



Universidad
Carlos III de Madrid

DEPARTAMENTO DE INGENIERÍA...

TRABAJO FIN DE GRADO

TÍTULO DEL TRABAJO

Autor: nombre del alumno

Director: nombre del director

Tutor: nombre del tutor

Ciudad, Mes año

Copyright ©año. Nombre del alumno

Esta obra está licenciada bajo la licencia Creative Commons
Atribución-NoComercial-SinDerivadas 3.0 Unported (CC BY-NC-ND 3.0).
Para ver una copia de esta licencia, visite
<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es> o envíe una carta a
Creative Commons, 444 Castro Street, Suite 900, Mountain View, California,
94041, EE.UU.
Todas las opiniones aquí expresadas son del autor, y no reflejan
necesariamente las opiniones de la Universidad Carlos III de Madrid.

Título: título del trabajo
Autor: nombre del alumno
Director: nombre del director
Tutor: nombre del tutor

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día de en, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO

PRESIDENTE

Agradecimientos

Agradezco a

Resumen

Este proyecto de resume en.....

Palabras clave: palabraclave1, palabraclave2, palabraclave3.

Abstract

In this project...

Keywords: keyword1, keyword2, keyword3.

Índice general

Agradecimientos	v
Resumen	vii
Abstract	ix
1. Introducción	1
1.1. Introducción a la robótica Open-Source TODO	1
1.2. Asociación de Robótica de la Universidad Carlos III .	2
1.3. Descripción del proyecto TODO	3
1.4. Estructura del documento TODO	3
2. Descripción de la línea de investigación	5
2.1. Línea de investigación de robots mini-humanoides . .	5
2.2. Campeonato CEABOT	6
3. Estado del arte	11
4. Objetivos TODO	13
4.1. Implementación de un controlador basado en visión por computador	13
4.2. Estudio de sensores y actuadores	14
4.3. Diseño electrónico	14
4.4. Diseño mecánico	15
4.5. Programación	15
5. Elección de componentes	19
5.1. Plataforma robótica	19
5.1.1. Selección de una plataforma robótica	19
5.2. Modificaciones básicas sobre el robot Bioloid	25
5.2.1. Alimentación	25
5.2.2. Cabeza móvil	25
5.3. Elección del controlador	25

5.3.1. Controlador de locomoción	25
5.3.2. Controlador de visión	25
5.4. Sensorización	25
5.4.1. Sensores de distancia	25
5.4.2. Sensor inercial	25
5.4.3. Cámara	25
5.4.4. Configuración final	25
6. Descripción de las herramientas a utilizar	27
6.1. Herramientas de diseño y fabricación de piezas	27
6.1.1. OpenSCAD	27
6.1.2. Impresión 3d	28
6.2. Herramientas de diseño de circuitos	29
6.2.1. KiCad	29
6.3. Herramientas de programación	30
6.3.1. OpenCV	30
6.3.2. Qt Creator	31
6.3.3. CMake	32
6.3.4. CM9 IDE	32
6.3.5. Git	33
7. Desarrollo	35
7.1. Diseño de las partes mecánicas	35
7.1.1. Cabeza	35
7.1.2. Tronco	35
7.1.3. Brazos	35
7.1.4. Piernas	35
7.2. Montaje del controlador	35
7.2.1. Esquema de montaje	35
7.2.2. Sistema de alimentación	35
7.2.3. Adecuación de sensores	35
7.2.4. Desarrollo de una placa de expansión	35
7.2.5. Puesta en marcha del controlador	35
8. Programación	37
8.1. Sistema de locomoción	37
8.1.1. Movimiento del servo PWM	37
8.1.2. Movimiento de los actuadores Dynamixel	37
8.1.3. Movimiento sincronizado de las articulaciones	39
8.1.4. Funciones de movimientos combinados	42
8.1.5. Creación de movimientos completos	42
8.1.6. Controlador de movimiento	42
8.2. Algoritmos de visión	42

<i>ÍNDICE GENERAL</i>	XIII
9. Evaluación de resultados	43
9.1. Pruebas de funcionamiento	43
9.2. Conclusión	43
9.3. Situación y desarrollos futuros	43
Bibliografía	45

Índice de figuras

1.1.	Logotipo de la UC3M ©UC3M	1
2.1.	Robots Nao y DARwIn-OP	5
2.2.	Prueba de la carrera de obstáculos	6
2.3.	Prueba de la escalera	7
2.4.	Prueba de sumo	8
2.5.	Marcador de la prueba de visión	9
5.1.	Hitec Robonova	20
5.2.	Kondo KHR-3HV	21
5.3.	Vstone Robovie-X	22
5.4.	Bioloid Premium	23
6.1.	Pantalla del editor de OpenSCAD	28
6.2.	Impresora 3D Prusa i2 Air	29
6.3.	KiCad	30
6.4.	Qt Creator	31
6.5.	CM9 IDE	32
8.1.	Amplitud de giro de un AX-12A	39

Capítulo 1

Introducción

En este capítulo...

Este tema.... Esto es un ejemplo de cita de un artículo [1].

- **ejemplo de lista de puntos.** Ejemplo.
- **ejemplo2 de lista.** Ejemplo2.

Ejemplo de referencia a figura (figura 1.1).



Figura 1.1: Logotipo de la UC3M ©UC3M

La idea...

1.1. Introduccion a la robótica Open-Source TODO

TODO TODO TODO TODO TODO (Lo siguiente son ideas sueltas sin orden)

En los ultimos años blablabla impresoras 3d. El uso de esta nueva herramienta ha revolucionado blablabla hacer robots. La fabricación de piezas, que en la mayoría de los caso resultaba inaccesible para los estudiantes interesados en construir sus propios prototipos, se ha visto impulsada enormemente. Gracias a esto ahora tenemos a nuestro alcance la posibilidad de fabricar de una forma muy rápida y económica robots de un nivel superior al que estamos acostumbrados a ver.

Una de las ramas de la robótica que mas puede haberse beneficiado de este hecho es la robótica humanoide.

El campo de la robótica humanoide tiene como objetivo el desarrollo de robots antropomórficos. El motivo principal es favorecer el hecho de que los robots se desenvuelvan en entornos diseñados por y para seres humanos. Otros robots, como los robots móviles movidos por ruedas, suelen necesitar entornos modificados convenientemente para poder suplir las necesidades que se requieren. Por el contrario, un robot humanoide posee la ventaja de poder interaccionar con un entorno ya existente, así como la posibilidad de utilizar herramientas humanas. Por supuesto, el acceso de un alumno a un robot de este tipo parece algo impensable. Es por ello que en la asociación de robótica existe un linea de investigación de mini-humanoides. A priori, puede parecer que este tipo de robots están muy alejados de los humanoides de los laboratorios mas famosos, sin embargo, suponen un punto de partida viable para el comienzo de su estudio. Los miembros del grupo de mini-humanoides, tienen la oportunidad de trabajar libremente con robots reales, experimentando con su construcción, programación y modificación.

Acostumbrados a verla como una linea de investigación inaccesible . blablabla es cara y practicamente la única forma que tiene un estudiante de darle caña es con simulaciones.

Si bien es cierto, esto no significa que los robots imprimibles vayan a sustituir a los robots caros de laboratorio, pero suponen una primera aproximación a ellos.

TODO TODO Esto hay que ver si se hila con el siguiente apartado o se separa

1.2. Asociación de Robótica de la Universidad Carlos III

La Asociación de Robótica de la Universidad Carlos III de Madrid, AsRob, surgió en el año 2006 con el objetivo de acercar la robótica a los alumnos de la universidad que compartían inquietu-

des e interés por el campo de la robótica.

A día de hoy, la asociación cuenta con mas de cien (j- TODO cuantos) miembros activos repartidos en cinco lineas de investigación independientes, como son:

- **Vehículos Aéreos no Tripulados (UAVs).**
- **Robot Devastation.**
- **Robots Personales de Competición.**
- **Robots Mini-Humanoides.**
- **Impresoras 3D Open-Source.**

Sin embargo, cabe destacar que aunque se trata de proyectos diferentes, existe una gran sinergia entre ellos. Particularmente, los miembros de la linea de Robots Mini-Humanoides, estamos muy ligados al estudio de las impresoras 3D, investigando diferentes técnicas de impresión, diseño de estructuras y materiales. Ejemplo de ello es el proyecto MYOD (j- TODO referencia), en el que se propone la construcción de robots mini-humanoides compuestos integralmente con piezas impresas y replicables.

1.3. Descripción del proyecto TODO

blablablabla mi proyecto es la monda

1.4. Estructura del documento TODO

A continuación y para facilitar la lectura del documento, se detalla el contenido de cada capítulo.

- En el capítulo 1 se realiza una introducción.
- En el capítulo 2 se hace un repaso...

Capítulo 2

Descripción de la linea de investigación

2.1. Linea de investigación de robots mini-humanoides

La sección de la asociación que enmarca este trabajo es la linea de investigación de robots mini-humanoides.

Los robots mini-humanoides son robots antropomórficos con una altura máxima de 50cm, tal y como indica la normativa del campeonato CEABOT (¡- TODO referencia al reglamento). De fomar orientativa, tomando como referencia la Humanoid League del campeonato RoboCup, el tamaño de los robots mini-humanoides es ligeramente inferior al de los participantes de la división "KidSize". Robots participantes de la RoboCup en la división KidSize como son el DARwIn-OP de Robotis o el Nao (figura 2.1) de Aldebaran no entrarían dentro de la definición mini-humanoide, ya que sobrepasan las dimensiones máximas estipuladas.



Figura 2.1: Robots Nao y DARwIn-OP

2.2. Campeonato CEABOT

Desde el año 2006, el campeonato nacional CEABOT reúne anualmente a robots mini-humanoides procedentes de universidades españolas y equipos independientes. Durante tres días, los equipos tienen la posibilidad de presentar a sus robots a diferentes pruebas de habilidad para demostrar sus capacidades. En el reglamento de la edición del 2014, existen un total de cuatro pruebas combinadas de diversa temática que ponen a prueba la locomoción y percepción y actuación sobre el entorno de los robots participantes. Las pruebas son puntuadas por separado, sumándose de forma proporcional a su dificultad en la clasificación final del concurso.

Las pruebas previstas son las siguientes:

- **Carrera de obstáculos.**

En la carrera de obstáculos los robots deben realizar de forma autónoma un recorrido de ida y vuelta sobre una pista de características fijas. El campo (figura 2.2) consiste en una superficie plana de color verde en cuya zona intermedia se colocan de forma arbitraria diferentes obstáculos inmóviles de color blanco. Estos obstáculos tienen forma paralelepípeda y unas dimensiones fijas de 20x20x50cm



Figura 2.2: Prueba de la carrera de obstáculos

El robot participante debe cruzar el campo de extremo a extremo, y una vez haya accedido a la zona de llegada debe darse la vuelta y realizar el recorrido en el sentido contrario. En esta prueba se puntuá favorablemente el menor tiempo ocupado y la mayor longitud recorrida, mientras que las caídas o bloqueos

que requieran la intervención de un juez, producen penalizaciones en la puntuación.

- **Escalera.**

La prueba de la escalera supone una combinación de las habilidades mecánicas y de sensorización de los robots. La prueba se desarrolla en un escenario formado por tres escalones de subida y tres escalones de bajada consecutivos (figura 2.3).

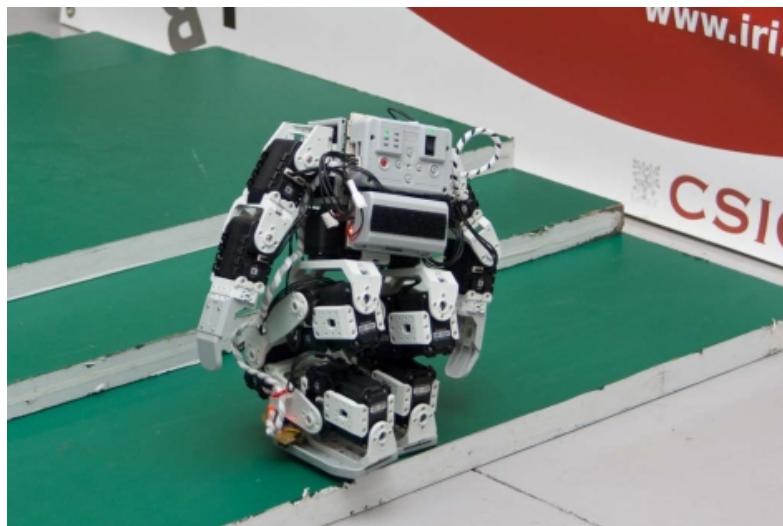


Figura 2.3: Prueba de la escalera

En este caso el robot debe sortear escalones con una altura fija e igual a 3cm, pero con amplitud variable. El desarrollo consiste en la superación de tres escalones descendentes, cruzar la cima de las escaleras y descender otros tres escalones hasta volver al suelo. De forma paralela a la prueba de navegación, se puntuán el número de escalones superados y el tiempo utilizado; mientras que las caídas y bloqueos que el robot no sea capaz de manejar por sí mismo contarán negativamente.

- **Lucha (Sumo).**

La prueba de sumo (figura 2.4) es una de las mas famosas del concurso. A diferencia del resto de pruebas, en el sumo los robots se enfrentan en parejas. Los duelos están constituidos por tres asaltos de dos minutos cada uno. El ring sobre el que se enfrentan los robots tiene forma circular, con un diámetro de 1.5m. Los robots compiten para derribar y/o sacar del ring a su adversario.



Figura 2.4: Prueba de sumo

■ Visión.

La prueba de visión se presenta como una novedad en la edición de 2014 del concurso. Por primera vez se implanta en la competición una prueba que obliga a los robots a portar una cámara y realizar procesamiento de imágenes para su superación. El tablero de juego se comparte con el campo de la carrera de obtáculos. En esta prueba, el robot se colocará en el centro del tablero, y a su alrededor se colocarán obstáculos (los mismos que en la carrera de obstáculos) en intervalos de 45°. En la parte superior de los obstáculos se colocará un rectángulo rojo con un código QR en su interior (figura 2.5). El robot deberá leer el código QR, en el que se le indicará una rotación que le permitirá encontrar el siguiente marcador. De esta forma, el robot deberá seguir una secuencia de rotaciones para pasar la prueba.

Palabras clave: palabraclave1, palabraclave2, palabraclave3.

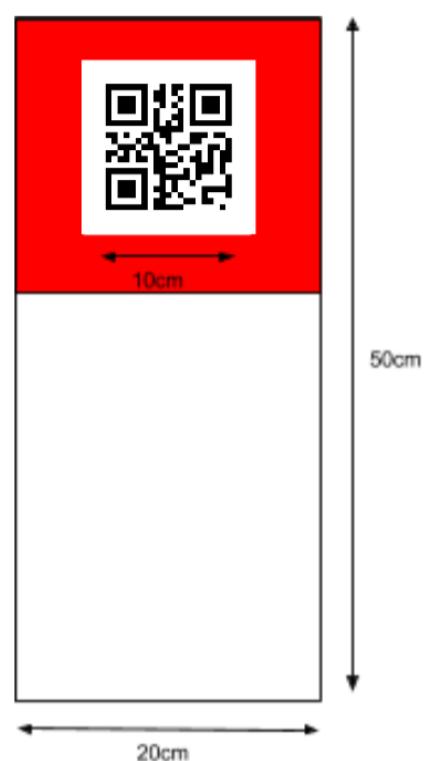


Figura 2.5: Marcador de la prueba de visión

Capítulo 3

Estado del arte

Este proyecto de resume en.....

Palabras clave: palabraclave1, palabraclave2, palabraclave3.

Capítulo 4

Objetivos TODO

4.1. Implantación de un controlador basado en visión por computador

El objetivo principal de este proyecto consiste en la implantación de un controlador para un robot humanoide que proporcione las capacidades necesarias para el desarrollo de algoritmos de interacción con el entorno basados en técnicas de visión por computador.

De este modo, se pretende dotar al robot de un comportamiento autónomo en ejercicios de interacción con el entorno tales como la navegación y la manipulación de objetos físicos (TODO : Pegarle un guantazo a otro robot).

- **Sistema de visión.**

Se realizará un estudio de las posibilidades existentes para incluir un sistema de visión, tanto a nivel de software tanto de hardware. Para ello se evaluarán las opciones actuales para elegir componentes, teniendo en cuenta el rendimiento que pueden proporcionar respecto a otros factores como su coste y su facilidad de adquisición.

- **Implementación de OpenCV.**

El control de los algoritmos se basará de las librerías de código abierto OpenCV. Sin embargo, el robot a parte de la cámara posee otro tipo de sensores, por lo que se requerirá realizar una programación que permita la coexistencia de los datos extraídos de la cámara con otros métodos de reconocimiento del entorno.

4.2. Estudio de sensores y actuadores

Durante la construcción del robot he utilizado diversos tipos de sensores y actuadores. En este proyecto, se han seleccionado algunos de ellos teniendo en cuenta su funcionamiento, consumo energético, conexión con el controlador... etc.

- **Estudio y elección de sensores.**

Actualmente, existe una amplia variedad de sensores de bajo coste aptos para su uso en este tipo de robots. El principal objetivo es seleccionar los sensores que pueden proporcionar una funcionalidad óptima aplicada a la programación y a las características físicas (tales como sus dimensiones y peso) del robot.

- **Estudio y elección de actuadores.**

De forma paralela, existen diversos actuadores adecuados en el mercado para su montaje en robots micro-humanoides. Aunque en este caso el factor mas determinante pueda ser el aspecto económico, existen varias alternativas a considerar, siendo crítico mantener unas proporciones de rendimiento y tamaño acordes al tipo de robot que se propone en el proyecto.

4.3. Diseño electrónico

- **Configuración de un controlador partido.**

Para favorecer la escalabilidad del sistema y su división en bloques funcionales, se ha decidido utilizar dos placas controladoras diferentes en una configuración de maestro y esclavo. El sistema se compone de un SBC y una placa de prototipado basada en un microcontrolador mas sencillo. De esta forma, el microcontrolador (esclavo) se encargará de almacenar y activar los actuadores del robot con una programación a bajo nivel, mientras que el SBC (maestro) será quien procese la información de la cámara y los sensores para...

- **Diseño y fabricación de una placa de expansión.**

El montaje propuesto requiere de múltiples conexiones eléctricas entre las placas de control, la alimentación, los sensores y los actuadores. Con la necesidad de realizar un montaje limpio y seguro, se ha diseñado y fabricado un placa de expansión que reune los conectores de cada elemento del sistema en un mismo punto.

- **Eleción de batería y alimentación.**

Uno de los aspectos mas críticos en el diseño electrónico de un sistema como el propuesto, es la autonomía de funcionamiento. El amplio número de actuadores de un robot humanoide se traduce en un requerimiento de energía.....
TODO

4.4. Diseño mecánico

Durante la construcción del robot se han diseñado diversas partes para posibilitar la integración de las diferentes partes del robot. Si bien es cierto que la plataforma parte de un robot comercial, la mayor parte de las piezas han sido rediseñadas e impresas para conseguir un conjunto mecánico optimizado a los componentes que se han montado.

- **Impresión 3D.**

La fabricación de las piezas se realizará mediante la utilización de una impresora 3D open-source. Esto significa que los diseños deberán cumplir una serie de características que los haga válidos para ser impresos, de forma que no solo se evaluarán las piezas por su modelo virtual, sino también por su estructura de impresión.

- **Plataforma móvil para rotar una cámara en varias direcciones.**

Una forma de potenciar la implantación de la cámara es dotando a ésta de un sistema de movilidad que le permita rotar en dos direcciones.

- **Modificaciones estructurales.**

Como decía al principio de este apartado, todas las piezas del robot han sido diseñadas con el objetivo de permitir el montaje de los actuadores, sensores, y placas de control seleccionados. A parte, las piezas se han optimizado para variar su rigidez y peso en función de las necesidades de cada zona del cuerpo.

4.5. Programación

Una vez montado todo el sistema se comenzará con la programación de los diferentes niveles de control del sistema.

- **Elección de un sistema operativo.**

Son varias las opciones disponibles que se pueden encontrar en internet a la hora de seleccionar el sistema operativo mas adecuado para un robot. En este proyecto se ha realiza una comparativa de diferentes distribuciones de Linux, con el objetivo de seleccionar la opción mas adecuada para el caso que se presenta.

- **Biblioteca de interfaz de las entradas y salidas del SBC.**

El SBC supone el elemento maestro que controla el resto de dispositivos. Es por ello que establecer las conexiones necesita implementar diferentes protocolos de comunicación entre componentes. Se propone la creación de una librería que simplifique el problema mediante el desarrollo de una interfaz sencilla que abstraiga el funcionamiento de cada dispositivo a un formato común.

- **Control de servos Dynamixel.**

Los servomotores de la familia Dynamixel son unos potentes actuadores que permiten al usuario controlar una amplia gama de variables mas allá de la posición inicial y final, como son la velocidad, el torque o incluso la temperatura de funcionamiento. Para llevar a cabo el movimiento sincronizado de los 19 servos es necesario el desarrollo de una biblioteca de funciones que se encarguen de parametrizar las variables de cada actuador y configurar automáticamente su valor para adaptarse a las condiciones y exigencias solicitadas en tiempo real.

- **Funciones de desplazamiento.**

Dado que se trata de un robot micro-humanoide con locomoción bípeda, otorgarle la capacidad de desplazarse omnidireccionalmente no es un asunto trivial. En este apartado se presenta el estudio de diferentes técnicas de marcha bípeda para buscar una aplicación factible que posibilite el desplazamiento del robot de una forma estable y controlada. (TODO cuando sepa qué haré exactamente tengo que reescribir esto)

- **Algoritmos de visión.**

Apoyándome en las librerías de OpenCV, la propuesta de este trabajo consiste en sacar datos del entorno del robot de una forma eficiente. Siendo un factor limitante la capacidad de procesamiento del robot, se buscarán funciones simples y optimizadas al problema propuesto.

- **Estrategias para la competición en CEABOT.**

La participación en el campeonato nacional CEABOT supone un gran reto para — además de ser un marco fantástico para la evaluación de los resultados obtenidos. Para finalizar el proyecto, se propone la presentación de algoritmos que permitan realizar las pruebas del campeonato utilizando las herramientas que se han desarrollado.

Palabras clave: TODO palabraclave1, palabraclave2, palabraclave3.

Capítulo 5

Elección de componentes

Como punto de partida, se han seleccionado una serie de componentes adecuados para dotar al robot con las capacidades necesarias para cumplir los objetivos.

5.1. Plataforma robótica

El primer paso para la realización de este proyecto fue el estudio y selección de las plataformas robóticas que pueden conseguirse actualmente. Dado que el objetivo es encontrar un robot humanoide sobre el que se pueda implantar un sistema de visión, es necesario analizar diversos aspectos; algunos mecánicos como el numero y fuerza de los actuadores, y otros electrónicos como la capacidad de procesamiento y velocidad del controlador. Sin embargo, dado que este proyecto es autofinanciado en su mayor medida, el factor económico también es un limitante destacable.

En el siguiente apartado se presenta un estudio las principales principales opciones.

5.1.1. Selección de una plataforma robótica

En el mercado existe una gran variedad de robots educativos que cumplen las características antropomórficas necesarias para ser considerados robots mini-humanoides. Los robots que se muestran a continuación son una recopilación de algunos de los modelos mas accesibles y extendidos.

Hitec Robonova

El Robonova (figura 5.1) es uno de los mini-humanoides mas extendidos a nivel mundial. Fue uno de los primeros robots de este

tipo que se fabricó comercialmente y marcó un antes y un después en su categoría. Es por esto que es muy común encontrar Robonovas en competiciones como Ceabot, ya que durante muchos años fue el robot mini-humanoide mejor equipado y más vendido. En la Asociación de Robótica de la Universidad Carlos III, se han utilizado Robonovas en competiciones y proyectos desde su fundación.



Figura 5.1: Hitec Robonova

El kit de fábrica cuenta con 16 grados de libertad. Sus actuadores son servos digitales HSR 8498HB, que desarrollan un torque de 7.4kg/cm. Cabe destacar de estos servos su función "Motion Feedback", es decir, su capacidad para leer posiciones y comunicárselas al controlador. La placa de control del Robonova está basada en un microcontrolador ATMega 128 y cuenta con hasta 40 pines GPIO (puertos binarios de entrada y salida), 8 entradas analógicas, 3 salidas PWM, puerto serie y conexión I2C. Gracias a esto, el Robonova es fácilmente ampliable con sensores y actuadores, no necesariamente de la misma marca. En cuanto al software, Hitec da soporte a la programación con RoboBasic, un entorno de desarrollo completo con un lenguaje basado en Basic.

Kondo KHR-3HV

El KHR-3HV (figura 5.2), del fabricante japonés Kondo, es uno de los mini-humanoides más avanzados actualmente. Puede presu-

mir de ser el modelo de serie mas utilizado en el campeonato Robo-One, siendo seleccionado por los equipos por su gran agilidad y tamaño compacto.



Figura 5.2: Kondo KHR-3HV

En su configuración standard, el KHR-3HV cuenta con 17 servomotores KRS-2552HV de 14kg/cm de torque. Dichos actuadores, además, incluyen un pequeño microcontrolador, lo que les permite conectarse en daisy chain. El robot incluye una controladora RCB-4, expandible con 10 entradas analogicas y 10 GPIOs, y con capacidad para controlar hasta 35 servos. El software de programación ofrecido por Kondo es el Heart to Heart V4, que puede ser descargado gratuitamente desde su web oficial. Es importante recalcar que gracias a su inmensa comunidad de usuarios, existen varios proyectos de código abierto con librerías que permiten programar el KHR-3HV en lenguajes mas convencionales, como C y Python.

Vstone Robovie-X

El Robovie-X, uno de los robots humanoides de la empresa Vstone, se presenta en tres versiones diferentes: Lite, Standard y PRO. La diferencia entre los tres modelos radica en el número y tipo de ser-

vos que montan, manteniendo comunes el resto de partes del robot. El modelo Standard (figura 5.3) posee 17 grados de libertad, movidos por servos VS-S092J que desarrollan un torque de 9.2kg/cm. El controlador es un VS-C1, el cual tiene 30 canales para controlar servos. Vstone también fabrica diversas placas de expansión para la conexión de sensores en el Robovie-X.



Figura 5.3: Vstone Robovie-X

Junto al robot se suministra el programa RobovieMaker2, necesario para programarlo. El método de programación está orientado a la construcción de diagramas de flujo desde los que se controlan tanto los movimientos como la lectura de sensores externos.

Robotis Bioloid

La empresa coreana Robotis, comercializa un kit robótico conocido como Bioloid. Este kit proporciona una amplia gama de piezas diferentes para montar distintos modelos de robots. La modularidad de los componentes le convierten en una base excelente sobre la que realizar modificaciones, pudiendo diseñar configuraciones alternativas con gran facilidad.

El robot Bioloid (figura 5.4) incluye 18 servos Dynamixel, modelo AX-12A o AX-18A, dependiendo de la versión del kit. Los ac-

tuadores Dynamixel están controlados internamente por un microcontrolador ATMega8. Gracias a él, estos servos permiten realizar funciones avanzadas tales como el control de velocidad, torque, temperatura de ejecución... etc, posibilitando procesar información de bajo nivel directamente dentro del actuador y pudiendo abstraer el control de la controladora del robot a un nivel superior.

En cuanto a su controladora, los kits proporcionan controladoras de Robotis de la serie CM, mas específicamente, la CM-5 en el caso del Bioloid Comprehensive y la CM-510 o CM-530 (según qué versión) en el Bioloid Premium y GP.



Figura 5.4: Bioloid Premium

- **Robotis CM-5.**

Cuenta con un microcontrolador ATMega128. Permite la conexión de sensores específicos de la marca en el bus TTL, como el Dynamixel AX-S1.

- **Robotis CM-510.**

Basada en un microcontrolador ATMega2561. Además de los sensores de la propia marca (Dynamixel AX-S1, giróscopo...), que pueden montarse conectados al bus TTL, tiene cinco puertos para la conexión de sensores de salida analógica. Adicionalmente, posee un puerto para conectar un receptor ZigBee y teleoperar el robot.

- **Robotis CM-530.**

Presentado como la evolución de la CM-510, en este caso el microcontrolador de la placa es un ARM Cortex STM32F103RE, de 32 bits. El resto de características son similares a las de la CM-510, tiene cinco puertos de expansión para sensores analógicos y un puerto para conectar un receptor ZigBee o Bluetooth.

En cuanto a la programación, Roboplus (TODO foto) es una suite de programas distribuida gratuitamente por Robotis para la programación de sus robots educativos. Destaca por ser un entorno con un lenguaje de programación (R+) muy visual e intuitivo, preparado para su uso por niños o gente sin conocimientos muy avanzados de programación. Sin embargo, esto lo convierte en un lenguaje de programación muy limitado, hasta el punto que no permite utilizar todo el potencial de los actuadores Dynamixel.

comparativa y elección final

5.2. Modificaciones básicas sobre el robot Bioloid

5.2.1. Alimentación

5.2.2. Cabeza móvil

5.3. Elección del controlador

5.3.1. Controlador de locomoción

Arbotix

Serie OpenCM

5.3.2. Controlador de visión

BeagleBone Black

Raspberry Pi

Comparativa y selección

5.4. Sensorización

5.4.1. Sensores de distancia

Infrarrojos

5.4.2. Sensor inercial

5.4.3. Cámara

5.4.4. Configuración final

Palabras clave: palabraclave1, palabraclave2, palabraclave3.

Capítulo 6

Descripción de las herramientas a utilizar

A continuación se presentan las herramientas básicas que serán necesarias durante el desarrollo del proyecto.

6.1. Herramientas de diseño y fabricación de piezas

Dado que la plataforma robótica seleccionada requiere modificaciones mecánicas para permitir el montaje de los componentes necesarios, se requiere construir una serie de piezas que sustituyan a las originales y que aporten al robot algunas características de las que carece. Todas las piezas del robot serán diseñadas con OpenSCAD e impresas posteriormente con una impresora 3D open-source.

6.1.1. OpenSCAD

OpenSCAD es un programa destinado a la creación de objetos sólidos tridimensionales. Se trata de software libre y es compatible con Linux/UNIX, Windows y Mac OS X. A diferencia de otros programas de diseño 3D, OpenSCAD no se centra en los aspectos artísticos del diseño, sino en el aspecto técnico. Por ello, es una aplicación muy interesante cuando nuestro objetivo es crear piezas mecánicas,y en este caso particular, para un robot.

La propiedad mas característica de OpenSCAD y que le hace diferente de otros programas de diseño como SolidWorks o FreeCAD, es su interfaz (figura 6.1). Este programa funciona como un compilador de objetos 3D, leyendo un script qu describe el objeto y renderizando el objeto a partir de ese archivo. Gracias a esto, el usuario tiene total

28 CAPÍTULO 6. DESCRIPCIÓN DE LAS HERRAMIENTAS A UTILIZAR

control sobre el proceso de modelado, permitiendo la realización de modelos variables a partir de parámetros configurables.

OpenCad también permite el diseño de modelos planos, siendo compatible con formatos como DXF. Sin embargo, para el caso de este proyecto, los archivos que nos interesa producir son los STL.

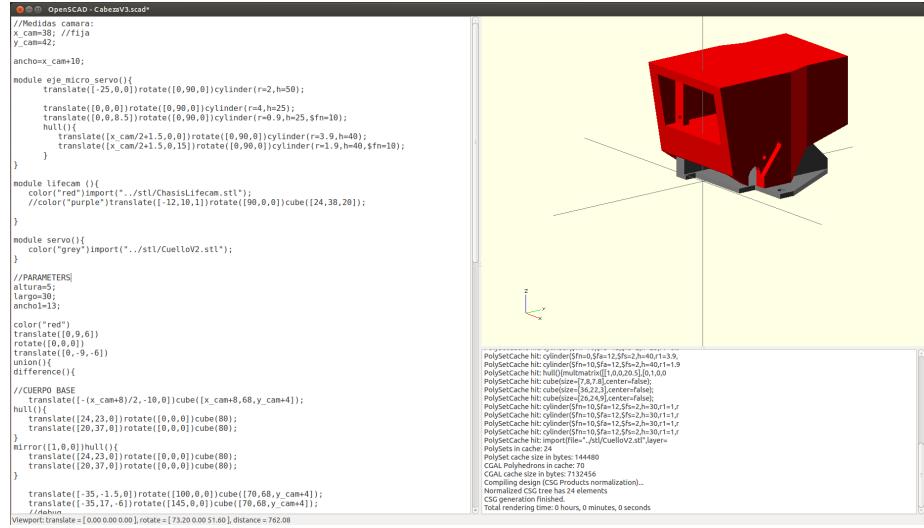


Figura 6.1: Pantalla del editor de OpenSCAD

6.1.2. Impresión 3d

Para la fabricación de las piezas que integran el robot, se ha utilizado una impresora 3D replicable open-source modelo Prusa i2 Air, de construcción casera. La configuración habitual de esta impresora ha sido mejorada con un extrusor Budaschnozzle 1.3 y una electrónica Sanguinololu 1.3b (figura 6.2).

Esta impresora, dado su funcionamiento, pertenece a la familia del modelado por deposición fundida. La materia prima que utilizan este tipo de impresoras es un rollo de filamento de plástico termofusible de entre 1.5 y 3mm de diámetro. En este caso, se utilizará ABS de 3mm. El filamento de plástico es dirigido a un extrusor que empuja el material a través de un conducto caliente conocido como hotend. Al llegar a este punto, el filamento se funde y se hace pasar por un agujero de salida de tamaño muy inferior al de entrada, produciendo hilos de material fundido. Durante la impresión, el extrusor deposita plástico fundido a lo largo de diferentes trayectorias con el objetivo de formar capas horizontales sólidas. Mediante la apilación de estas capas se consigue dotar de altura al modelo y crear la pieza requerida. A través del programa Repetier-Host, que se ocupa de

gestionar el funcionamiento de la impresora desde el ordenador, y partiendo de los archivos STL que han sido generados anteriormente desde OpenSCAD, la impresora nos permite desarrollar prototipos y piezas finales para el proyecto.

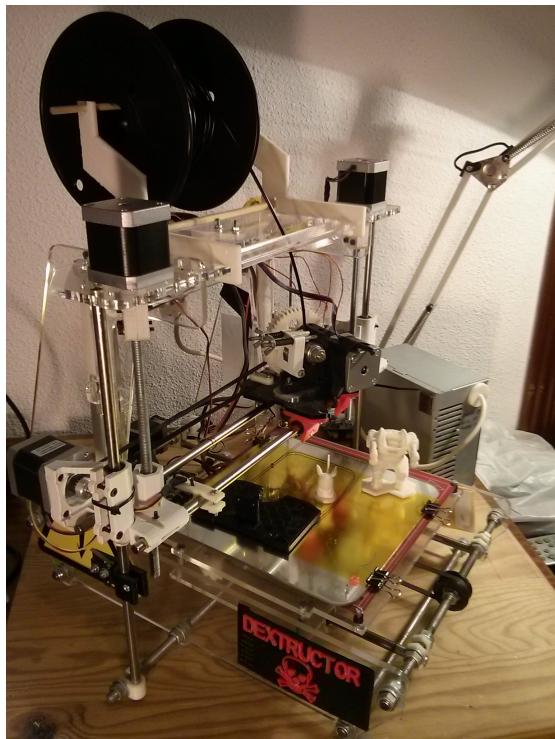


Figura 6.2: Impresora 3D Prusa i2 Air

6.2. Herramientas de diseño de circuitos

Dado que se va a necesitar expandir las conexiones físicas del controlador, se requiere diseñar una placa de expansión que permita realizar un montaje adecuado del sistema.

6.2.1. KiCad

KiCad es una de las herramientas libres más avanzadas para el diseño electrónico asistido (EDA) que pueden encontrarse a día de hoy. Cuenta con un grupo de más de 150 desarrolladores y una extensa comunidad de usuarios entre quienes se pueden destacar laboratorios como el CERN.

KiCad permite crear diseños electrónicos pasando por todas las fases del proceso, desde la idea a los ficheros de fabricación y la

30 CAPÍTULO 6. DESCRIPCIÓN DE LAS HERRAMIENTAS A UTILIZAR

simulación 3D de la placa. El entorno (figura 6.3) cuenta con cuatro aplicaciones independientes:

- **Eeschema.** Editor del esquemático.
- **Pcbnew.** Editor de la placa de circuito impreso.
- **Gerbview.** Visor de archivos GERBER
- **Cvpcb.** Editor de huellas para componentes.

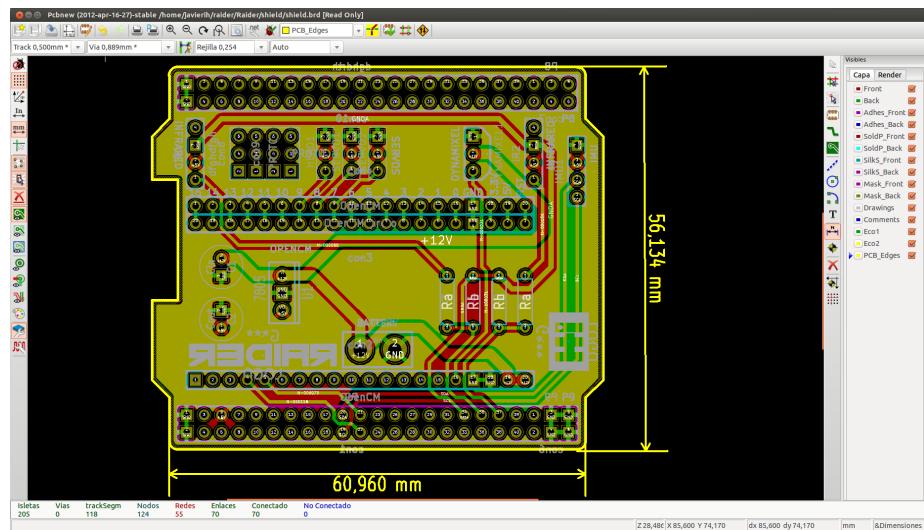


Figura 6.3: KiCad

KiCad es un programa multiplataforma, escrito con wxWidgets para ser ejecutado en FreeBSD, Linux, Microsoft Windows y Mac OS X. Existe una amplia colección de bibliotecas de componentes, a las cuales se suma la posibilidad para el usuario de crear componentes personalizados. Además, existen herramientas para importar los componentes de otros EDA, como por ejemplo desde EAGLE.

6.3. Herramientas de programación

Este proyecto conlleva una programación a diferentes niveles, desde el control de movimientos de los actuadores desde un microcontrolador hasta el diseño de algoritmos de navegación a alto nivel.

6.3.1. OpenCV

OpenCV es una biblioteca libre de visión artificial. Fue creada en 1999 por Intel y liberada un año mas tarde en la conferencia

anual IEEE Conference on Computer Vision and Pattern Recognition. Existen infinidad de aplicaciones, como la programación de detección de movimiento para cámaras de seguridad o la detección visual de características importantes en diferentes etapas del proceso de fabricación de un producto. La gran popularidad de OpenCV se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación.

OpenCV también es multiplataforma y existen versiones para GNU/Linux, Windows y Mac OS X. Entre sus funcionalidad, se abarcan campos de la visión por computador como el reconocimiento de objetos, calibración de cámaras, visión 3D y visión robótica. Además, OpenCV está programado de forma muy optimizada en C y C++, lo que le otorga una alta eficiencia y aptitud para aplicaciones en tiempo real.

6.3.2. Qt Creator

Qt Creator (figura 6.4) es un entorno de desarrollo multiplataforma y open-source desarrollado por la compañía noruega Trolltech. Soporta C++, JavaScript y QML. El editor incluye resalto de sintaxis y funciones de autocompletado, muy útiles cuando se trabaja con proyectos de una extensión media o alta. Qt Creator utiliza el compilador de C++ de GNU Compiler Collection (o lo que es lo mismo, GCC). Qt Creator interpreta por defecto archivos de CMake.

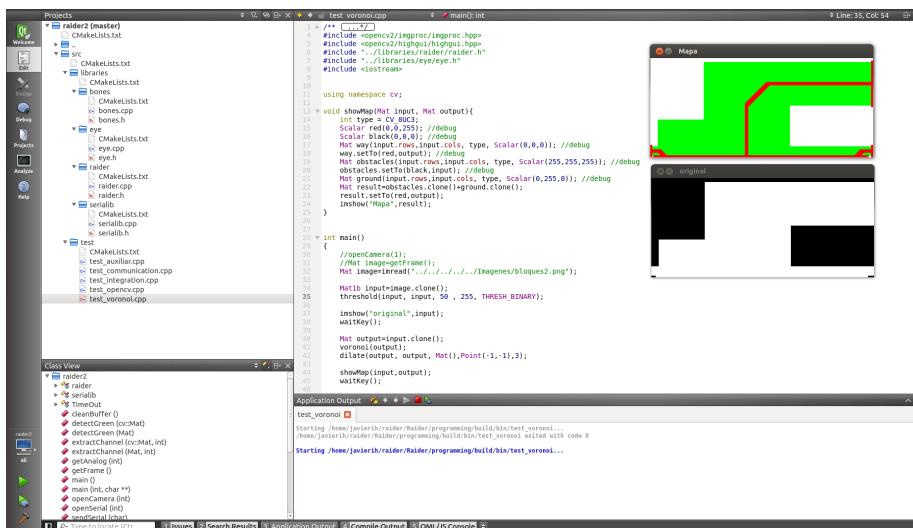


Figura 6.4: Qt Creator

32 CAPÍTULO 6. DESCRIPCIÓN DE LAS HERRAMIENTAS A UTILIZAR

6.3.3. CMake

CMake es una herramienta open-source para la administración de la generación de código. El nombre CMake es una abreviatura de cross platform make (make multiplataforma).

CMake es un conjunto de herramientas creado para construir, probar y empaquetar software. Se utiliza para controlar el proceso de compilación del software usando ficheros de configuración sencillos e independientes de la plataforma. CMake genera makefiles nativos y espacios de trabajo que pueden usarse en el entorno de desarrollo deseado. CMake soporta la generación de ficheros para varios sistemas operativos, lo que facilita el mantenimiento y elimina la necesidad de tener varios conjuntos de ficheros para cada plataforma.

El proceso de construcción se controla creando uno o más ficheros CMakeLists.txt en cada directorio del proyecto.

6.3.4. CM9 IDE

CM9 IDE es un entorno de programación basado en Arduino IDE, preparado para programar placas electrónicas de la serie OpenCM. Soporta programación en C/C++ y es compatible con la mayoría de las librerías de Arduino. CM9 (figura 6.5) constituye un entorno centralizado desde el cual se puede escribir código, compilarlo y cargarlo en la placa OpenCM sin necesidad de ninguna otra aplicación adicional.

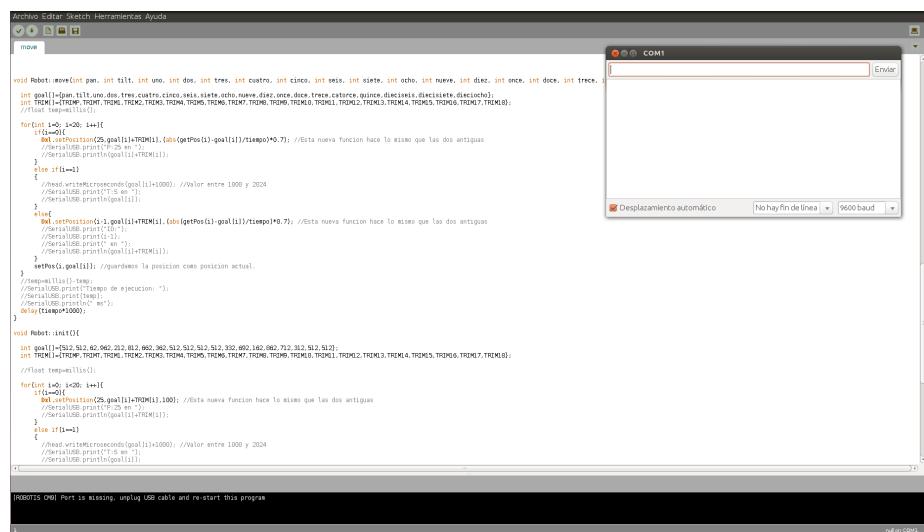


Figura 6.5: CM9 IDE

6.3.5. Git

Git es un software de control de versiones diseñado para controlar el código de un proyecto en sus distintas iteraciones. Este proyecto se ha desarrollado en un repositorio online de Github, que será liberado cuando TODO (cuando?)

Palabras clave: palabraclave1, palabraclave2, palabraclave3.

Capítulo 7

Desarrollo

Este proyecto de resume en.....

7.1. Diseño de las partes mecánicas

7.1.1. Cabeza

7.1.2. Tronco

7.1.3. Brazos

7.1.4. Piernas

7.2. Montaje del controlador

7.2.1. Esquema de montaje

7.2.2. Sistema de alimentación

7.2.3. Adecuación de sensores

7.2.4. Desarrollo de una placa de expansión

7.2.5. Puesta en marcha del controlador

Palabras clave: palabraclave1, palabraclave2, palabraclave3.

Capítulo 8

Programación

8.1. Sistema de locomoción

Antes de comenzar con esta sección, es importante señalar algunos detalles importantes. Si bien es cierto, el control del robot girará en torno a un algoritmo de visión, su funcionamiento de apoyará en un control de locomoción robusto que permita al robot realizar desplazamientos seguros sobre el terreno. Por tanto, dado que se trata de un robot bípedo, la programación de movimientos no es un tema trivial como lo podría ser en un robot con ruedas.

Para conseguir que Raider pueda moverse con soltura se han seguido varios pasos, cada cual de un nivel superior al anterior.

8.1.1. Movimiento del servo PWM

El control de servo PWM de Raider, situado en su cabeza, se realiza con la biblioteca Servo.h originalmente desarrolladas para Arduino y que posteriormente han sido portadas para su utilización en OpenCM. Dada la función de este servo, no será necesario realizar un control de su velocidad, por lo que un simple control de su posición será suficiente.

La biblioteca Servo.h nos permite controlar el movimiento de un servo a partir de un valor de posición, y se encarga de producir la señal PWM correspondiente. Aunque los métodos utilizados pueden encontrarse en la documentación de Arduino (TODO), a continuación introduzco una breve explicación de los que han sido utilizados en este proyecto:

Servo::attach(int)

Con esta función configuramos un pin de la OpenCM para que actúe como emisora de señales PWM.

Servo::writeMicroseconds(int)

Para mover el servo se utiliza la función writeMicroseconds, cuyo parámetro define la amplitud (TODO) de la onda PWM . Si bien es cierto, existe otra función paralela llamada write(int) que directamente utiliza como parámetro la posición en grados, se ha utilizado writeMicroseconds por su mayor precision. Para (TODO)

8.1.2. Movimiento de los actuadores Dynamixel

Para comunicarse con los actuadores Dynamixel (de los que hablamos en TODO) debemos utilizar su protocolo particular. Los servos son controlados mediante el envío de paquetes de datos binarios. Existen dos tipos de paquetes en el protocolo: Los paquetes de instrucciones, que son los que envía el controlador a los servos; y los paquetes de estado, que son los que los servos envían al controlador.

Cada servo tiene una ID, o dicho de otra forma, un número de identidad propio e irrepetible que identifica a un servo particular dentro del bus. La comunicación en el bus se realiza mediante el intercambio de paquetes de instrucciones y estados con una ID concreta. Por esta razón, en un mismo bus no deben existir servos con la misma ID, ya que provocarán colisiones entre los paquetes e impedirán el correcto funcionamiento del sistema. Sin embargo, estas ID son fácilmente reprogramables y pueden modificarse realizando una escritura sobre el registro 3 (0X03).

El protocolo de comunicación utilizado es una comunicación serie asíncrona de 8 bits, con 1 bit de Stop y sin paridad.

La conexión, de tipo Half Duplex, no permite la transmisión y recepción de paquetes de forma simultánea. Esto la convierte en una conexión bastante típica en los sistemas que utilizan un solo bus de comunicación. Como en el mismo bus existe más de un dispositivo, todos deben permanecer en modo de escucha salvo el que esté transmitiendo en ese instante. El controlador principal, la placa OpenCM 9.04, asigna la dirección del bus en modo escucha, y solo cambia la dirección del bus a modo de envío mientras manda un paquete. (TODO revisar)

Los Dynamixel AX-12A poseen una tabla de registros (tabla TODO) sobre la cual podemos variar varios parámetros referente a su

estado y su funcionamiento. La tabla de registros puede consultarse en TODO .

Para realizar una rotación simple en un servomotor, sería suficiente con escribir en el registro 32 (Goal Position) un valor comprendido entre 0 y 1023, y el servo se situará inmediatamente en esa posición. Sin embargo, existen otros parámetros interesantes en el mapa de registros que conviene controlar, como la posición instantánea, la velocidad de giro, el consumo eléctrico o incluso la temperatura del dispositivo.

Para mover los 19 AX-12A de Raider se utilizan los parámetros de posición objetivo (Goal Position) y velocidad de giro (Moving Speed) de forma combinada. Dado que está programación se ha realizado desde la OpenCM 9.04 se ha utilizado la biblioteca Dynamixel.h, que funciona como una macro para leer y escribir en los registros de forma sencilla y eficiente. Dentro de la biblioteca utilizaremos la función writeWord con la siguiente sintaxis:

```
Dxl.writeWord(
    Dynamixel_Motor_Number,
    Address_Number,
    Address_Data
);
```

A modo de ejemplo, para asignar una velocidad de $3,5 rad/s$ al servo con la ID 5, primero calcularíamos el valor correspondiente para una resolución de 10 bits. Según el manual de los servos Dynamixel (TODO citar) AX-12A, la velocidad máxima de estos servomotores es de $114 rpm$. Por tanto, se realizaría la siguiente conversión:

$$3,5 \cdot \frac{rad}{s} \cdot \frac{60s}{2\pi rad} \cdot rpm \cdot \frac{1024}{114 rpm} = 300,216 \simeq 300$$

Dentro del código, utilizaremos la función writeWord para asignar este valor en el registro 32 (Moving Speed):

```
Dxl.writeWord(5,32,300);
```

Seguidamente, asignaríamos al servo una posición final siguiendo el criterio de la figura 8.1

Continuando con el ejemplo, calcularemos el valor que debemos darle al registro para mover el servo a una posición de 120° , teniendo en cuenta que las especificaciones nos indican una amplitud de giro total de 300° reales con una resolución de 10 bits. De esta forma, haríamos la siguiente conversión:

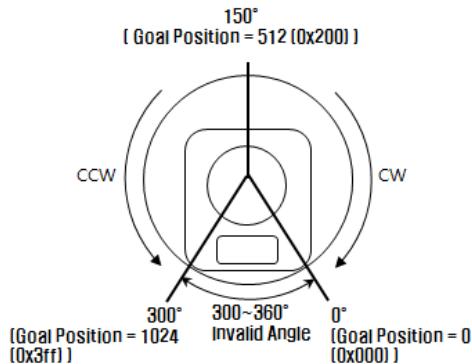


Figura 8.1: Amplitud de giro de un AX-12A

$$120^\circ \cdot \frac{1024}{300^\circ} = 409,6 \simeq 410$$

En nuestro programa escribiríamos en el registro 30 (Goal Position) de la siguiente forma:

```
Dxl.writeWord(5,30,410);
```

Como resumen, con estos pasos hemos conseguido mover el servo con la ID 5 (Que corresponde al codo del brazo izquierdo de Raider), a una posición de 120° con una velocidad de $3,5rad/s$

8.1.3. Movimiento sincronizado de las articulaciones

En el apartado anterior se ha mostrado cómo se realiza el movimiento de un servo, sin embargo, para mover el cuerpo del robot necesitaremos mover todos al mismo tiempo de una forma sincronizada. Si en el apartado anterior utilizábamos la posición objetivo y la velocidad de movimiento como parámetros, en este punto, por comodidad a la hora de programar, utilizaremos como parámetros la posición objetivo de los 20 servos, su posición actual y el tiempo total durante el que se realizará su movimiento entre ambos puntos.

Este es quizás uno de los apartados más críticos a la hora de diseñar las funciones que moverán el robot. Se pretende programar una biblioteca que permita mover 20 servos simultáneamente, con velocidades diferentes condicionadas por un tiempo de ejecución común. De esta forma, los servos cuya posición objetivo sea lejana a su posición actual se moverán con una velocidad mayor que la de los servos cuya posición objetivo sea cercana a su posición actual.

Las funciones pueden encontrarse en la biblioteca *openCM/raider_motion.h*:

class Robot

La clase Robot abarca todas las funciones relativas al movimiento de Raider (aunque por el momento en esta sección solo se presenta algunas de ellas), y abstrae su controlador principal, la BeagleBone Black de la parte de locomoción.

Dentro de la clase, encontramos tres variables miembros importantes:

- **int currentPosition[20].**

currentPosition es un array de 20 posiciones destinado a almacenar los valores de la posición actual de las 20 articulaciones del robot con valores comprendidos entre 0 y 1023. El primer valor es el servo AX-12A, el segundo es el servo PWM de la cabeza y a partir de ese punto están el resto de AX-12A ordenados según su ID, del número 1 al 18.

- **int targetPosition[20].**

targetPosition sigue la misma estructura de currentPosition, con la diferencia de que en este caso los valores guardados en el array corresponderán con la posición objetivo o posición final de las articulaciones.

- **int TRIM[20].**

Por último, TRIM es un array de trims. Un trim es una variable de ajuste para calibrar la posición de los servos. Tanto los servos Dynamixel como los servos PWM suelen tener un pequeño error en su posición cero. Los trimmers constituyen un offset aplicado individualmente a cada servo en absolutamente todos los movimientos que se realizarán durante el programa. Las holguras y otros factores pueden provocar el desajuste de estos valores, por lo que es necesario volver a calibrarlo cada cierto tiempo. Una mala calibración de los trims puede radicar en problemas de asimetrías en movimientos, y por tanto, resultados inesperados.

Robot::Robot()

El constructor de la clase Robot tiene como función la apertura del bus de control para los servos AX-12A, la configuración del servo PWM y la asignación de trims en el array de trims.

void Robot::setTargetPosition(int,int,... int)

setTargetPosition accede directamente al miembro privado targetPosition[20] para asignarle nuevos valores.

void Robot::setTargetOffset(int,int,... int)

setTargetOffset varía los valores del miembro privado targetPosition[20] para sumarles un valor. La función permite variar una posición con un giro determinado sin necesidad de conocer la posición actual de la articulación.

void Robot::updateCurrentPosition()

Esta función tiene un funcionamiento sencillo, se ocupa de volver los datos de la posición objetivo en la posición actual. Es la forma que tiene el robot de actualizar su posición actual tras un movimiento.

void Robot::move(float)

La función move es la mas importante de todas, ya que es la función que se encarga de mover las articulaciones. A esta función se le pasa un valor de tiempo expresado en segundos, y tal y como se comentó al principio de este apartado, será el tiempo en el que los servos pasarán de la posición actual (currentPosition[20]) a la posición final (targetPosition[20]).

Para ello, la función calcula la amplitud del movimiento y asigna una velocidad independiente para ese servo. Gracias a esto, todos los servos empiezan y terminan de moverse al mismo tiempo y permiten un control mas sencillo de las inercias entre movimientos consecutivos.

8.1.4. Funciones de movimientos combinados

Llegado este punto hemos abordado como mover un servo y como mover los 20 servos de forma coordinada. En este apartado se presentan algunas funciones intermedias entre lo comentado y movimientos de alto nivel como puede ser el desplazamiento bípedo.

Para facilitar la programación de movimientos mas complejos se han programado una serie de utilidades que permiten mover los servos en pequeños grupos que desempeñan una función común. Estas funciones modifican los valores del array de posiciones finales, targetPosition[20], lo que quiere decir que para efectuar el movimiento será necesario realizar una llamada a la función move(float).

Por tanto, es posible la utilización de varias funciones en un mismo movimiento, dando la posibilidad de sumar sus modificaciones y superponer su utilidades.

void movHead(int)

movHead es una función que permite mover el servo PWM de la cabeza del robot. Sirve para mover la cámara independientemente de la posición del robot.

void movVertical(int,int)

void movLateral(int,int)

void movFrontal(int,int)

8.1.5. Creación de movimientos completos

8.1.6. Controlador de movimiento

8.2. Algoritmos de visión

Palabras clave: palabraclave1, palabraclave2, palabraclave3.

Capítulo 9

Evaluación de resultados

Se presentan a continuación las conclusiones...

9.1. Pruebas de funcionamiento

9.2. Conclusión

Una vez finalizado el proyecto...

9.3. Situación y desarrollos futuros

Un posible desarrollo...

Bibliografía

- [1] M. González-Fierro, A. Jardón, S. Martínez de la Casa, M.F. Stoelen, J.G. Víctores, and C. Balaguer. Educational initiatives related with the ceabot contest.