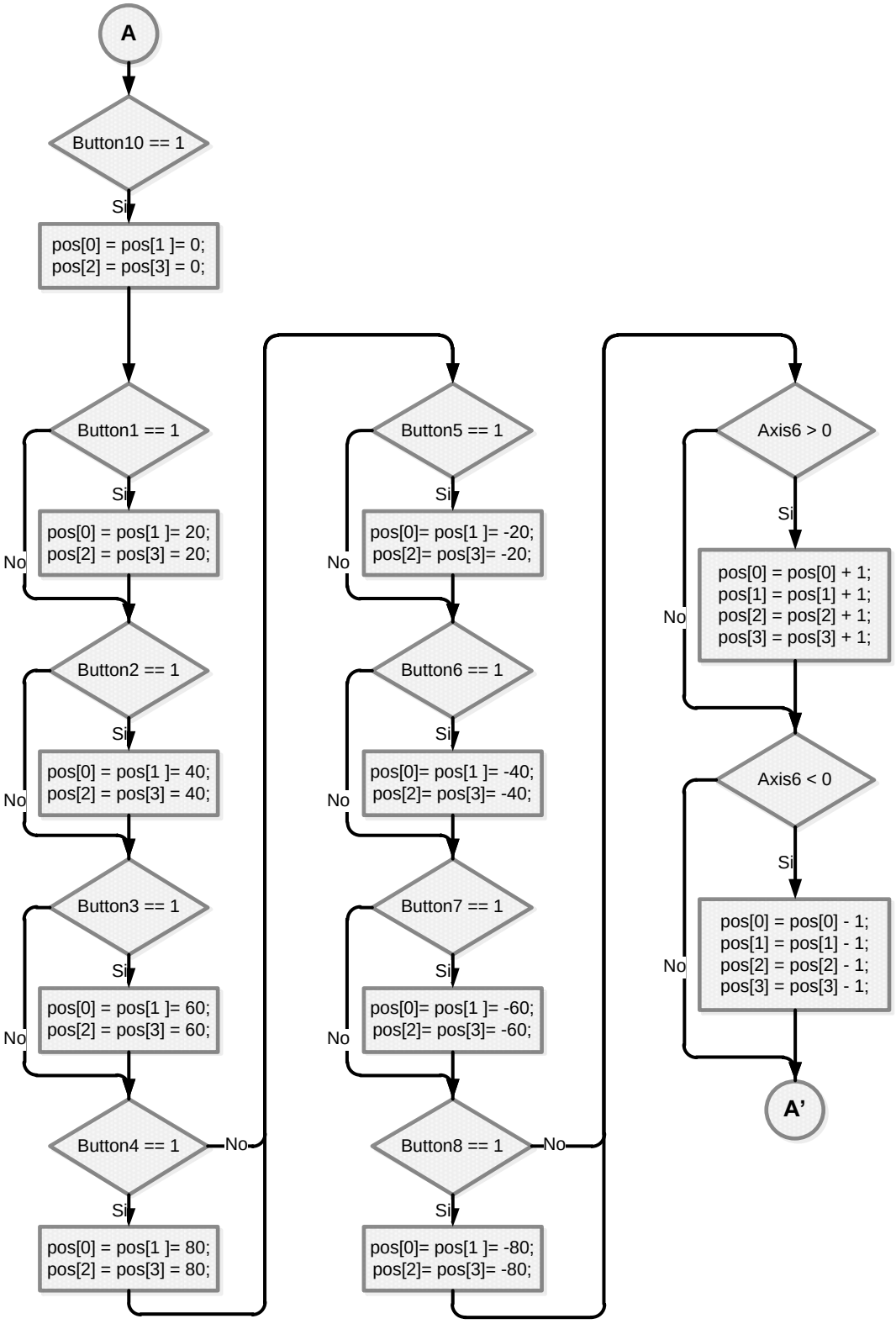


Figura 4. 9: Diagrama de flujo modulo *Gamepad2FTrack*. Sub-esquema A.

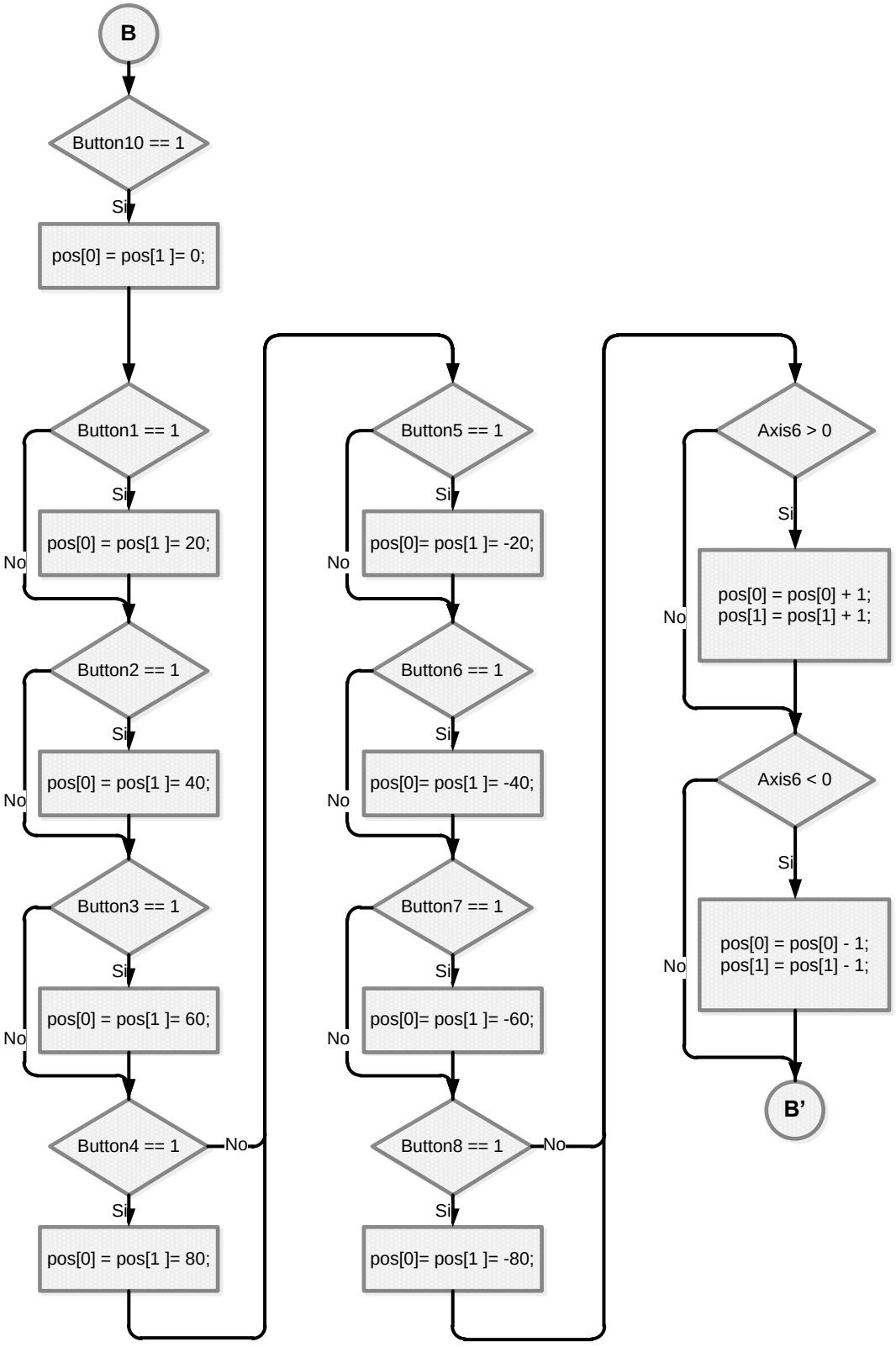


Figura 4. 10: Diagrama de flujo modulo *Gamepad2FTrack*. Sub-esquema B.

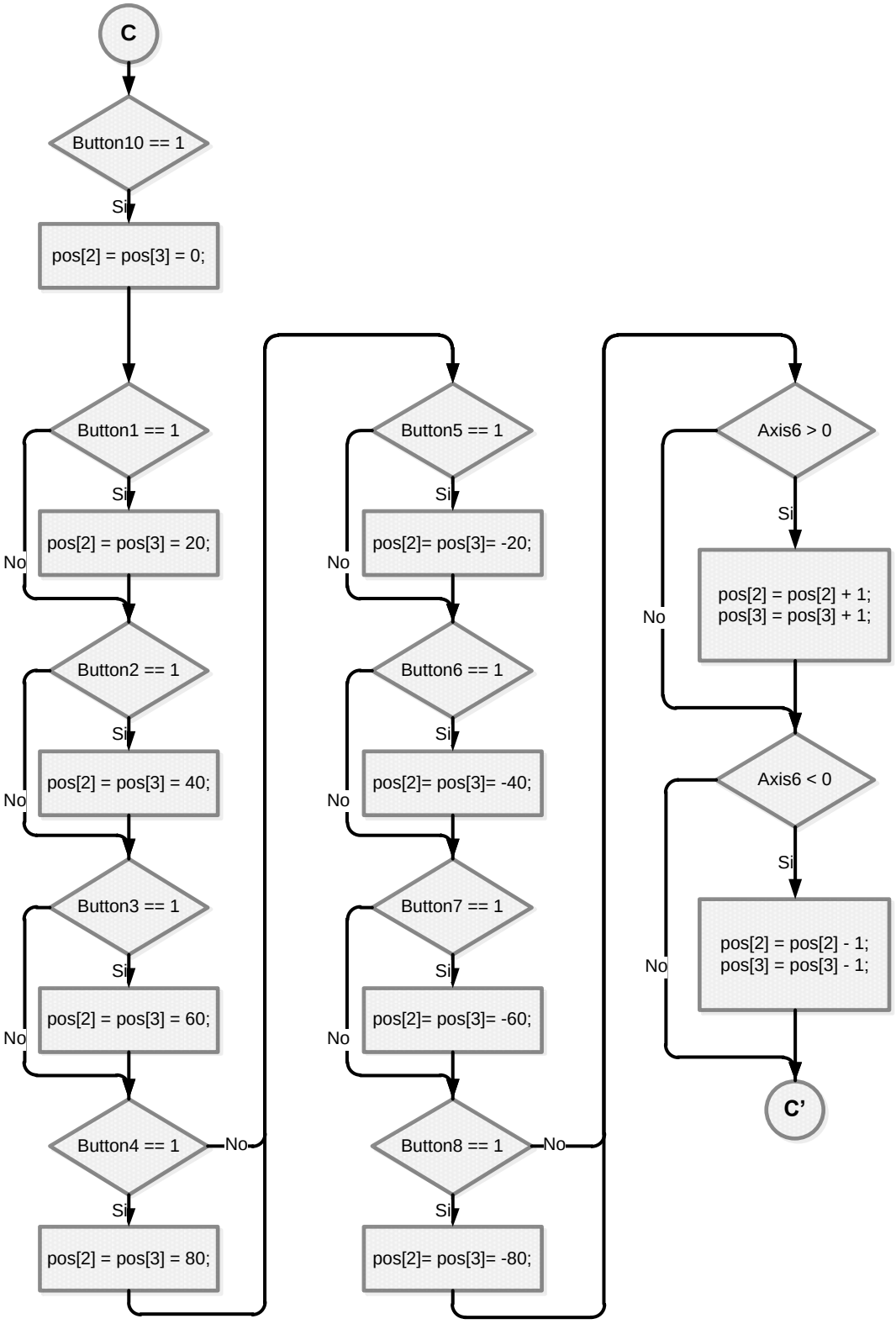


Figura 4. 11: Diagrama de flujo modulo *Gamepad2FTrack*. Sub-esquema C.

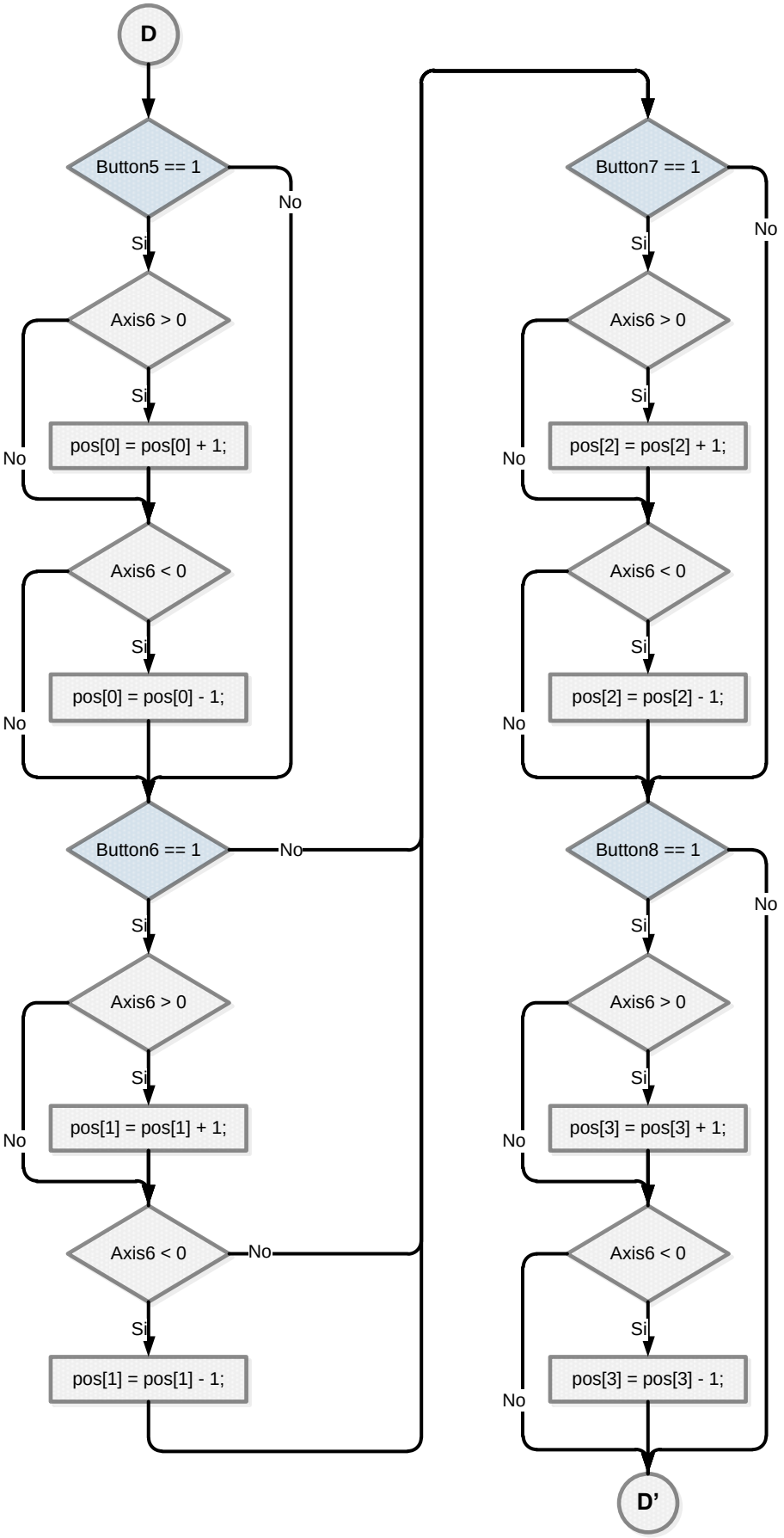


Figura 4. 12: Diagrama de flujo modulo *Gamepad2FTrack*. Sub-esquema D.

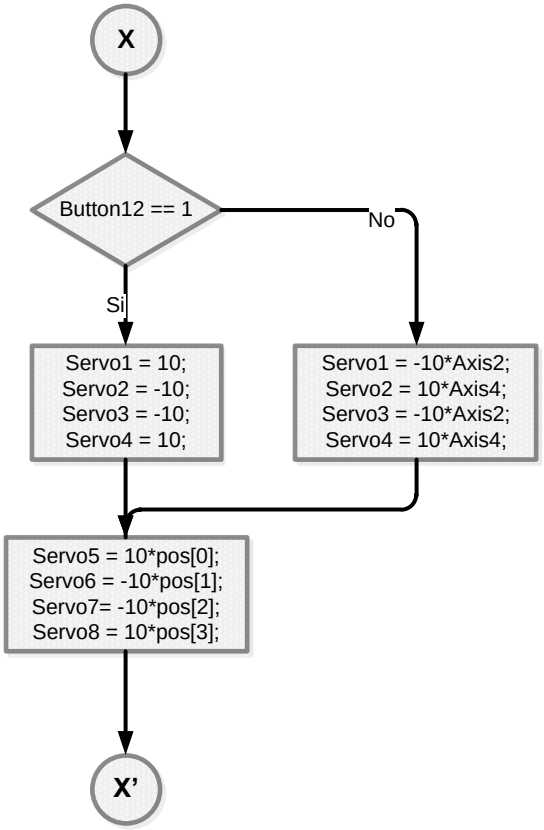


Figura 4. 13: Diagrama de flujo modulo Gamepad2FTrack. Sub-esquema X.

- ***FTrackClientModule***

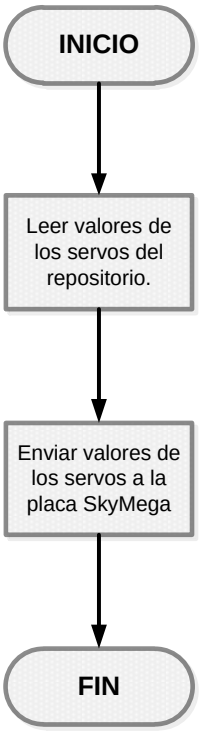


Figura 4. 14: Diagrama de flujo modulo *FTrackClient*.

4.3 Diseño de la aplicación. Robot Orugator.

Una vez diseñada la aplicación para el robot F-Track resulta mucho más simple y rápido el diseño y desarrollo de una aplicación para otro robot como es el Orugator. Esto es debido al empleo del software modular OpenRDK que hace que ciertas partes del código sean totalmente reutilizables por la nueva aplicación. Así los módulos *GamepadReaderModule* y *FTrackClientModule* serán reutilizados, siendo únicamente necesario crear el modulo *Gamepad2OrugatorModule*.

Este nuevo modulo que ha de ser programado tiene como objetivo determinar la velocidad o posición de los servos en función de los valores del gamepad. Es decir, la misión va a ser la misma que la de *Gamepad2FTrackModule*, pero el código va a ser distinto ya que mientras el robot F-Track disponía de 8 servomotores, el robot Orugator dispone únicamente de 2 servos. Por tanto será necesario diseñar un modulo que en función de los valores del gamepad determine los valores de velocidad de los servos 1 y 2, y no de los servos 1, 2 ,3… 7 y 8.

En la siguiente figura (Figura 4.15) se muestra la estructura del agente RDK que emplea los antiguos módulos *GamepadReaderModule* y *FTrackClientModule* y el nuevo modulo *Gamepad2OrugatorModule*. El proceso es el siguiente: primero el agente recupera los datos del gamepad, lo siguiente que hace el módulo *Gamepad2OrugatorModule* es determinar los valores de control de los servos y por último se envían las órdenes de movimiento al robot.

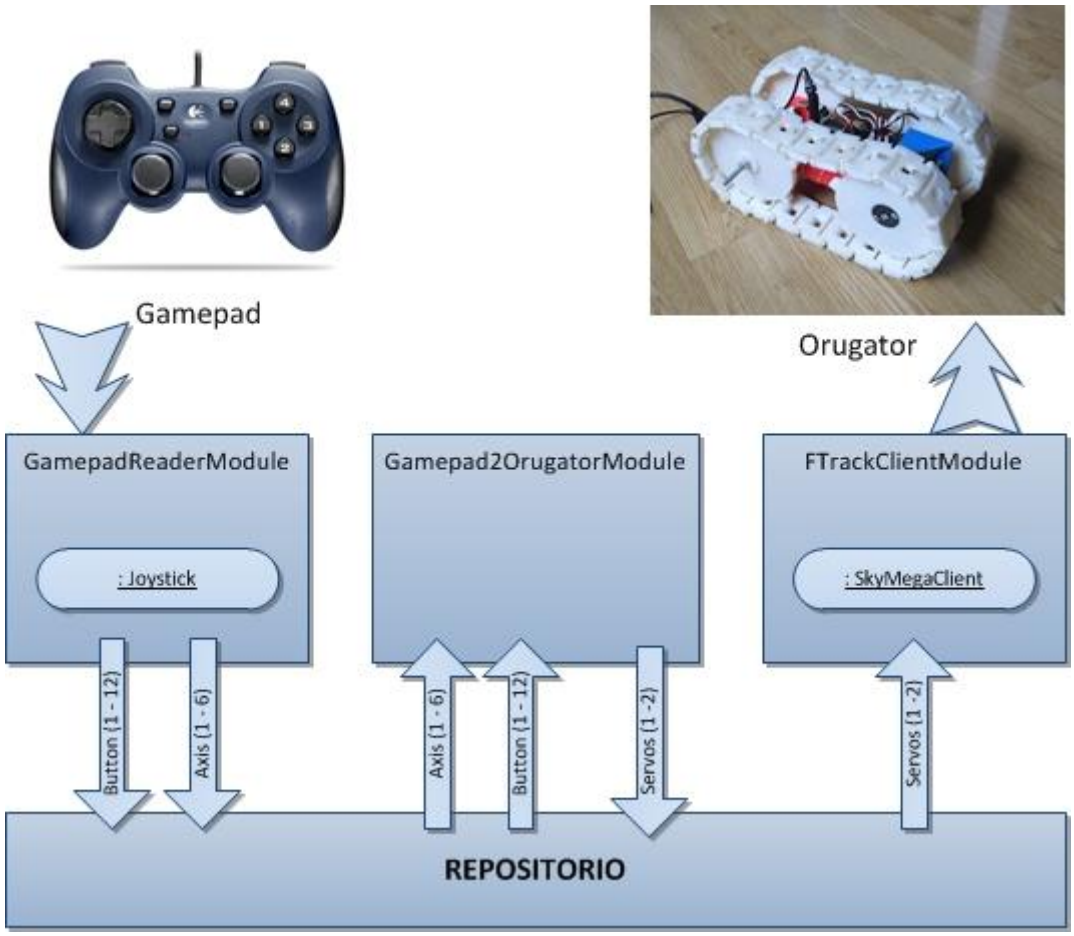


Figura 4. 15: Estructura del agente RDK para el robot Orugator

Tal y como se puede observar en la imagen (Figura 4.15) las únicas diferencias entre el agente RDK para el robot Orugator y el agente para el F-Track radican en el intercambio del modulo Gamepad2Ftrack por el modulo Gamepad2Orugator, y en que este ultimo solo escribe en el repositorio el valor de los servos 1 y 2.

La Figura 4.16 da una visión esquemática más precisa de las operaciones llevadas a cabo por cada modulo y la comunicación e interacción entre estos. Mención especial requiere el modulo *Gamepad2Orugator* en el cual se observa que dependiendo del modo de control seleccionado (modo 1, 2 o 3) la asignación de valores de los servos se realizara de distinta forma. Todo lo relacionado a los modos de control se estudiara más profundamente en el capítulo 5 *Descripción de la aplicación*.

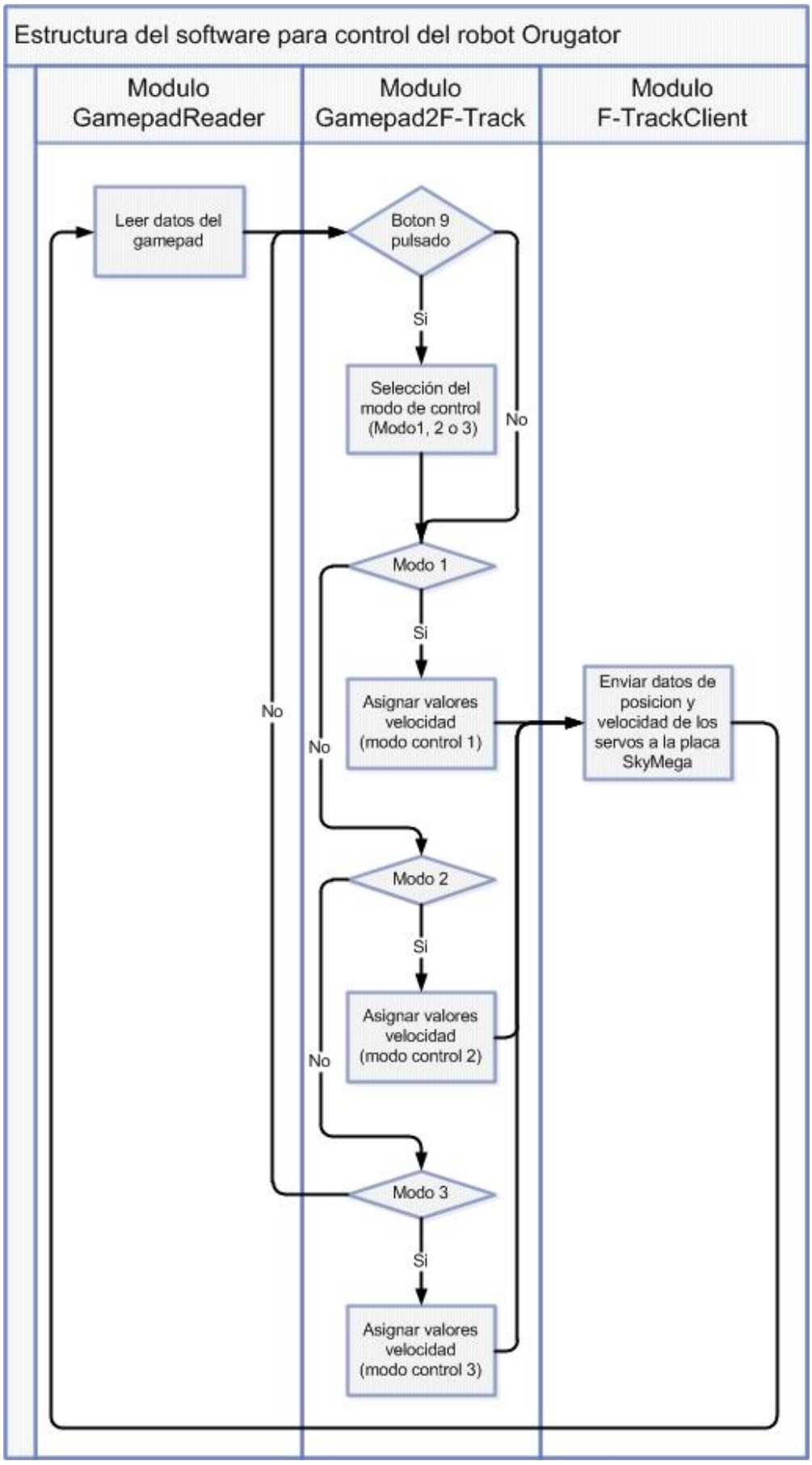


Figura 4. 16: Estructura del software para control del robot Orugator

4.3.1 Diagramas de flujo.

- ***Gamepad2OrugatorModule***

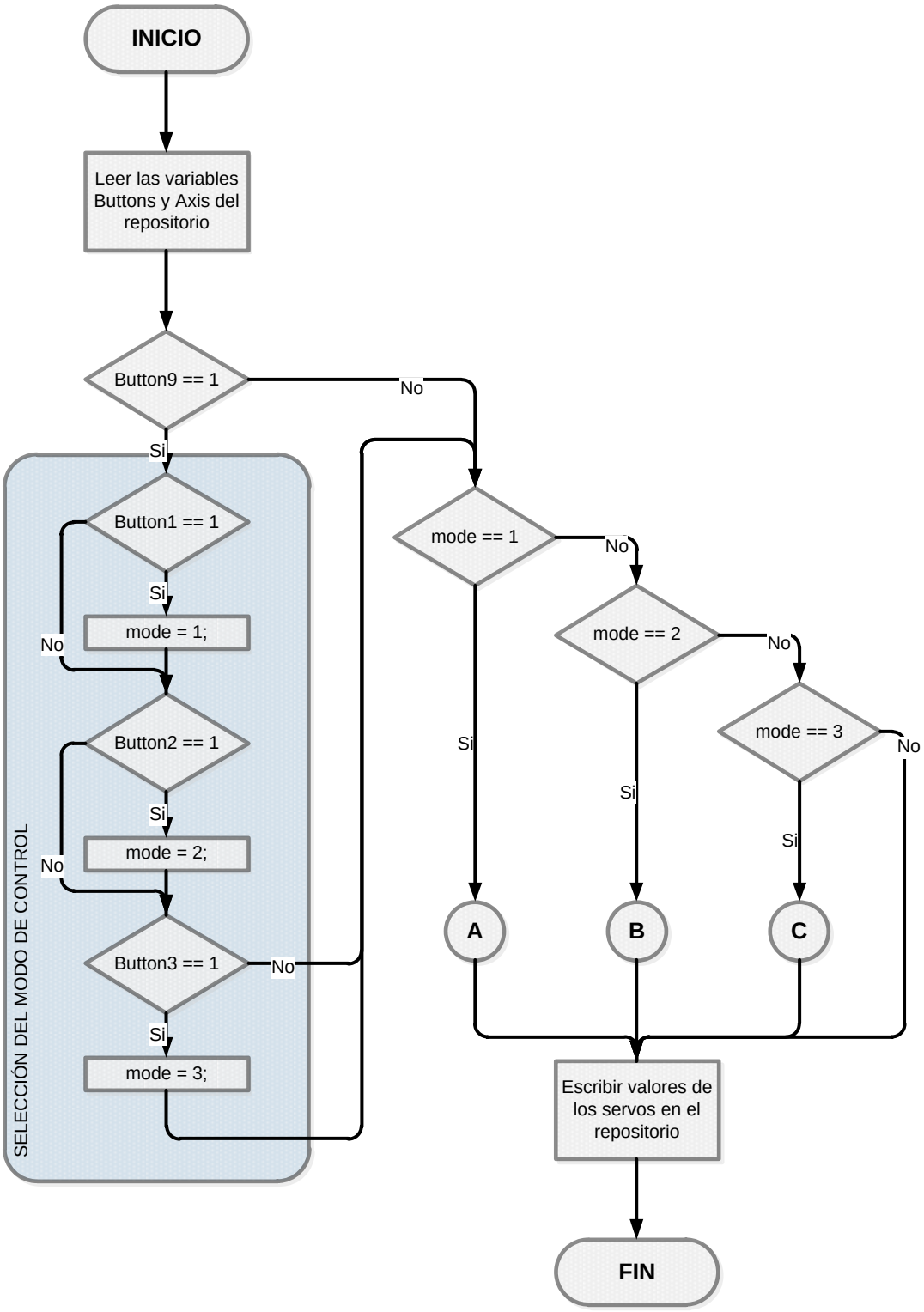


Figura 4. 17: Diagrama de flujo modulo *Gamepad2Orugator*. Esquema general.

El diagrama de flujo del *Gamepad2OrugatorModule* posee una estructura similar a la del *Gamepad2FTrackModule* y presenta una gran extensión. Por eso se ha optado por ser representado en varios trozos. Es decir, en la Figura 4.17 se puede ver el diagrama general de dicho modulo, pero tal y como se aprecia ciertas partes de este han sido reducidas a una referencia con el objetivo de ser desarrolladas en las siguientes hojas. Estas referencias vienen dadas por las siguientes letras: A, B y C que se corresponden respectivamente con las Figuras 4.18, 4.19 y 4.20.

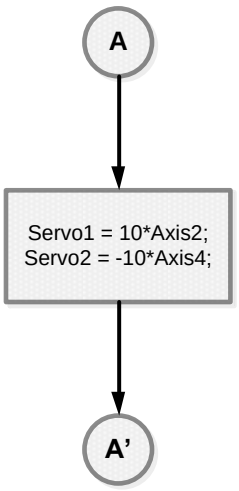


Figura 4. 18: Diagrama de flujo modulo *Gamepad2FTrack*. Sub-esquema A.

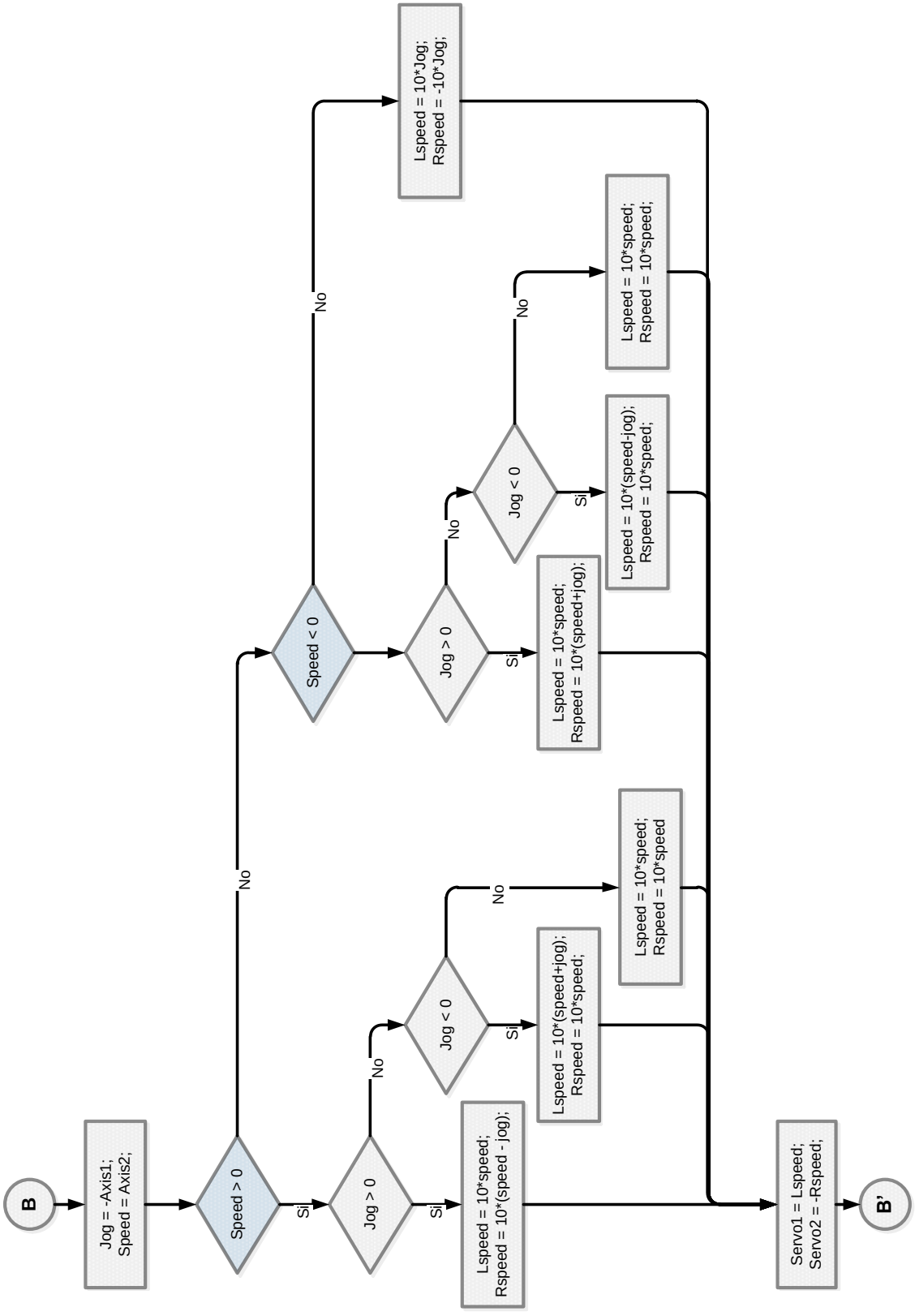


Figura 4. 19: Diagrama de flujo modulo *Gamepad2FTrack*. Sub-esquema B.

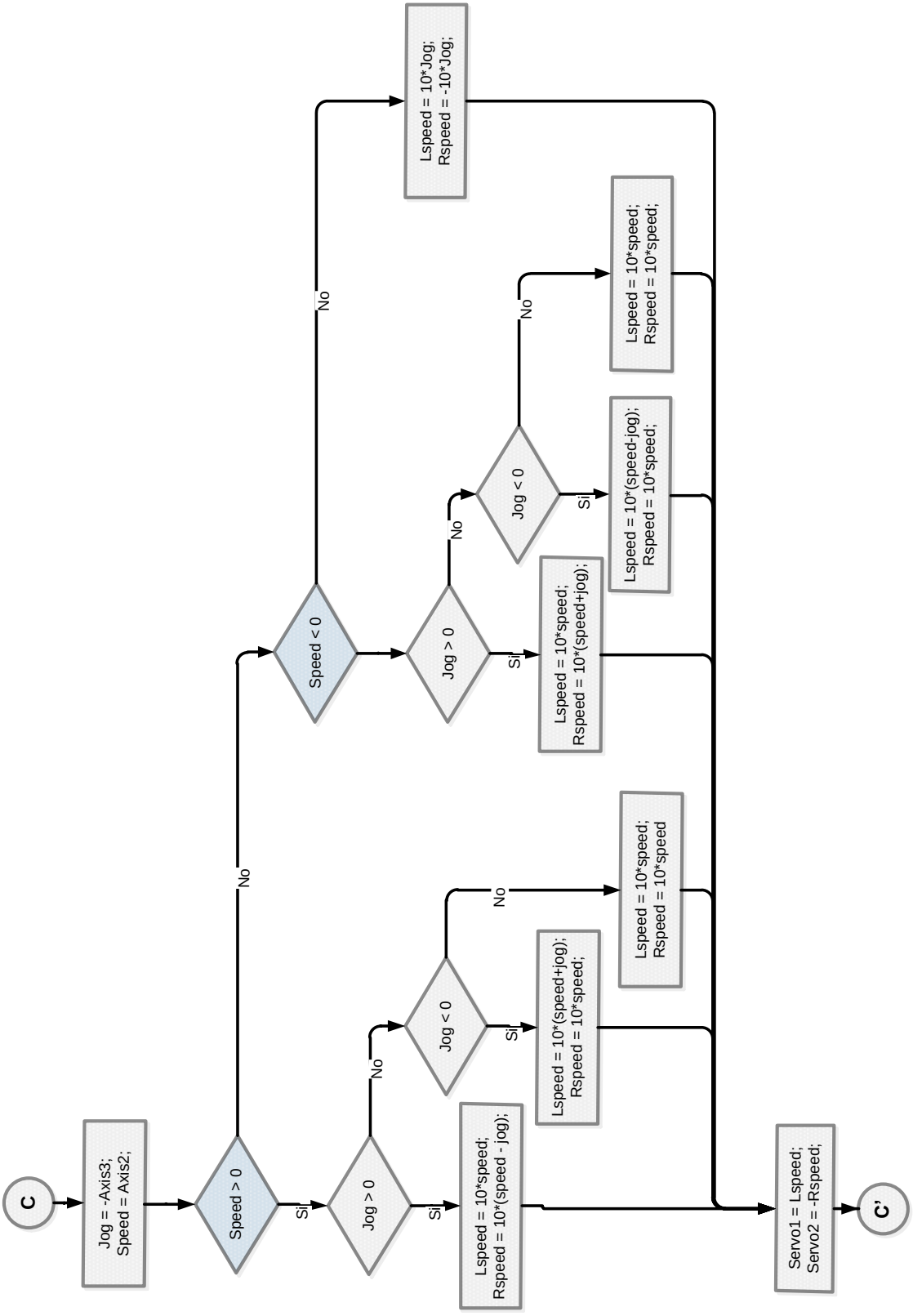


Figura 4. 20: Diagrama de flujo modulo *Gamepad2FTrack*. Sub-esquema C.

Capítulo 5

Descripción de la aplicación

A lo largo de este capítulo se procederá a describir la aplicación, así como a explicar detenidamente los diferentes modos de control y cuál es el más idóneo para cada situación. Además se aportaran todos los conocimientos necesarios para que cualquier usuario de los robots imprimibles F-Track y Orugator pueda manejarlos a través del *Gamepad*, es decir, se explicara la función de cada uno de los botones o *joystick* del periférico de control.

5.1 Controles del robot F-Track.

Antes de comenzar con la descripción de la aplicación para el robot F-Track y las nociones básicas para su control, se va a realizar un breve repaso de la estructura de este para que quede claro que partes son las que se accionan cuando se pulsan los diferentes botones.

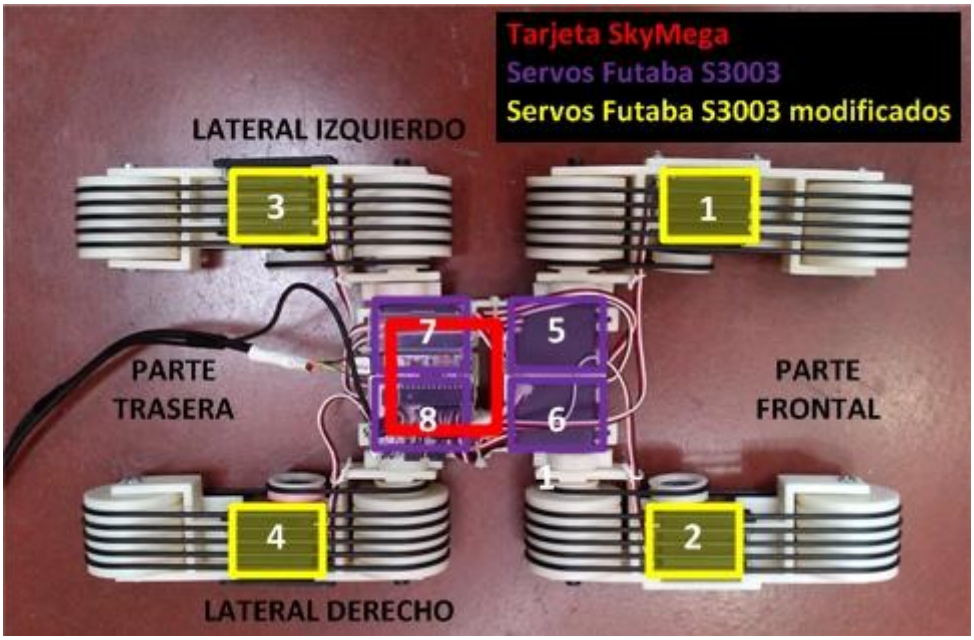


Figura 5. 1: Estructura robot F-Track

En la imagen superior (Figura 5.1) se aprecia que el robot F-Track se divide en dos partes simétricas la una respecto a la otra. Estas dos partes son la Frontal y la Trasera, siendo la primera la que marca el sentido positivo de desplazamiento del vehículo.

La parte Frontal está compuesta por dos ruedas de oruga, la número 1 (lateral izquierdo) y la número 2 (lateral derecho), cuyo encargado del movimiento son los servos modificados para giro continuo del mismo número. Además dichas dos orugas están unidas al cuerpo principal del robot a través de las articulaciones número 5 y 6

controladas a su vez por los servos sin modificar del mismo número. Estos servos permiten el movimiento relativo de las patas de oruga respecto al cuerpo central, logrando posicionar a estas en un rango comprendido entre los 90º y -90º. Donde la posición 0º es la correspondiente a la pata situada en el mismo plano que la parte central, y un ángulo positivo implica que el extremo de la pata se encuentra por encima de dicho plano.

En cuanto a la parte Trasera se puede aplicar lo mismo que se ha dicho para la parte Frontal por ser ambas simétricas. Lo único que cambia es la nomenclatura de sus partes, siendo la rueda de oruga del lateral izquierdo la número 3 unida al cuerpo central a través de la articulación Nº7 y la del lateral derecho la número 4 unida a través de la articulación Nº8.

Ahora que poseemos una forma de referirnos a las diferentes partes móviles del robot imprimible F-Track se va a proceder a explicar el control de este.

Tal y como se ha venido explicando a lo largo de esta tesis un esquema general del sistema de control completo sería el siguiente:

Figura 5. 2: Esquema del sistema de control

El usuario interacciona con el *Gamepad* pulsando sus botones y moviendo sus *joysticks*. El PC recibe los datos del *Gamepad*, los procesa y en función de estos envía diferentes órdenes al robot. El robot atiende las órdenes y realiza los movimientos consecuentes. Este sistema se ve realimentado por el usuario, el cual está en contacto visual con el robot, tomando las decisiones para los siguientes movimientos en función de lo observado.

En cuanto al Gamepad (Ver Figura 5.3), este presenta las siguientes características:

- 12 botones digitales a los que haremos referencia por su número, es decir, botón 1 o B1, botón 2 o B2, y así sucesivamente hasta el botón 12 o B12. Cabe destacar que los botones del 5 al 6 (B5, B6, B7 y B8) también recibirán el nombre de botones traseros.

- 2 joysticks analógicos. El joystick izquierdo compuesto por los ejes 1 y 2 (Axis1 y Axis2 en la Figura 5.3) y el joystick derecho por los ejes 3 y 4 (Axis3 y Axis4).
- 1 pad octodireccional o también llamado cruceta formado por los ejes 5 y 6 (Axis5 y axis6).

Figura 5. 3: Mapeado del Gamepad

Como el robot F-Track presenta gran versatilidad de movimientos y el periférico de control solo consta de 12 botones, 2 joysticks y un pad octodireccional, se ha optado por desarrollar varios modos de control. Dichos modos de control presentan la característica de que un mismo botón en función del modo de control seleccionado presentara comportamientos diferentes. Por tanto mediante el empleo y combinación de diferentes modos de control va a ser posible llevar a cabo cualquier tipo de movimiento.

5.2 Modos de control del robot F-Track.

Se definen cuatro modos de control para el robot F-Track. Para cambiar el modo de funcionamiento simplemente hay que dejar pulsado el botón 9, y sin soltar este pulsar el botón del mismo número del modo de control que se quiere emplear. Es decir:

- B9 + B1. Mando configurado en modo de control 1. Es el modo por defecto.
- B9 + B2. Modo de control 2.
- B9 + B3. Modo de control 3.
- B9 + B4. Modo de control 4.

A continuación se procede a explicar el comportamiento y control de cada uno de los modos.

5.2.1 Modo de control 1.

Es el modo de control por defecto, es decir, el que al lanzar la aplicación se está ejecutando. Se caracteriza por que el control de posicionamiento de las cuatro patas u orugas se realiza de forma conjunta. Por ejemplo si se pulsa el botón B1 las cuatro patas se posicionan a 20º respecto al plano del cuerpo central (Ver Figura 5.4, imagen superior derecha).

Figura 5. 4: Ejemplos de posicionamiento conjunto de las patas u orugas del robot F-Track

En la Figura 5.4 se observa de izquierda a derecha, y de arriba abajo a modo de ejemplo lo que sucede cuando en el modo de control 1 se pulsa el botón B10, el B1, B2 o B8. Cabe destacar que la posición por defecto es con las patas u orugas a 0º, es decir, cuando se lanza la ejecución de la aplicación o se reinicia esta, el robot adquirirá esta configuración como posición de inicio.

A continuación se recogen en la siguiente tabla las diferentes operaciones que llevan a cabo cada uno de los botones o joysticks en el modo de control 1:

Modo 1		
Botón	Acción	
B1	Las 4 patas u orugas se mueven a la posición 20º.	
B2	Las 4 patas u orugas se mueven a la posición 40º.	
B3	Las 4 patas u orugas se mueven a la posición 60º.	
B4	Las 4 patas u orugas se mueven a la posición 80º.	
B5	Las 4 patas u orugas se mueven a la posición -20º.	
B6	Las 4 patas u orugas se mueven a la posición -40º.	
B7	Las 4 patas u orugas se mueven a la posición -60º.	
B8	Las 4 patas u orugas se mueven a la posición -80º.	
B9	Manteniendo pulsado mientras se pulsa B1 o B2 o B3 o B4, cambia al modo de control 1, 2, 3 o 4 respectivamente.	
B10	Las 4 patas u orugas se mueven a la posición 0º.	
B11	--- *No asignado* ---	
B12	Mientras este pulsado las ruedas de oruga delanteras se mueven en sentido contrario a las traseras. Empleado en superficies planas cuando el robot se quiere elevar sobre sus patas, es decir, cuando en superficies planas se pulsa alguno de los siguientes botones: B5, B6, B7 o B8.	
Cruceta	Acción	
Axis5 ↕	↑: Las 4 patas u orugas incrementan su posición en 1º. ↓: Las 4 patas u orugas disminuyen su posición en 1º.	
Axis6 ↔	--- *No asignado* ---	
Joystick		Acción
Izquierdo	Axis2 ↕	↑: Las ruedas de oruga del lateral izquierdo (1 y 3, ver Figura5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral izquierdo giran en sentido negativo.
	Axis1 ↔	--- *No asignado* ---
Derecho	Axis4 ↕	↑: Las ruedas de oruga del lateral derecho (2 y 4, ver Figura5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral derecho giran en sentido negativo.
	Axis3 ↔	--- *No asignado* ---

Tabla 5. 1: F-Track. Controles Modo 1

Los controles del modo de control 1 recogidos en la Tabla 5.1 se pueden resumir diciendo que mientras los botones frontales (B1, B2, B3 y B4, Fig.5.3) posicionan las orugas en ángulos positivos múltiplos de 20º, los botones traseros (B5, B6, B7 y B8) los posicionan en ángulos negativos también múltiplos de 20º. Y en caso de que se desee un ángulo intermedio basta con dejar pulsada la cruceta hasta que se alcance.

En cuanto a los movimientos que permiten el desplazamiento del robot adelante atrás además de girar, vienen gobernados por el joystick izquierdo y derecho, cada uno de los

cuales es responsable del movimiento de las ruedas del lateral izquierdo y derecho respectivamente. Por lo que los movimientos que se podrán realizar son:

- Si los dos joystick se mueven conjuntamente hacia arriba el robot se desplazara hacia delante.
- Si los dos joystick se mueven conjuntamente hacia abajo el robot se desplazara hacia atrás.
- Si el joystick izquierdo se mueve hacia arriba mientras que el derecho hacia abajo el robot gira hacia la derecha.
- Si el joystick izquierdo se mueve hacia abajo mientras que el derecho hacia arriba el robot gira hacia la izquierda.

Cabe destacar que como los joysticks son analógicos estos van a arrojar un valor mayor cuanto más disten de su posición de reposo, lo que permite que podamos controlar la velocidad de los anteriores cuatro movimientos. Es decir, el robot avanzara o girara más deprisa cuanto más hacia arriba o abajo se empujen las palancas del *gamepad*.

Mención aparte requiere el comportamiento asociado al botón B12. Este como ya se ha comentado en la Tabla 5.1, mientras se mantenga pulsado, las ruedas de oruga delanteras giraran en sentido contrario a las traseras. La finalidad de este comportamiento es la de ayudar al robot cuando se encuentra en superficies planas horizontales a posicionar las patas u orugas en ángulos negativos. Es decir, permitir que el robot se incorpore sobre sus patas.

Por tanto se recomienda mantener pulsado el botón B12 cuando el robot se encuentre situado sobre una superficie plana y horizontal y se lleve a cabo alguna de las siguientes acciones:

- Pulsar alguno de los botones traseros del *gamepad* (B5, B6, B7 o B8).
- Mantener pulsada la cruceta hacia abajo cuando el posicionamiento de las orugas es igual a 0º o inferior.

Este movimiento fue incorporado a los controles cuando en los primeros ensayos con el robot se aprecio que los servos encargados de mover las articulaciones (servos 5, 6, 7 y 8, Fig.5.1) no poseían el par necesario para elevar el robot sobre sus patas. Tras este acontecimiento se presentaron dos posibles soluciones: Cambiar dichos servos por unos más potentes, lo que requeriría del rediseño del robot, o crear un movimiento que facilitase la tarea a los servos. Se acabo optando por la segunda opción ya que esta tesis no abarca el tema del diseño de los robots.

5.2.2 Modo de control 2.

El modo de control 2, al que se puede acceder mediante la combinación de botones B9+B2, se caracteriza por controlar el posicionamiento de las dos patas u orugas delanteras de forma conjunta. También se caracteriza por no poderse controlar el posicionamiento de las patas traseras, quedando estas ancladas en la posición que ocupasen antes de pasar al modo de control 2.

Modo 2		
Botón	Acción	
B1	Las 2 patas u orugas delanteras se mueven a la posición 20º.	
B2	Las 2 patas u orugas delanteras se mueven a la posición 40º.	
B3	Las 2 patas u orugas delanteras se mueven a la posición 60º.	
B4	Las 2 patas u orugas delanteras se mueven a la posición 80º.	
B5	Las 2 patas u orugas delanteras se mueven a la posición -20º.	
B6	Las 2 patas u orugas delanteras se mueven a la posición -40º.	
B7	Las 2 patas u orugas delanteras se mueven a la posición -60º.	
B8	Las 2 patas u orugas delanteras se mueven a la posición -80º.	
B9	Manteniendo pulsado mientras se pulsa B1 o B2 o B3 o B4, cambia al modo de control 1, 2, 3 o 4 respectivamente.	
B10	Las 2 patas u orugas delanteras se mueven a la posición 0º.	
B11	--- *No asignado* ---	
B12	Mientras este pulsado las ruedas de oruga delanteras se mueven en sentido contrario a las traseras.	
Cruceta	Acción	
Axis5 ↕	↑: Las 2 patas u orugas delanteras incrementan su posición en 1º. ↓: Las 2 patas u orugas delanteras disminuyen su posición en 1º.	
Axis6 ↔	--- *No asignado* ---	
Joystick		Acción
Izquierdo	Axis2 ↕	↑: Las ruedas de oruga del lateral izquierdo (1 y 3, Fig.5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral izquierdo giran en sentido negativo.
	Axis1 ↔	--- *No asignado* ---
Derecho	Axis4 ↕	↑: Las ruedas de oruga del lateral derecho (2 y 4, Fig.5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral derecho giran en sentido negativo.
	Axis3 ↔	--- *No asignado* ---

Tabla 5. 2: F-Track. Controles Modo 2

Como resumen de la Tabla 5.2 se puede decir que los controles son iguales que para el modo de control 1, con la única diferencia de que los botones que en el modo 1 controlaban el posicionamiento conjunto de las cuatro patas, aquí solo van a controlar las delanteras.

Cabe destacar que en este modo no va a ser posible modificar el posicionamiento de las orugas traseras, para lograr tal fin será necesario ejecutar el modo de control 3.

5.2.3 Modo de control 3.

Es el modo complementario del modo 2. Si en este se controlaba el posicionamiento conjunto de las patas u orugas delanteras independientemente de la posición de las traseras, en el modo 3 se controlara la posición conjunta de las traseras con independencia de las delanteras

Cabe recordar que para acceder al modo de control 3 es necesario pulsar la combinación de botones B9 + B3.

Modo 3		
Botón	Acción	
B1	Las 2 patas u orugas traseras se mueven a la posición 20º.	
B2	Las 2 patas u orugas traseras se mueven a la posición 40º.	
B3	Las 2 patas u orugas traseras se mueven a la posición 60º.	
B4	Las 2 patas u orugas traseras se mueven a la posición 80º.	
B5	Las 2 patas u orugas traseras se mueven a la posición -20º.	
B6	Las 2 patas u orugas traseras se mueven a la posición -40º.	
B7	Las 2 patas u orugas traseras se mueven a la posición -60º.	
B8	Las 2 patas u orugas traseras se mueven a la posición -80º.	
B9	Manteniendo pulsado mientras se pulsa B1 o B2 o B3 o B4, cambia al modo de control 1, 2, 3 o 4 respectivamente.	
B10	Las 2 patas u orugas traseras se mueven a la posición 0º.	
B11	--- *No asignado* ---	
B12	Mientras este pulsado las ruedas de oruga delanteras se mueven en sentido contrario a las traseras.	
Cruceta	Acción	
Axis5 ↕	↑: Las 2 patas u orugas traseras incrementan su posición en 1º. ↓: Las 2 patas u orugas traseras disminuyen su posición en 1º.	
Axis6 ↔	--- *No asignado* ---	
Joystick		Acción
Izquierdo	Axis2 ↕	↑: Las ruedas de oruga del lateral izquierdo (1 y 3, ver Fig.5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral izquierdo giran en sentido negativo.
	Axis1 ↔	--- *No asignado* ---
Derecho	Axis4 ↕	↑: Las ruedas de oruga del lateral derecho (2 y 4, ver Fig.5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral derecho giran en sentido negativo.
	Axis3 ↔	--- *No asignado* ---

Tabla 5. 3: F-Track. Controles Modo 3

5.2.4 Modo de control 4.

El modo de control 4 se caracteriza por permitir el control individual de posicionamiento de cada una de las cuatro patas u orugas del robot F-Track. Es decir, esta configuración va a permitir posicionar cada una de las patas en un ángulo distinto y no como sucedía con los anteriores modos en los que las orugas se posicionaban conjuntamente a un mismo ángulo.

Cabe recordar que para acceder al modo de control 4 es necesario pulsar la combinación de botones B9 + B4.

Modo 4		
Botón	Acción	
B1	--- *No asignado* ---	
B2	--- *No asignado* ---	
B3	--- *No asignado* ---	
B4	--- *No asignado* ---	
B5	Mientras este pulsado, pata u oruga nº 1 (Fig.5.1) seleccionada.	
B6	Mientras este pulsado, pata u oruga nº 2 (Fig.5.1) seleccionada.	
B7	Mientras este pulsado, pata u oruga nº 3 (Fig.5.1) seleccionada.	
B8	Mientras este pulsado, pata u oruga nº 4 (Fig.5.1) seleccionada.	
B9	Manteniendo pulsado mientras se pulsa B1 o B2 o B3 o B4, cambia al modo de control 1, 2, 3 o 4 respectivamente.	
B10	--- *No asignado* ---	
B11	--- *No asignado* ---	
B12	Mientras este pulsado las ruedas de oruga delanteras se mueven en sentido contrario a las traseras.	
Cruceta	Acción	
Axis5 ↕	↑: Las patas u orugas seleccionadas por los botones traseros (B5, B6, B7 y B8) incrementan su posición en 1º. ↓: Las patas u orugas seleccionadas disminuyen su posición en 1º.	
Axis6 ↔	--- *No asignado* ---	
Joystick		Acción
Izquierdo	Axis2 ↕	↑: Las ruedas de oruga del lateral izquierdo (1 y 3, ver Fig.5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral izquierdo giran en sentido negativo.
	Axis1 ↔	--- *No asignado* ---
Derecho	Axis4 ↕	↑: Las ruedas de oruga del lateral derecho (2 y 4, ver Fig.5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral derecho giran en sentido negativo.
	Axis3 ↔	--- *No asignado* ---

Tabla 5. 4: F-Track. Controles Modo 4

Se pueden resumir los controles del modo 4 recogidos en la Tabla 5.4 diciendo que en lo referente a los movimientos que provocan que el robot se desplace y pueda girar los

controles son los mismos que para los anteriores 3 modos. Pero en lo relativo al posicionamiento de las patas u orugas los controles han cambiado.

Para ajustar la posición de una de las orugas habrá que dejar pulsado el botón correspondiente a la selección de dicha oruga (B5, B6, B7 o B8), y mientras se realiza esto pulsar la cruceta hacia arriba o abajo hasta que su posición sea la deseada.

Cabe destacar que está permitido modificar la posición de varia orugas a la vez. Simplemente basta con mantener pulsados los botones de selección necesarios mientras se presiona la cruceta hacia arriba o abajo. Cualquier combinación de orugas es válida.

Por ejemplo, si se mantiene pulsado los botones B5 y B8 (selección de oruga delantera izquierda y oruga trasera derecha) y se presiona la cruceta hacia arriba, el posicionamiento de las orugas delantera izquierda y trasera derecha incrementaran su posición en 1º.

Figura 5. 5: Robot F-Track en modo de control 4

Este tipo de control va a resultar muy útil, cuando el robot se enfrenta a obstáculos de superficie irregular, permitiendo que se realice un ajuste fino de cada una de sus patas con el objetivo de aumentar los puntos de apoyo, y por tanto su estabilidad.

5.2.5 Controles comunes.

Tal y como se ha podido venir apreciando con el estudio de los múltiples modos de control. Existen diferentes botones o combinaciones de ellos que tienen asignado el mismo comportamiento independientemente del modo de control que se esté ejecutando.

Estos controles comunes se recogen en la siguiente tabla:

Controles comunes independientemente del número de modo		
Botón	Acción	
B1	Si se presiona mientras B9 pulsado se cambia al modo de control 1.	
B2	Si se presiona mientras B9 pulsado se cambia al modo de control 2.	
B3	Si se presiona mientras B9 pulsado se cambia al modo de control 3.	
B4	Si se presiona mientras B9 pulsado se cambia al modo de control 4.	
B5	----------	
B6	----------	
B7	----------	
B8	----------	
B9	Manteniendo pulsado mientras se pulsa B1 o B2 o B3 o B4, cambia al modo de control 1, 2, 3 o 4 respectivamente.	
B10	----------	
B11	----------	
B12	Mientras este pulsado las ruedas de oruga delanteras se mueven en sentido contrario a las traseras.	
Cruceta	Acción	
Axis5 ↕	----------	
Axis6 ↔	----------	
Joystick		Acción
Izquierdo	Axis2 ↕	↑: Las ruedas de oruga del lateral izquierdo (1 y 3, ver Fig.5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral izquierdo giran en sentido negativo.
	Axis1 ↔	----------
Derecho	Axis4 ↕	↑: Las ruedas de oruga del lateral derecho (2 y 4, ver Fig.5.1) giran en sentido positivo. ↓: Ruedas de oruga lateral derecho giran en sentido negativo.
	Axis3 ↔	----------

Tabla 5. 5: Controles comunes

A modo de resumen de la Tabla 5.5 se puede decir que los controles comunes son los que llevan a cabo las siguientes operaciones:

- **Cambio de un *modo de control* a otro**. La combinación de botones destinada a tal fin va a ser la misma independientemente del modo de control que se esté ejecutando.

- **Movimientos de desplazamiento y giro del robot.** Tanto el joystick izquierdo como el derecho van a tener en todos los modos el mismo comportamiento asociado. Si los dos se mueven hacia arriba el robot avanza, si se mueven hacia abajo retrocede y si mueven cada uno en una dirección gira.
- **Movimiento de ayuda en el levantamiento sobre sus patas u orugas.** El botón B12, siempre que se mantenga pulsado, hará girar las ruedas de oruga delanteras en sentido contrario a las traseras.

5.3 Controles del robot Orugator.

Antes de comenzar con la descripción de la aplicación para el robot Orugator y las nociones básicas para su control, se va a realizar un breve repaso de la estructura de este para que quede claro que partes son las que se accionan cuando se pulsan los diferentes botones.

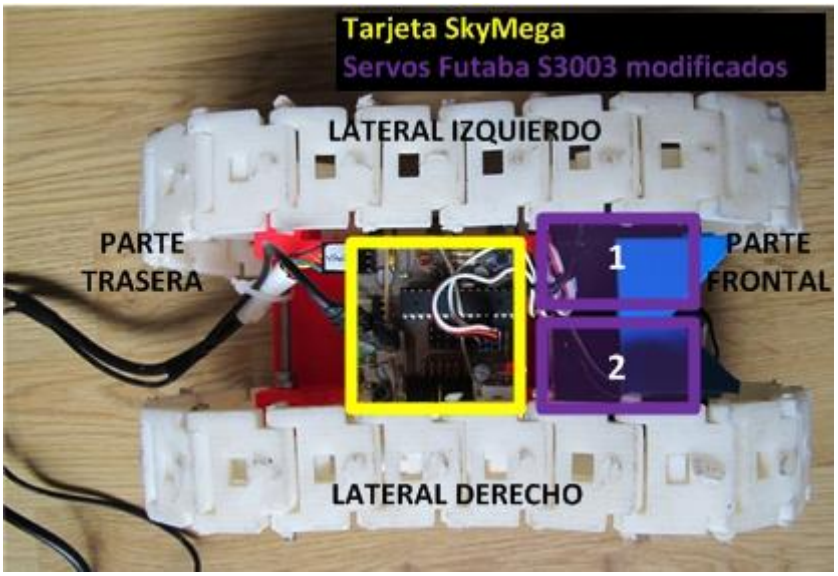


Figura 5. 6: Estructura robot Orugator

Tal y como se observa en la Figura 5.6, el robot orugator consta de dos ruedas de oruga a las que nos referiremos como la del lateral izquierdo y la del lateral derecho. Cada una de dichas ruedas es accionada por un servo modificado para giro continuo a los que nombraremos servo nº1 (lateral izquierdo) y nº2 (lateral derecho).

Este robot en comparación con el robot F-Track puede llevar a cabo un número de operaciones muy reducidas debido a que solo consta de dos partes móviles (oruga derecha e izquierda) por lo que todas sus operaciones se reducirán a movimiento de avance, de retroceso y giro y al control de la velocidad de dichos movimientos.

Por tanto, va a poder ser controlado haciendo únicamente uso de los joysticks del gamepad. Este robot no va a requerir que se programen varios modos de control ya que con uno se puede dar servicio a todas las posibilidades de control que presenta. Sin embargo se van a programar 3 modos de control, con el objetivo de evaluar que empleo de los joysticks es el más adecuado.

5.3.1 Modo de control 1.

En este modo de control, los movimientos que permiten avanzar, retroceder y girar se controlan igual que para el F-Track. Es decir, vienen gobernados por el joystick izquierdo y derecho, cada uno de los cuales es responsable del movimiento de las ruedas del lateral izquierdo y derecho respectivamente. Por lo que los movimientos que se podrán realizar son:

- Si los dos joystick se mueven conjuntamente hacia arriba o hacia abajo el robot se desplazara hacia delante o atrás respectivamente.
- Si el joystick izquierdo se mueve hacia arriba mientras que el derecho hacia abajo el robot gira hacia la derecha.
- Si el joystick izquierdo se mueve hacia abajo mientras que el derecho hacia arriba el robot gira hacia la izquierda.

Cabe destacar que como los joysticks son analógicos, van a permitir que se pueda controlar la velocidad de los movimientos del robot. Este avanzara o girara más deprisa cuanto más hacia arriba o abajo se empujen las palancas del *gamepad*.

Modo 1		
Botón	Acción	
B1	--- *No asignado* ---	
B2	--- *No asignado* ---	
B3	--- *No asignado* ---	
B4	--- *No asignado* ---	
B5	--- *No asignado* ---	
B6	--- *No asignado* ---	
B7	--- *No asignado* ---	
B8	--- *No asignado* ---	
B9	--- *No asignado* ---	
B10	--- *No asignado* ---	
B11	--- *No asignado* ---	
B12	--- *No asignado* ---	
Cruceta	Acción	
Axis5 ↕	--- *No asignado* ---	
Axis6 ↔	--- *No asignado* ---	
Joystick		Acción
Izquierdo	Axis2 ↕	↑: La rueda de oruga del lateral izquierdo (ver Figura5.4) gira en sentido positivo. ↓: Rueda de oruga lateral izquierdo gira en sentido negativo.
	Axis1 ↔	--- *No asignado* ---
Derecho	Axis4 ↕	↑: La rueda de oruga del lateral derecho (Fig.5.4) gira en sentido positivo. ↓: Rueda de oruga lateral derecho gira en sentido negativo.
	Axis3 ↔	--- *No asignado* ---

Tabla 5. 6: Orugator. Controles Modo 1

5.3.1 Modo de control 2.

El modo de control 2 permite gestionar todos los movimientos del robot con un solo joystick, el joystick izquierdo. Si este se mueve hacia arriba el robot avanzara, si se mueve hacia abajo se desplazara hacia atrás y si se mueve hacia la derecha o izquierda el robot girara respectivamente a derechas o izquierda.

Cabe destaca que la velocidad de desplazamiento así como la de giro será mayor cuanto la palanca del joystick diste más de su posición central de reposo.

Modo 2		
Botón	Acción	
B1	--- *No asignado* ---	
B2	--- *No asignado* ---	
B3	--- *No asignado* ---	
B4	--- *No asignado* ---	
B5	--- *No asignado* ---	
B6	--- *No asignado* ---	
B7	--- *No asignado* ---	
B8	--- *No asignado* ---	
B9	--- *No asignado* ---	
B10	--- *No asignado* ---	
B11	--- *No asignado* ---	
B12	--- *No asignado* ---	
Cruceta	Acción	
Axis5 ↕	--- *No asignado* ---	
Axis6 ↔	--- *No asignado* ---	
Joystick		Acción
Izquierdo	Axis2 ↕	↑: El robot se desplaza hacia adelante. ↓: El robot se desplaza hacia atrás.
	Axis1 ↔	→: El robot gira hacia la derecha. ←: El robot gira hacia la izquierda.
Derecho	Axis4 ↕	--- *No asignado* ---
	Axis3 ↔	--- *No asignado* ---

Tabla 5. 7: Orugator. Controles Modo 2

5.3.1 Modo de control 3.

Este modo de control 3 es muy parecido al modo de control 2 con la única diferencia de que mientras en el modo 2 un solo joystick se encargaba de gestionar los movimientos de desplazamiento y giro, en el modo 3 se han repartido estos movimientos entre las dos palancas. Es decir el joystick izquierdo se encarga de los movimientos de desplazamiento del robot hacia delante y atrás y de la velocidad con la que los realiza, mientras que el joystick derecho gestiona el movimiento de giro a derechas e izquierda y también su velocidad.

Modo 3		
Botón	Acción	
B1	--- *No asignado* ---	
B2	--- *No asignado* ---	
B3	--- *No asignado* ---	
B4	--- *No asignado* ---	
B5	--- *No asignado* ---	
B6	--- *No asignado* ---	
B7	--- *No asignado* ---	
B8	--- *No asignado* ---	
B9	--- *No asignado* ---	
B10	--- *No asignado* ---	
B11	--- *No asignado* ---	
B12	--- *No asignado* ---	
Cruceta	Acción	
Axis5 ↕	--- *No asignado* ---	
Axis6 ↔	--- *No asignado* ---	
Joystick		Acción
Izquierdo	Axis2 ↕	↑: El robot se desplaza hacia adelante. ↓: El robot se desplaza hacia atrás.
	Axis1 ↔	--- *No asignado* ---
Derecho	Axis4 ↕	--- *No asignado* ---
	Axis3 ↔	→: El robot gira hacia la derecha. ←: El robot gira hacia la izquierda.

Tabla 5. 8: Orugator. Controles Modo 3

Durante los ensayos con el robot Orugator se concluyo que la forma más rápida y precisa de controlarlo es la del modo de control 1.

Capítulo 6

Conclusiones y líneas futuras

6.1 Conclusiones

El proyecto realizado tenía como principal objetivo lograr que el robot móvil imprimible F-Track pudiese ser controlado remotamente mediante el empleo de un gamepad. Pues bien, podemos afirmar que dicho objetivo ha sido superado junto con todos los subojetivos u objetivos secundarios marcados.

Es decir, se ha conseguido un control total del robot mediante el manejo de un gamepad consiguiéndose una forma de control del robot sencilla e intuitiva apta para todo tipo de persona con o sin conocimientos acerca de la materia. Además en dicho proceso se han superado todos los subobjetivos, es decir:

- Se consiguió dotar al robot móvil de los medios necesarios para que pudiese llevar a cabo todos los movimientos para los que fue diseñado. Es decir mediante el empleo de los Servos Futaba S3003 (modificados y sin modificar) y la placa microcontroladora SkyMega 1.0 se logro gestionar y controlar de forma rápida y precisa todos los movimientos del robot.
- También, se consiguió con éxito equipar el robot móvil con los medios necesarios para que pudiese ser teleoperado. Lográndose una comunicación rápida y fluida entre el robot y el PC mediante el empleo del puerto serie de la placa SkyMega y el cable USB-Serie (FTDI TTL-232R-5V).
- Se diseño y desarrollo satisfactoriamente una aplicación capaz de controlar el robot móvil mediante el empleo de un *gamepad*. Obteniéndose como resultado una forma sencilla e intuitiva a través de la cual, cualquier usuario puede interaccionar con el robot.
- Además se logro con éxito que la aplicación objetivo de la tesis, pudiese ser reutilizada y permitiese fácilmente teleoperar otros robots móviles imprimibles basados en la misma arquitectura de control, sin la necesidad de diseñar una nueva aplicación específica para cada nueva máquina. Es decir, gracias al desarrollo del software mediante el empleo de módulos de programación RDK se consiguió controlar no únicamente el robot F-Track, sino también el robot Orugator de una forma precisa mediante el empleo de un gamepad.

Cabe destacar que con la realización de este proyecto no solo se han superado los objetivos marcados, sino que se han rebasado obteniéndose una herramienta muy potente para el estudio, desarrollo y evaluación de robots móviles imprimibles. Ya que la aplicación desarrollada a lo largo de esta tesis proporciona a futuros estudiantes o investigadores en este campo, una forma rápida y simple de familiarizarse con los movimientos del robot y evaluar su diseño.

6.2 Líneas Futuras

Pese a que se han logrado todos los objetivos marcados y la aplicación es válida y suficiente para operar con eficiencia un robot, queda la posibilidad de seguir mejorando todos y cada uno de los aspectos abordados en este proyecto.

Así pues, a la vista de los resultados y la experiencia adquirida a lo largo de esta tesis se proponen las siguientes líneas de trabajo y desarrollo para diseños y aplicaciones futuras:

Aspectos del diseño y la mecánica del robot F-Track.

El robot F-Track tal y como se ha podido apreciar durante las diferentes pruebas a las que ha sido sometido, posee un buen diseño que lo hace apto para desenvolverse en las más diversas superficies y le permite superar una gran variedad de obstáculos, llegando incluso a poder subir y bajar escaleras. Sin embargo, se han apreciado diversas posibilidades para la mejora de su diseño y su mecánica.

Se observo que la articulación o punto de unión entre las patas u orugas y el cuerpo central presentaba cierta holgura lo que repercutía en un pequeño error de posicionamiento de dichas patas respecto al robot. Además dicha holgura dificultaba ligeramente los movimientos de desplazamiento y giro del vehículo.

También se aprecio que algunas juntas tóricas empleadas para la transmisión patinaban lo que repercutía en que ciertas ruedas tuviesen mas agarre o tracción que otras. Para mejorar esto se propone para diseños futuros cambiarlas por correas dentadas y rediseñar las poleas y tambores empleados en la transmisión para adecuarse a estas.

En cuanto a los actuadores, el robot F-Track hace uso de ocho servomotores Futaba S3003, cuatro de ellos han sido modificados para giro continuo y se emplean para hacer mover las ruedas de cada oruga, mientras que los otros cuatro servos se emplean sin modificar para posicionar las orugas respecto al cuerpo central. Pues bien, se ha observado que dichos últimos cuatro servos a penas poseen el par necesario para posicionar las orugas lo que convierte ciertos movimientos cuyos responsables son ellos en movimientos lentos y carentes de fuerza. Por tanto sería recomendable sustituir estos 4 servos por otros más potentes, es decir, por un modelo que presente mayor par.

Aspectos electrónicos.

En cuanto a los aspectos electrónicos se puede destacar la falta de sensores de los robots. Esto es debido a que para la realización de este proyecto no se requerían, ya que en todo momento los movimientos del robot van a ser controlados a través de un gamepad gestionado por un operario. Pero para futuras evoluciones del robot sería recomendable dotarlo de diversos sensores: inercial, inclinación, proximidad, contacto… Ya que esto permitiría cierto grado de autonomía del robot pudiéndose llevar a cabo un control compartido entre el operario y la maquina.

También otra posible línea de desarrollo podría ser convertir la comunicación robot-PC en inalámbrica mediante el empleo de tecnología de radiofrecuencias o Bluetooth. Esto junto con dotar al robot de baterías eléctricas permitiría que el campo de acción de este se incrementase notablemente ya que en este momento está limitado a la longitud del cable que lo une con el PC y la fuente de alimentación.

Aspectos prueba y evaluación de la aplicación

Además, otro trabajo interesante a realizar en el futuro, sería realizar un estudio en el que se sometiese a varias personas a unas pruebas determinadas de teleoperación con el gamepad.

Estas pruebas nos permitirían evaluar la aplicación y conocer la opinión de los usuarios para su posterior mejora. Lo ideal sería estructurar las pruebas de forma que los usuarios teleoperasen en los diferentes modos y en diferentes entornos, para finalmente estudiar los tiempos empleados, el porcentaje de éxito, y las opiniones de los operadores.

Capítulo 7

Presupuesto

Este séptimo capítulo denominado "Presupuesto", recopila los costes incurridos a lo largo de esta tesis. Con este fin, este capítulo incluye una división en fases y subfases de desarrollo del proyecto, su correspondiente diagrama de Gantt y un desglose de costes de personal, costes del material y costes totales.

7.1 Planificación del proyecto

La ejecución del proyecto se ha llevado a cabo a lo largo de 10 meses en la División de Ingeniería de Sistemas y Automática de la Escuela Politécnica Superior de la Universidad Carlos III de Madrid. A continuación se describen brevemente las fases en las que se ha dividido el desarrollo del proyecto:

Investigación inicial: Previamente incluso a la planificación de este proyecto, se ha realizado un estudio sobre el estado actual de desarrollo de robots móviles y aplicaciones para el control de estos tales como el software modular OpenRDK.

Formación: Para llevar a cabo esta tesis ha sido necesario afianzar y ampliar los conocimientos, ya adquiridos a lo largo de los diferentes cursos de la titulación de Ingeniería Industrial, sobre lenguaje de programación C y sobre microcontroladores y su correcta programación.

También ha sido necesario iniciarse en programación orientada a objetos típica de C++ y adquirir unos conocimientos básicos sobre RDK.

Programación placa SkyMega1.0: En esta fase se procedió al estudio, desarrollo y evaluación de diferentes programas para el microcontrolador ATmega168/V de Atmel instalado en la placa SkyMega1.0 con el objetivo de familiarizarse y hacer lo más simple y eficiente el control de los servomotores y la comunicación a través del puerto serie.

Como resultado de esta fase se obtuvo el microcontrolador correctamente configurado para la recepción de datos a través del puerto serie, y en función de dichos valores realizar el control de los servomotores.

Selección periférico para el control del robot móvil: Se lleva a cabo la búsqueda del periférico más adecuado que permita un completo, intuitivo y fácil control del robot y de todos sus posibles movimientos.

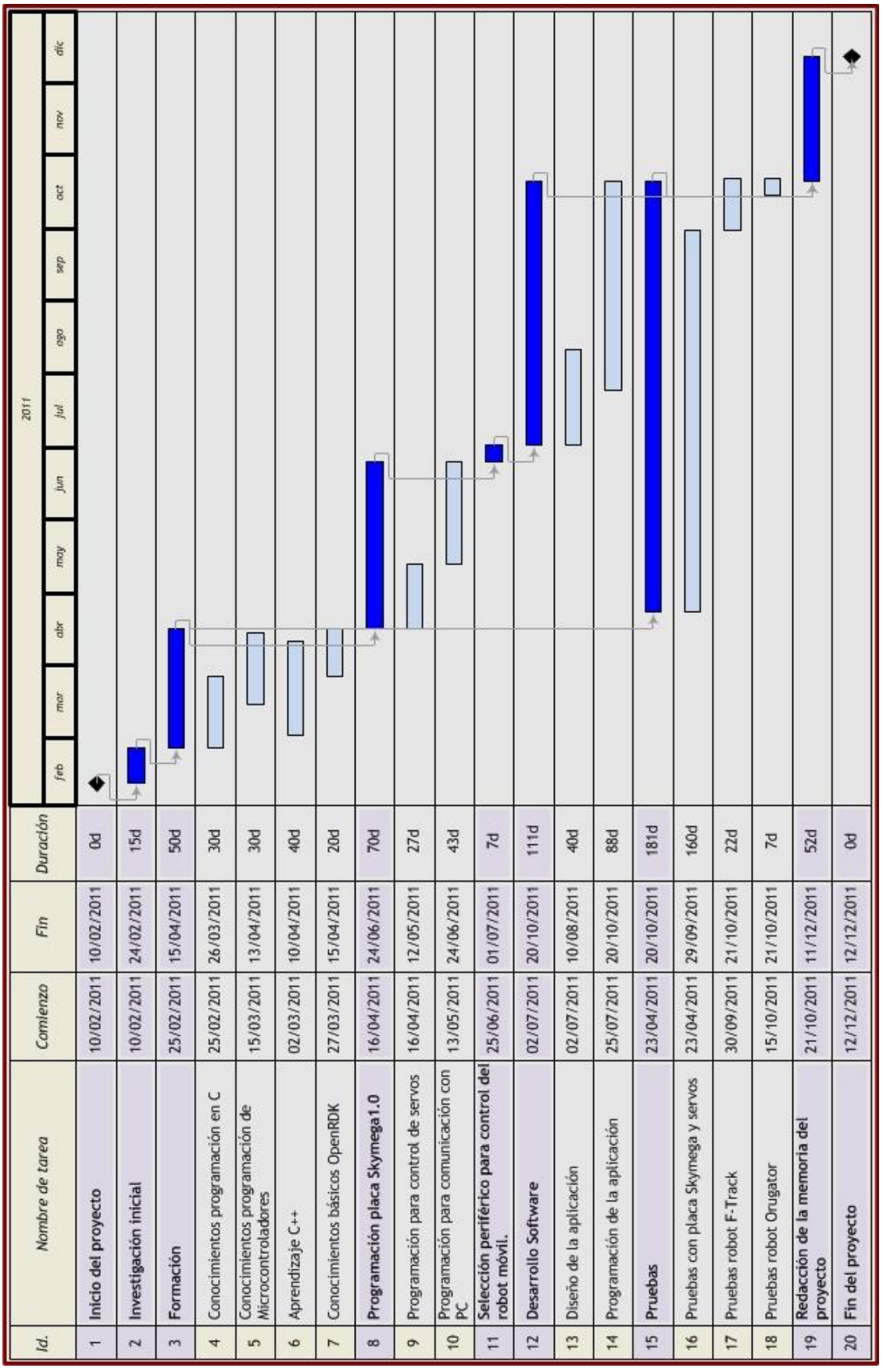
Esta fase termina con la elección de un *gamepad* o periférico para videojuegos compuesto por dos *joysticks* y 12 botones.

Desarrollo software: En esta fase se realiza un breve diseño previo de la aplicación para un posterior desarrollo del software empleando C++ y programación modular RDK.

Pruebas: Pruebas de funcionamiento con el robot F-Track y el robot Orugator en medios favorables y adversos (con y sin obstáculos), para la posterior corrección, depuración y mejora del software a partir de los resultados arrojados por los ensayos.

Redacción de la memoria del proyecto: Tras conseguir los hitos anteriores y tras haber realizado las pruebas y analizar los resultados. Se obtuvieron las conclusiones, se anotaron las posibles líneas futuras de desarrollo y se procedió a la redacción de la presente tesis.

A continuación se refleja mediante un diagrama de Gantt las principales actividades que se desarrollan en las etapas anteriormente mencionadas. Con esto se pueden observar las relaciones entre actividades.



Id.	Nombre de tarea	Comienzo	Fin	Duración
1	**Inicio del proyecto**	10/02/2011	10/02/2011	0d
2	**Investigación inicial**	10/02/2011	24/02/2011	15d
3	**Formación**	25/02/2011	15/04/2011	50d
4	Conocimientos programación en C	25/02/2011	26/03/2011	30d
5	Conocimientos programación de Microcontroladores	15/03/2011	13/04/2011	30d
6	Aprendizaje C++	02/03/2011	10/04/2011	40d
7	Conocimientos básicos OpenRDK	27/03/2011	15/04/2011	20d
8	**Programación placa Skymega1.0**	16/04/2011	24/06/2011	70d
9	Programación para control de servos	16/04/2011	12/05/2011	27d
10	Programación para comunicación con PC	13/05/2011	24/06/2011	43d
11	**Selección periférico para control del robot móvil.**	25/06/2011	01/07/2011	7d
12	**Desarrollo Software**	02/07/2011	20/10/2011	111d
13	Diseño de la aplicación	02/07/2011	10/08/2011	40d
14	Programación de la aplicación	25/07/2011	20/10/2011	88d
15	**Pruebas**	23/04/2011	20/10/2011	181d
16	Pruebas con placa Skymega y servos	23/04/2011	29/09/2011	160d
17	Pruebas robot F-Track	30/09/2011	21/10/2011	22d
18	Pruebas robot Orugator	15/10/2011	21/10/2011	7d
19	**Redacción de la memoria del proyecto**	21/10/2011	11/12/2011	52d
20	**Fin del proyecto**	12/12/2011	12/12/2011	0d

Tabla 7. 1: Diagrama de Gantt

7.2 Estructura de descomposición del proyecto

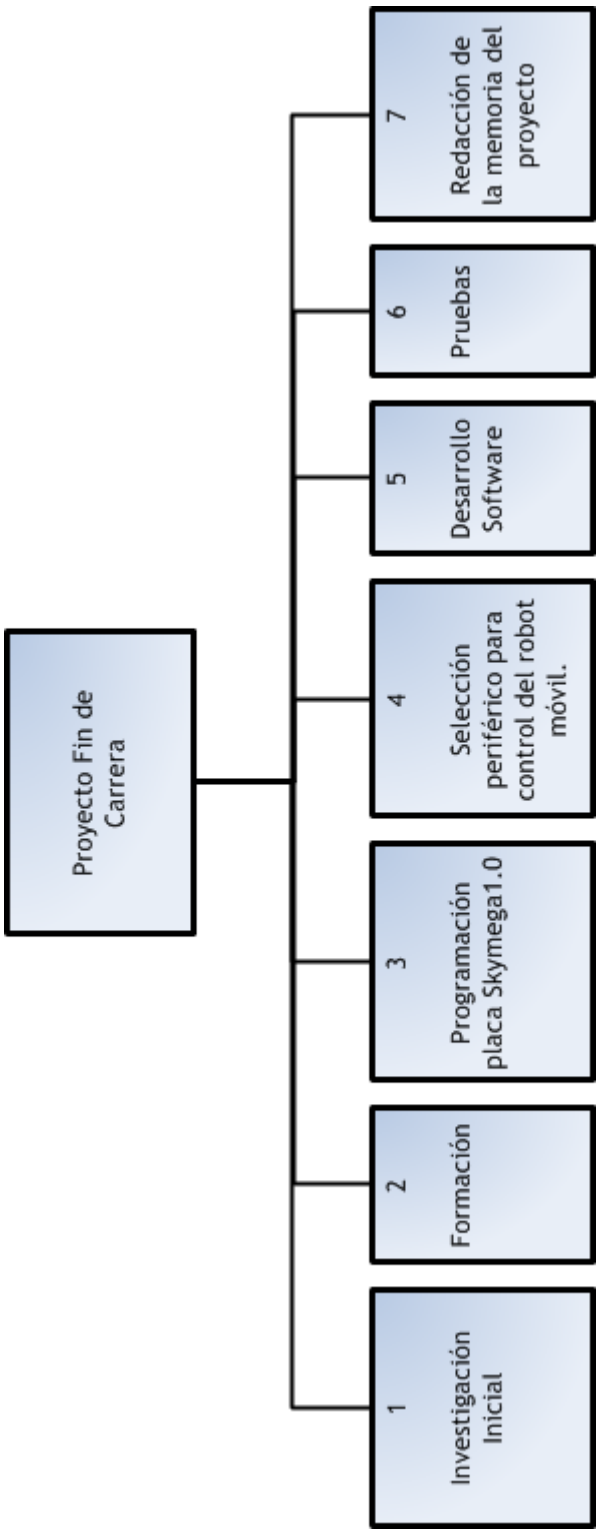


Figura 7. 1: Estructura de descomposición del proyecto

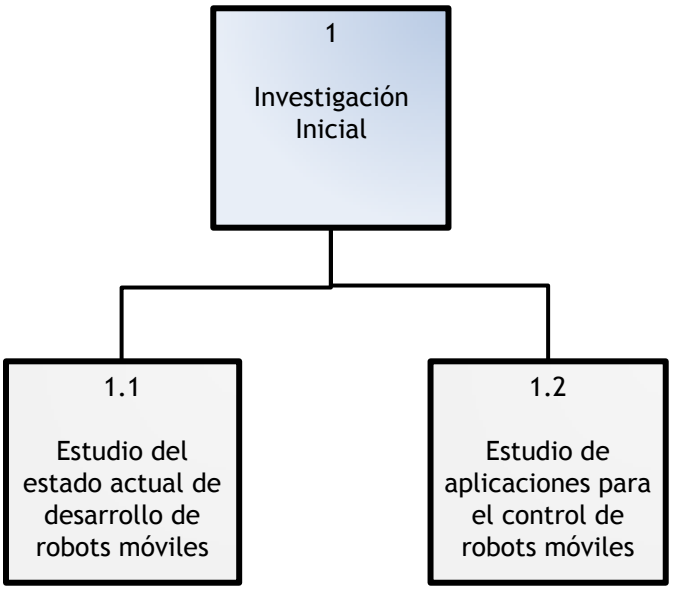


Figura 7. 2: Investigación inicial

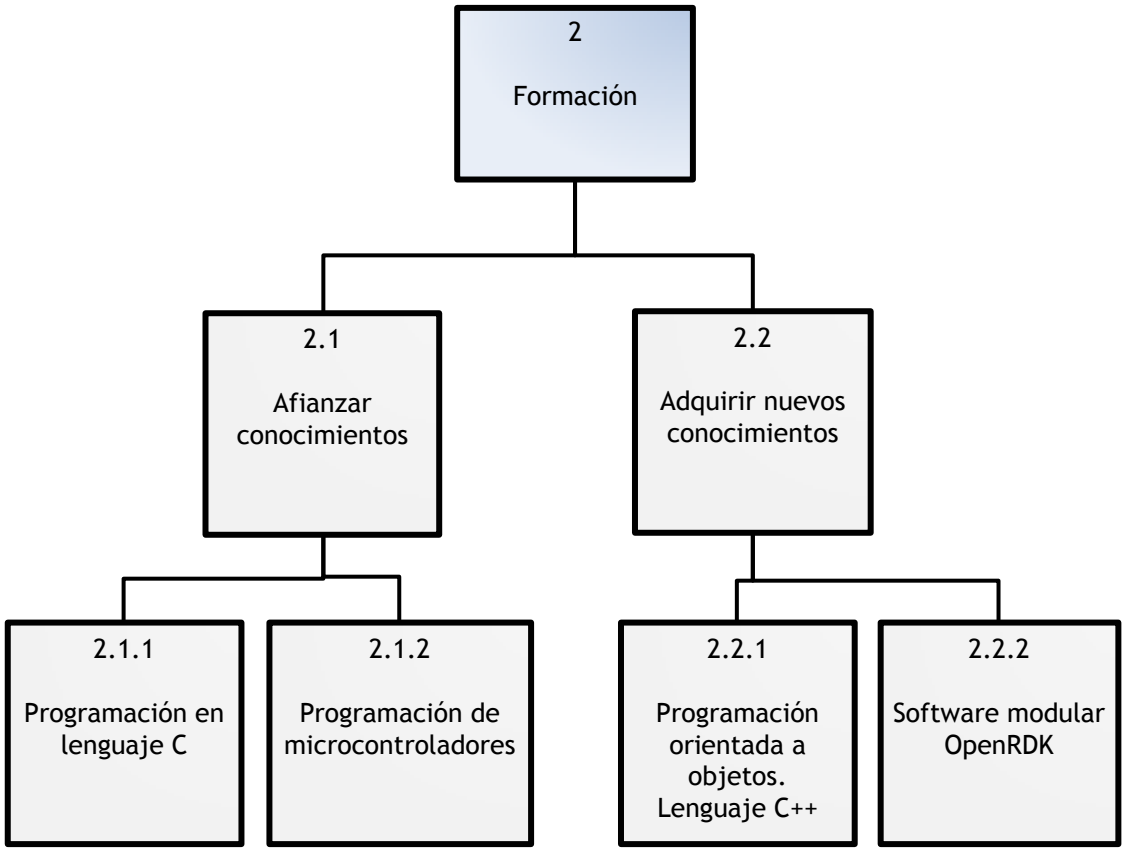


Figura 7. 3: Formación

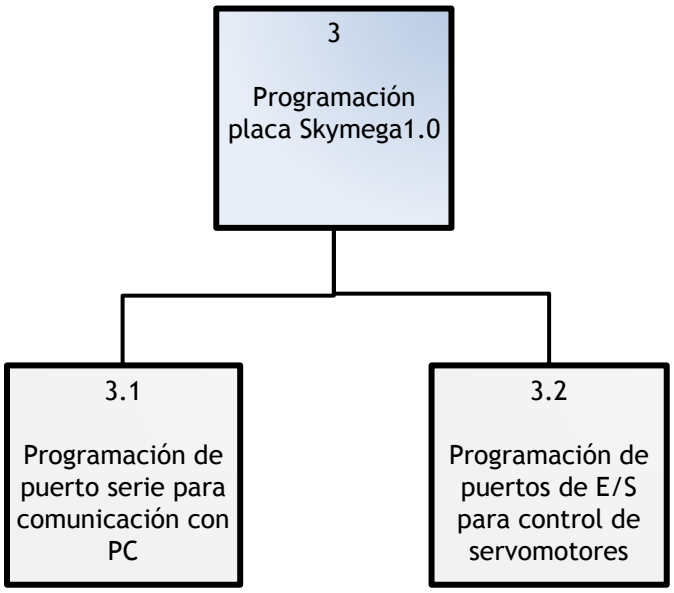


Figura 7. 4: Programación placa Skymega 1.0

Figura 7. 5: Selección periférico para control del robot móvil

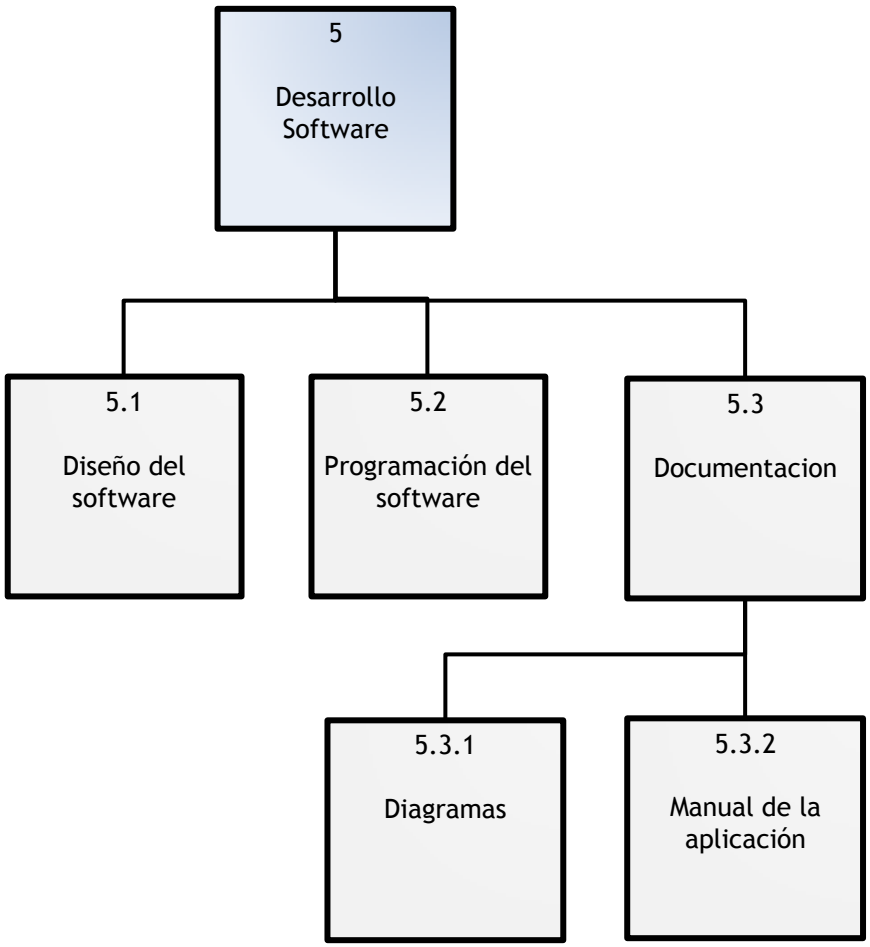


Figura 7. 6: Desarrollo Software

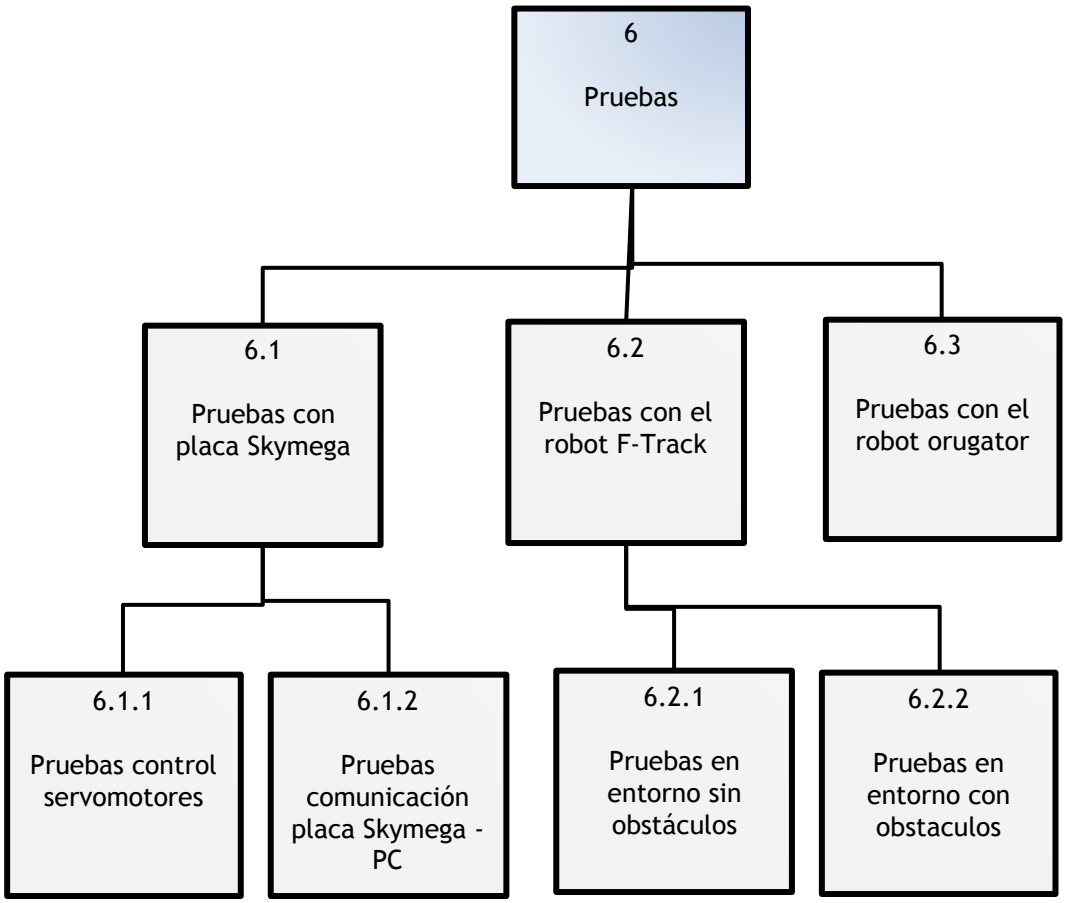


Figura 7. 7: Pruebas

7.3 Costes del material, costes del personal y costes totales

En la siguiente tabla quedan recogidos los medios empleados y sus precios unitarios, para luego poder realizar la estimación de costes de personal, costes materiales y costes totales del proyecto.

Medios empleados y precio unitario		
Descripción	***Unidades***	***Precio unitario***
Partida Software		
Sistema operativo Ubuntu	1	0 €
OpenRDK	1	0 €
Partida Hardware		
Robot imprimible F-Track	1	50 €
Robot imprimible Orugator	1	40 €
Placa Skymega1.0	2	10 €
Servomotores *'Futaba S3003'*	10	4 €
Gamepad *'Logitech Dual Action'*	1	20 €
PC	1	600 €
Partida de mano de obra		
Proyectante	500 h	25 €/h
Director	70 h	55 €/h

Tabla 7. 2: Medios empleados y precio unitario

Presupuesto	
Partida Software	0 €
Partida Hardware	724 €
Partida de mano de obra	16350 €
Suma total	**17074 €**
+ 6% Costes indirectos	1024.44 €
+ 13% Beneficio industrial	2219.62 €
+ 16% I.V.A.	2731.84 €
TOTAL	**23050 €**

Tabla 7. 3: Presupuesto total

El **presupuesto total** de este proyecto asciende a la cantidad de **23050 €.**

Veintitrés mil cincuenta euros.

Referencias

[1] Adrian Bowyer. The Self-replicating Rapid Prototyper, Manufacturing for the Masses. In *8th National Conference on Rapid Design, Prototyping & Manufacturing*, June 2007.

[2] Simon Bradshaw, Adrian Bowyer, and Patrick Haufe. The Intellectual Property Implications of Low-Cost 3D Printing. *SCRIPTed 5,* 2010.

[3] Stefano Carpin, Michael Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. Usarsim: a robot simulator for research and education. In *2007 IEEE International Conference on Robotics and Automation, ICRA 2007,* pages 1400-1405, 2007.

[4] Erik de Bruijn. On the viability of the open source development model for the design of physical objects. Lessons learned from the RepRap project, November 2010.

[5] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *11th International Conference on Advanced Robotics (ICAR 2003), Portugal*, pages 317-323, June 2003.

[6] Rachel Goldman, Amy Eguchi, and Elizabeth Sklar. Using educational robotics to engage inner-city students with technology. In *Proceedings of the 6th international conference on Learning sciences,* ICLS '04, pages 214-221. International Society of the Learning Sciences, 2004.

[7] Proyecto Source Forge, http://openrdk.sourceforge.net, Accedido en Diciembre 2011

[8] Gerhard Fischer. Distances and diversity: Sources for social creativity. In *Proceedings of Creativity and Cognition*, pages 128–136. ACM Press, 2005.

[9] Rhys Jones, Patrick Haufe, Edward Sells, Pejman Iravani, Vik Olliver, Chris Palmer, and Adrian Bowyer. RepRap-the replicating rapid prototyper. *Robotica,* 29(01):177- 191, January 2011.

[10] Rishab Aiyer Ghosh. Understanding Free Software Developers: Findings from the FLOSS Study. In Joseph Feller, Brian Fitzgerald, Scott A Hissam, and Karim R Lakhani, editors, *Perspectives on Free and Open Source Software, pages* 23–46. MIT Press, Boston/Mass., 2005.

[11] Joshua Noble. Programming Interactivity: A Designer's Guide to Processing, Arduino, and Openframeworks. O'Reilly Media, 1 edition, July 2009.

[12] D.H. Pink. Drive: The Surprising Truth about What Motivates Us. Penguin Group USA, 2011.

[13] K.S. Rawat and G.H. Massiha. A hands-on laboratory based approach to undergraduate robotics education. In *Robotics and Automation, 200/t. Proceedings. ICRA '0/t. 2004 IEEE International Conference on,* volume 2, pages 1370 - 1374 Vol.2, 2004.

[14] http://wiki.makerbot.com/thingomatic , Accedido en Noviembre 2011

[15] Igor M. Verner, Shlomo Waks, and Eli Kolberg. Educational robotics: An insight into systems engineering. *European Journal of Engineering Education*, 24(2):201, 1999.

[16] J. Gonzalez-Gomez, A. Valero-Gomez, A. Prieto-Moreno, M. Abderrahim. New Open Source 3D-printable Mobile Robotic Platform for Education. *6th International Symposium on Autonomous Minirobots for Research and Edutainment. May 2011.*

[17] Juan Gonzalez-Gomez, Alberto Valero-Gomez, Mario Almagro. Boosting Mechanical Design with the C++ OOML and Open Source 3D Printers. *IEEE Educon 2012*.

[18]Alberto Valero-Gomez, Juan Gonzalez-Gomez, Victor Gonzalez-Pacheco and Miguel Angel Salichs. Printable Creativity in Plastic Valley UC3M. *IEEE Educon 2012*.

[19] http://www.thingiverse.com/thing:7640 , Accedido en Diciembre de 2011

[20] http://proton.ucting.udg.mx/materias/robotica/r166/r68/r68.htm , Accedido en Diciembre de 2011

[21] Victoria E. de las Heras: 'DISEÑO Y CONSTRUCCIÓN DE UN MICROROBOT EUROBOT 2009', *PFC Universidad Carlos III de Madrid, 2009.*

[22] http://www.iearobotics.com/wiki/index.php?title=Servos_Futaba_3003 , Accedido en Diciembre de 2011

[23] http://www.iearobotics.com/wiki/index.php?title=SkyMega , Accedido en Diciembre de 2011

[24] Juan Gonzalez. Robots libres e imprimibles. *Convención de Open Source Hardware, Electronica y Robotica. Septiembre 2011.*

[25] Robot imprimible Orugator http://www.thingiverse.com/thing:8559 , Accedido en Diciembre 2011

[26] Robot imprimible F-Track http://www.thingiverse.com/thing:13298 , Accedido en Diciembre 2011

[27] Juan Ignacio Forcén Carvalho: ‘SISTEMA DE CONTROL REMOTO DE ROBOTS MÓVILES BASADO EN PDA’, *PFC Universidad Politecnica de Madrid, 2009.*

ANEXOS

ANEXO I- Código programación del microcontrolador.

ANEXO II- Robot F-Track. Secuencia de acciones recomendadas ante obstáculos.

ANEXO I- Código programación del microcontrolador.

```
// Control de 8 servos a través del puerto UART mediante el empleo de un vector de
// caracteres de tamaño 9, donde el primer carácter sirve para la sincronización "PC-Microcontrolador",
// los siguientes 4 caracteres son la velocidad en "ticks" de cada uno de los 4 primeros servos, y los últimos
// 4 caracteres es la posición a la que deben ir cada uno de los 4 últimos servos.
//
// Para compilar y cargar en el microcontrolador usar:
//          $ avr-gcc ftrackserver.c -mmcu=atmega168 -Wall -o main.o
//          $ avr-objcopy -O ihex main.o main.hex
//          $ avrdude -q -patmega168 -cstk500v1 -P/dev/ttyUSB0 -b19200 -D -Uflash:w:main.hex
//

#include <avr/interrupt.h>
#include <avr/io.h>
#include <stdint.h>
#include <stdio.h>

//--- Definiciones ---

#define F_CPU 16000000
#define UART_BAUD 9600

#define DELAY_0_3ms 19
#define DELAY_0_2ms 12
#define DELAY_1ms   62
#define DELAY_2ms   125
#define DELAY_2_5ms 156

#define CENTRO DELAY_1ms

//--- Declaración de funciones ---

int uart_getchar(FILE *stream);
unsigned char deg2time(char);
void servo_pos(unsigned char , char);

//--- Variables globales ---

// Estados de la señal PWM usada para el control de los servos
//  - State0: Ancho de pulso 0.3ms (Señal PWM a 1)
//  - State1: Posición del servo. Ancho de pulso "pos". Ancho entre 0 y 2ms (señal PWM a 1)
//  - State2: Complementaria de la posición del servo. Width = 2ms - State1_Width (señal PWM a 0)
//  - State3: Ancho de pulso 0.2ms (señal PWM a 0)
enum { STATE0, STATE1, STATE2, STATE3 } state;

unsigned char servo=0;  // Indice del servo (de 0 a 7)
```

```
// Posición del servo en "ticks" (unidades de tiempo)
volatile unsigned char
pos[]={CENTRO,CENTRO,CENTRO,CENTRO,CENTRO,CENTRO,CENTRO,CENTRO};

// Flag de comienzo de nuevo ciclo PWM (20ms)
volatile unsigned char pwm_cycle_start=0;

// Número del puerto al que está conectado el servo (PORTB(0-3) o PORTC(0-3))
unsigned char n_port=0; // Varía entre 0 y 3.

// Usar la UART como un stream de salida
FILE uart_str = FDEV_SETUP_STREAM(NULL, uart_getchar, _FDEV_SETUP_READ);

//--- Rutina de atención a la interrupción (ISR) ---
// Se ejecuta cada vez que se dispara el comparador A.
// Es la encargada de generar la señal PWM que controla la posición de los servos.
ISR(TIMER0_COMPA_vect)
{
    // El "flag" de la interrupción es limpiado automáticamente.
    // Implementación de una maquina de estados finitos.
    switch(state) {
        case STATE0: //-- Señal PWM a 1. Ancho fijo de 0.3ms.
            if (servo%2){
                PORTB = (1<<n_port);
                n_port++;}
            else {
                PORTC = (1<<n_port);
            }
            OCR0A=DELAY_0_3ms;
            state = STATE1;  // Siguiente estado.
            break;

        case STATE1: //-- Señal PWM a 1. Ancho variable de valor pos[servo].
            OCR0A=pos[servo];
            state = STATE2;  // Siguiente estado.
            break;

        case STATE2: //-- Señal PWM a 0. Ancho variable de valor "DELAY_2ms - pos[servo]".
            PORTB=0;
            PORTC=0;
            OCR0A=DELAY_2ms - pos[servo];

            state = STATE3;  // Siguiente estado.
            break;

        case STATE3: //-- Señal PWM a 0. Ancho fijo de 0.2ms.
            OCR0A=DELAY_0_2ms;
            servo = (servo+1)%8;
            state = STATE0;  // Siguiente estado.

            if (servo==0){ // Si se cumple un nuevo ciclo PWM comienza.
                n_port=0;
```

```
                pwm_cycle_start=1;
            }
            break;
    }
}


//--- Inicializaciones del microcontrolador ---

void ports_init()
{
    //-- Configuración de los puertos.
    // Configura los cuatro primeros pines del puerto B y C como salida
    DDRB= 0x2F; // 0b00101111
    DDRC= 0x0F; // 0b00001111
}

void timer0_init()
{
    //-- Configuración timer0.
    // Configuración timer en modo CTC (Clear Timer on Compare mode) (WGM02:0 = 0b010)
    TCCR0A = _BV(WGM01); //_BV(i) Devuelve número binario 8bits donde el bit "i" es 1 y el resto
0.

    // Prescaler 256 (CS02:0 = 0b100). Reloj = 16Mhz (T = 1/16MHz = 62.5ns)
    // Ticks con prescaler: 62.5ns/256 = 16us
    TCCR0B = _BV(CS02);
    // Delay of 0.3ms --> OCR0A = 19 (18.750)
    // Delay of 0.2ms --> OCR0A = 12 (12.5)
    // Delay of 1ms  --> OCR0A = 62   (62.5)
    // Delay of 2ms -->  OCR0A = 125 (125)
    // Rango de la posición: 0 - 125 unidades de tiempo ("ticks") --> [-90, 90] grados
}

void interrupts_init()
{
    //-- Configuración de interrupciones.
    TIMSK0 = _BV(OCIE0A); // Habilitar interrupción del Comparador A (timer0)
    sei(); // Habilitar interrupciones
}

void uart_init()
{
    //-- Configuración de la UART (9600 bauds)
    UBRR0L = (F_CPU / (16UL * UART_BAUD)) - 1;
    UCSR0B = _BV(RXEN0); // Habilitar recepción (rx enable)
}



//--- FUNCION PRINCIPAL ---
int main(void)
{
    char c[8];
```

```
    int i;

    ports_init();          // Configurar puertos de Entrada/Salida
    timer0_init();         // Configurar timer0
    interrupts_init();     // Configurar interrupciones
    uart_init();           // Configurar UART

    stdin = &uart_str;   // Se define la UART como el stream de salida por defecto

    while(1) {
        //-- Asignación de la velocidad de los servos a través del puerto serie.
        // Si recibe el valor 181 (Valor de sincronización) por el puerto serie, se interpretan
        // los siguientes ocho caracteres como las velocidades en "ticks" y posiciones en grados
        // de los servos. En caso contrario, no se lee el puerto serie.
        if (getchar()==181){ // Sirve para sincronizar.
            for(i=0;i<4;i++){
                c[i]=getchar();
                if (c[i]>72){
                    pos[i]=125;}
                else if (c[i]<52){
                    pos[i]=0;}
                else{
                    pos[i]=c[i];}
            }
            for(i=4;i<8;i++){
                c[i]=getchar();
                if (c[i]>180){
                    servo_pos(i,0);}
                else{
                    servo_pos(i,c[i]-90);}
            }
        PORTB=PORTB^0x20; // El led parpadea si se están recibiendo datos a través del puerto
serie.
        }
        //fflush(stdin);
    }
    return 0;
}

//--- Definicion de funciones ---

int uart_getchar(FILE *stream)
{
    uint8_t c;

    loop_until_bit_is_set(UCSR0A, RXC0);
    if (UCSR0A & _BV(FE0)){ return _FDEV_EOF; }
    if (UCSR0A & _BV(DOR0)){ return _FDEV_ERR; }

    c = UDR0;
    return c;
}
```

```
// Conversión entre grados (deg) y "ticks" (time)
//-- La señal PWM que controla la posición del servo es calculada en "ticks"
//-- deg: ángulo en grados en el rango [-90,90]
//-- La función devuelve el ancho de la señal PWM en "ticks" (rango entre 0 y 125)
unsigned char deg2time(char deg)
{
    return 25*(90-deg)/36;
}
void servo_pos(unsigned char s, char deg)
{
    pos[s]=deg2time(deg);
}
```

ANEXO II- Robot F-Track. Secuencia de acciones recomendadas ante obstáculos.

A) Obstáculo cilíndrico.

Ante pequeños obstáculos puntuales tales como se muestran en la Figura A.1 se recomienda emplear únicamente el modo de control 1 (Ver *Capítulo 5* apartado *5.2.1 Modo de control 1*).

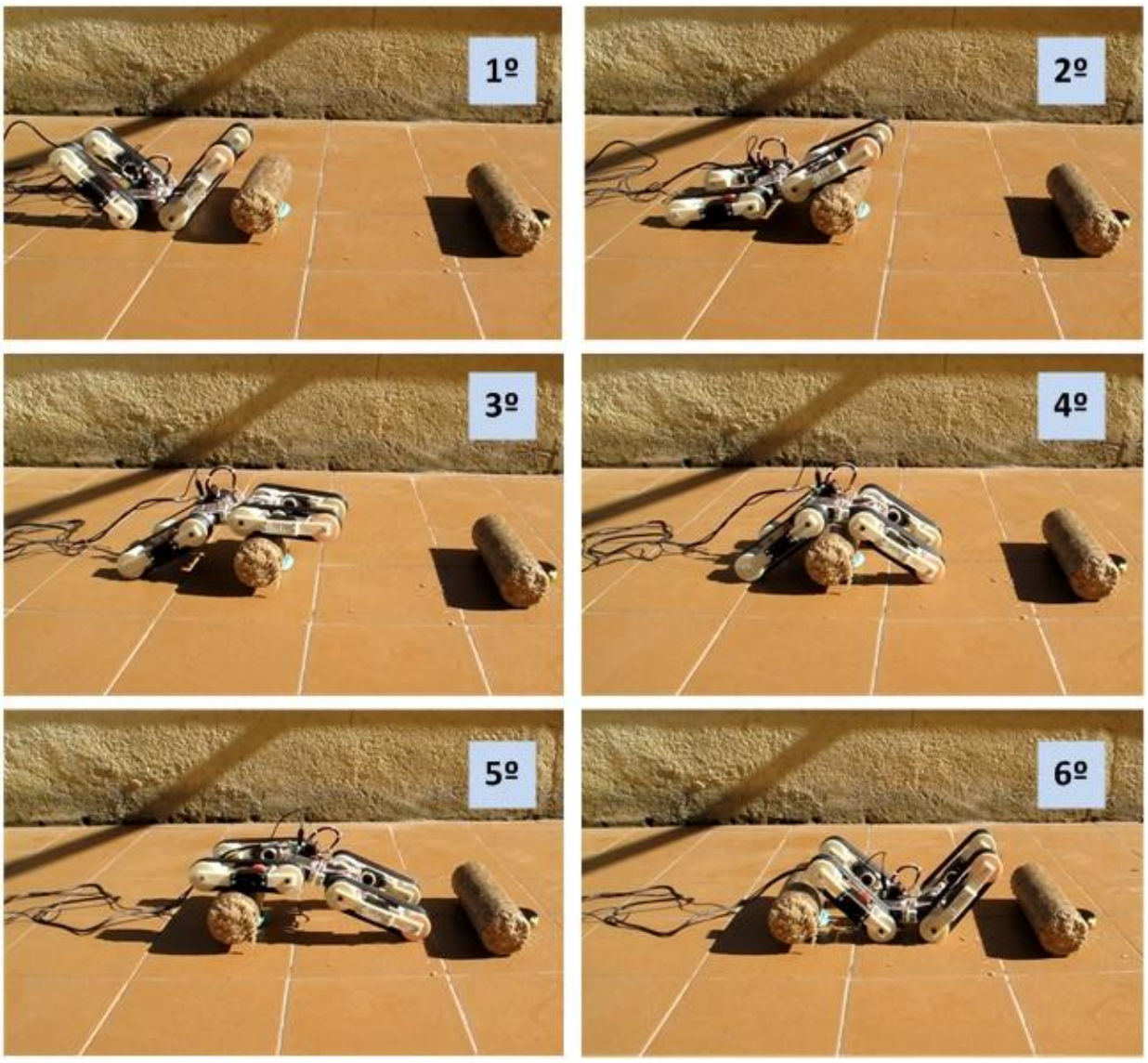


Figura A. 1: Robot F-Track ante obstáculo cilíndrico

1º Paso:

→ Seleccionar Modo de control 1. Pulsar B9 + B1 (Ver Fig. 5.3).

→ Situar el robot frente al obstáculo. Mover joysticks izquierdo y derecho hasta lograrlo.

→ Posicionar las patas u orugas del robot a 40º positivos. Pulsar B2.

2º y 3º Paso:

→ Desplazar robot hacia adelante. Mantener joystick derecho e izquierdo conjuntamente hacia arriba.

→ Mientras el robot avanza y empieza a superar el obstáculo, disminuir ángulo posicionamiento de las patas u orugas. Pulsar B1, B10, B5 y B6 de manera individual y en dicho orden según vaya siendo necesario.

4º y 5º Paso:

→ Seguir moviendo robot hacia adelante. Mantener joystick derecho e izquierdo conjuntamente hacia arriba.

→ Mientras el robot avanza, empezar a incrementar ángulo posicionamiento de las patas u orugas. Pulsar B5, B10 y B1 de manera individual y en dicho orden según vaya siendo necesario.

6º Paso:

→ El robot ha rebasado el obstáculo. Posicionar orugas a 40º positivos (Pulsar B2) para encarar nuevos obstáculos.

B) Escalera.

El robot F-Track podrá realizar la tarea de subir escaleras siempre y cuando los escalones no superen los 16.5 cm de altura. Para llevar a cabo tal trabajo se harán uso de los modos de control 1, 2 y 3 (Ver *Capítulo 5* apartado *5.2 Modos de control*).

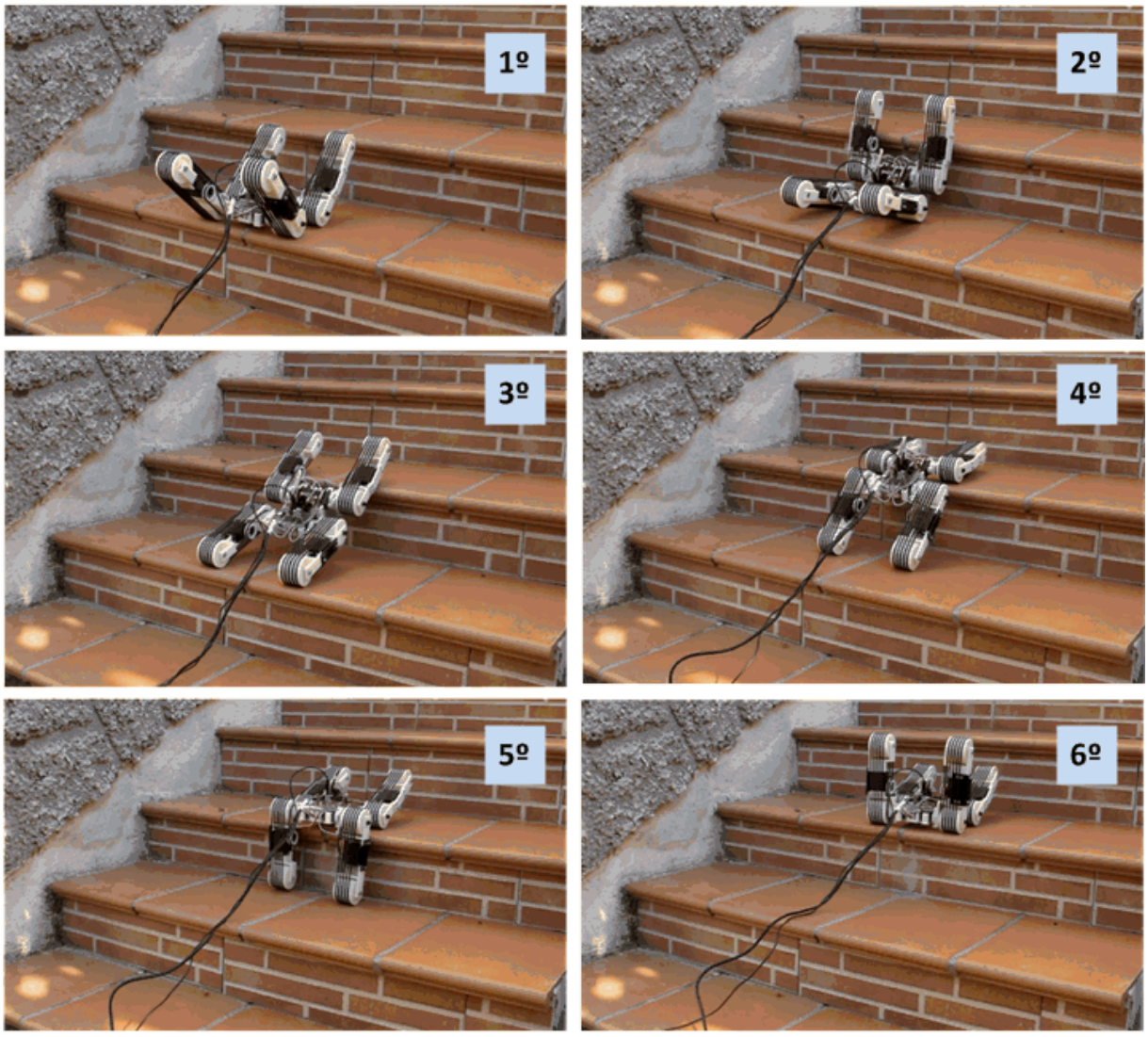


Figura A. 2: Robot F-Track ante escaleras

1º Paso:

→ Seleccionar Modo de control 1. Pulsar B9 + B1 (Ver Fig. 5.3).

→ Posicionar las patas u orugas del robot a 40º positivos. Pulsar B2.

→ Situar el robot frente al escalón. Mover joysticks izquierdo y derecho hasta lograrlo.

2º, 3º y 4º Paso:

→ Desplazar robot hacia adelante. Mantener joystick derecho e izquierdo conjuntamente hacia arriba.

→ Mientras el robot avanza y empieza a superar el escalón, disminuir ángulo posicionamiento de las patas u orugas. Pulsar B1, B10 y B5 de manera individual y en dicho orden según vaya siendo necesario.

5º Paso:

→ Cambiar a modo de control 2. Pulsar B9 + B2.

→ Posicionar patas u orugas delanteras a 20º positivos. Pulsar B1.

→ Cambiar a modo de control 3. Pulsar B9 + B3.

→ Posicionar patas u orugas traseras a 80º negativos. Pulsar B8.

6º Paso:

→ Desplazar robot hacia adelante. Mantener joystick derecho e izquierdo conjuntamente hacia arriba.

→ Mientras el robot intenta avanzar (joysticks se mantienen pulsados), posicionar orugas traseras a 80º positivos. Pulsar B4.

→ El robot ha superado un escalón. Volver al 1º Paso y realizar de nuevo las acciones para subir otro escalón.

C) Escalón.

Para subir y bajar bordillos o desniveles del terreno se optara por usar la secuencia descrita en el apartado *A) Obstáculo cilíndrico*, o la del apartado *B) Escaleras*. Dicha elección dependerá de la altura del bordillo eligiéndose la primera para pequeños desniveles.

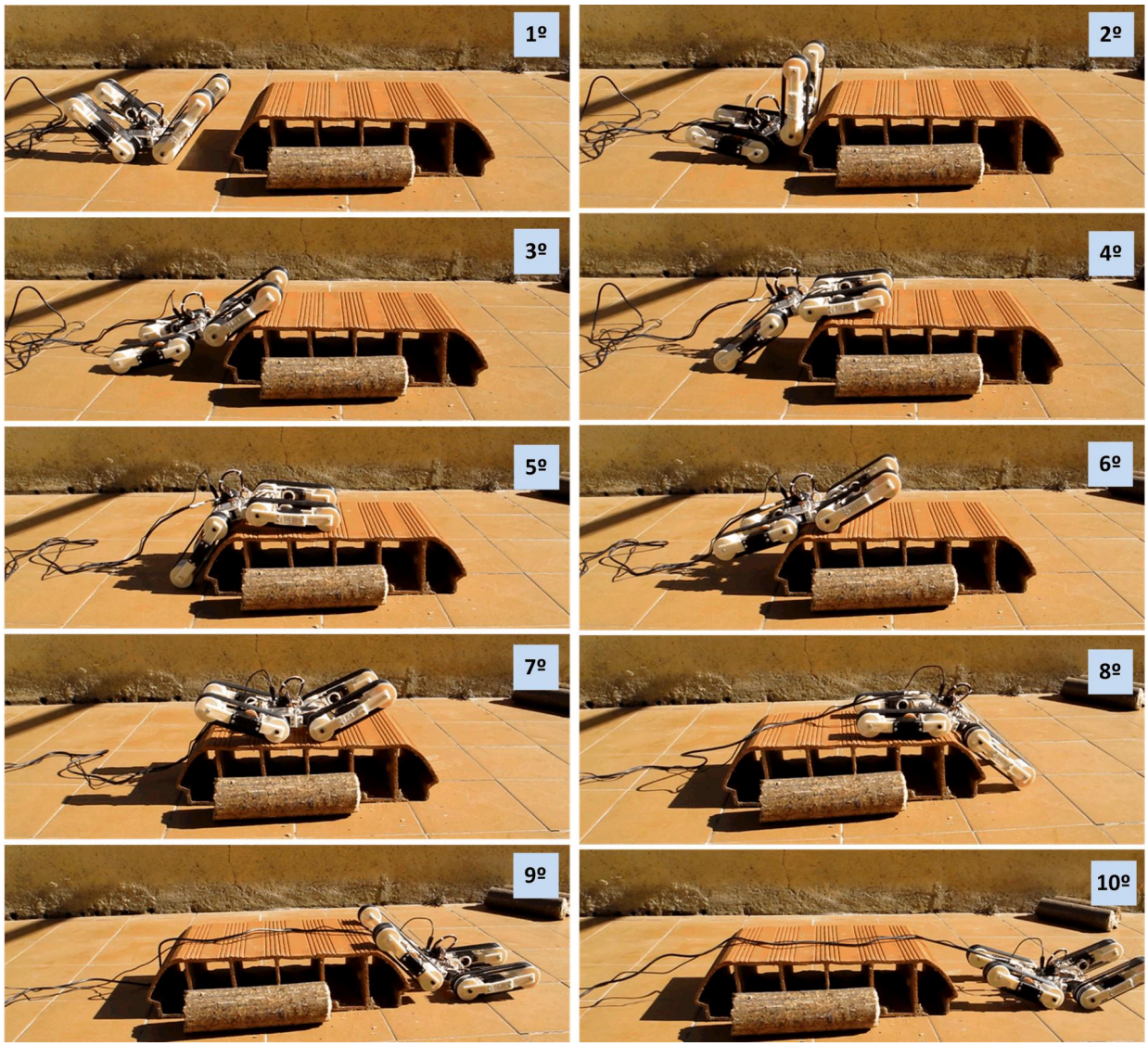


Figura A. 3: Robot F-Track ante escalón

1º Paso:

→ Seleccionar Modo de control 1. Pulsar B9 + B1 (Ver Fig. 5.3).

→ Situar el robot frente al obstáculo. Mover joysticks izquierdo y derecho hasta lograrlo.

→ Posicionar las patas u orugas del robot a 40º positivos. Pulsar B2.

2º, 3º, 4º y 5º Paso:

→ Desplazar robot hacia adelante. Mantener joystick derecho e izquierdo conjuntamente hacia arriba.

→ Mientras el robot avanza y empieza a superar el obstáculo, disminuir ángulo posicionamiento de las patas u orugas. Pulsar B1, B10 y B5 de manera individual y en dicho orden según vaya siendo necesario.

6º Paso:

→ Seguir moviendo robot hacia adelante. Mantener joystick derecho e izquierdo conjuntamente hacia arriba.

→ Mientras el robot avanza, empezar a incrementar ángulo posicionamiento de las patas u orugas. Pulsar B5, B10 y B1 de manera individual y en dicho orden según vaya siendo necesario.

7º Paso:

→ Desplazar robot hacia adelante hasta llegar al borde del desnivel. Mantener joystick derecho e izquierdo conjuntamente hacia arriba.

→ Posicionar orugas a 20º negativos (Pulsar B5) para bajar escalón.

8º, 9º y 10º Paso:

→ Desplazar robot hacia adelante hasta bajar desnivel. Mantener joystick derecho e izquierdo conjuntamente hacia arriba.

→ El robot ha rebasado el obstáculo. Posicionar orugas a 40º positivos (Pulsar B2) para encarar nuevos obstáculos.