



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

En este proyecto se ha desarrollado la plataforma robótica mini-humanoide Raider, (Robot Antropomórfico para la Investigación y Desarrollo en Entornos Reales) con capacidad para actuar de forma autónoma basándose en algoritmos de visión por computador. Para ello se ha escogido un robot mini-humanoide comercial del que se han extraído sus servomotores, y se ha diseñado una nueva estructura con piezas imprimibles en una impresora 3D. Sobre el robot, se ha integrado un sistema de procesamiento de imágenes formado por una cámara USB y un controlador desarrollado sobre un ordenador de tamaño reducido, la BeagleBone Black. También se ha realizado un estudio de sensores y actuadores aptos para su montaje en un robot mini-humanoide. Entre ellos se han seleccionado aquellos que aumentan las capacidades del robot.

Tras fabricar y montar las nuevas piezas se ha procedido a programar el robot. Se han seguido tres líneas: Locomoción, sensorización y visión. En la programación de la locomoción se presentan los pasos que se han seguido desde el movimiento de una articulación simple hasta la combinación de estos movimientos para producir movimientos más complejos como la caminata o el control del equilibrio. En el apartado de visión, se han estudiado y desarrollado técnicas de path planning basadas en la búsqueda de trayectorias mediante la detección y esquelitización del espacio navegable basada en el algoritmo de Zhang-Suen. Adicionalmente, se han programado otras funciones como el tracking de una pelota o la lectura de códigos qr. Por último, en la sensorización, se han hecho diferentes librerías para el control desde la BeagleBone Black de los sensores que se montan en el robot y la extracción de datos útiles que apoyan a la parte de visión. El procesamiento de imágenes se ha combinado con la información recibida por los sensores para diseñar aplicaciones aptas para la competición en CEABOT y otros eventos.

**Palabras clave:** palabraclave1, palabraclave2, palabraclave3.



# Abstract

(El resumen en inglés)

**Keywords:** keyword1, keyword2, keyword3.



# Índice general

<b>Agradecimientos</b>	<b>v</b>
<b>Resumen</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Linea de investigación de robots Mini-Humanoides . . . . .	1
1.2. Marco de trabajo . . . . .	2
1.3. Campeonato CEABOT . . . . .	3
1.3.1. Carrera de obstáculos . . . . .	4
1.3.2. Escalera . . . . .	5
1.3.3. Sumo . . . . .	5
1.3.4. Visión . . . . .	6
1.4. Estructura del documento TODO . . . . .	7
<b>2. Estado del arte</b>	<b>9</b>
2.1. Plataformas robóticas mini-humanoides . . . . .	9
2.2. Competiciones . . . . .	9
2.2.1. RoboGames . . . . .	9
2.2.2. Robo-One . . . . .	9
2.2.3. Robocup . . . . .	9
2.2.4. Ceabot . . . . .	9
2.3. Locomoción . . . . .	9
2.4. Visión . . . . .	10
<b>3. Objetivos</b>	<b>11</b>
3.1. Desarrollar una plataforma robótica mini-humanoide	11
3.1.1. Estudiar los componentes que necesita un robot humanoide . . . . .	11
3.1.2. Integrar una cámara USB . . . . .	11
3.1.3. Integrar un controlador . . . . .	12
3.1.4. Agregar sensores y actuadores . . . . .	12

3.1.5. Diseñar un sistema de alimentación . . . . .	12
3.1.6. Realizar las modificaciones estructurales que sean pertinentes . . . . .	13
3.2. Puesta en marcha y programación . . . . .	13
3.2.1. Programar la locomoción bípeda . . . . .	13
3.2.2. Programar sensores . . . . .	14
3.2.3. Desarrollar algoritmos de visión . . . . .	14
3.2.4. Desarrollar aplicaciones de competición . . . . .	14
<b>4. Elección de componentes</b>	<b>15</b>
4.1. Plataforma robótica . . . . .	15
4.1.1. Selección de una plataforma robótica . . . . .	16
4.2. Modificaciones estructurales . . . . .	20
4.2.1. Cabeza móvil TODO IMAGEN . . . . .	21
4.2.2. Cintura móvil TODO IMAGEN . . . . .	21
4.3. Sensorización TODO . . . . .	22
4.3.1. Sensores de distancia . . . . .	22
4.3.2. Sensores inerciales TODO . . . . .	24
4.3.3. Cámara TODO . . . . .	26
4.4. Elección del controlador TODO . . . . .	27
4.4.1. Controlador de locomoción TODO . . . . .	28
4.4.2. Controlador de visión TODO . . . . .	30
4.4.3. Diseño del sistema TODO . . . . .	33
4.5. Alimentación . . . . .	34
4.5.1. Batería . . . . .	35
4.5.2. Regulador de tensión . . . . .	37
<b>5. Descripción de las herramientas a utilizar</b>	<b>41</b>
5.1. Herramientas de diseño y fabricación de piezas . . . . .	41
5.1.1. OpenSCAD . . . . .	41
5.1.2. Impresión 3d . . . . .	42
5.2. Herramientas de diseño de circuitos . . . . .	43
5.2.1. KiCad . . . . .	43
5.3. Herramientas de programación . . . . .	44
5.3.1. OpenCV . . . . .	44
5.3.2. Qt Creator . . . . .	45
5.3.3. CMake . . . . .	46
5.3.4. CM9 IDE . . . . .	46
5.3.5. Git . . . . .	47

<b>6. Diseño de las partes mecánicas TODO</b>	<b>49</b>
6.1. Cabeza . . . . .	49
6.2. Cintura . . . . .	49
6.3. Tronco . . . . .	50
6.4. Brazos . . . . .	50
6.5. Piernas . . . . .	50
6.6. Pies y tobillos . . . . .	50
6.7. Lista de piezas . . . . .	50
<b>7. Desarrollo TODO MONTAJES</b>	<b>55</b>
7.1. Adecuación de sensores . . . . .	55
7.1.1. Infrarrojos Sharp . . . . .	55
7.1.2. Brújula y sensor inercial . . . . .	58
7.2. Desarrollo de una placa de expansión TODO . . . . .	59
7.3. Montaje . . . . .	59
7.3.1. Montaje de la placa de expansión . . . . .	59
7.3.2. Integración de la cámara en la cabeza . . . . .	60
<b>8. Programación TODO</b>	<b>61</b>
8.1. Configuración de la BeagleBone Black . . . . .	61
8.1.1. Sistema operativo . . . . .	61
8.1.2. Instalación de librerías . . . . .	62
8.1.3. Configuración de la cámara . . . . .	63
8.1.4. Configuración de pines . . . . .	65
8.2. Sistema de locomoción TODO . . . . .	67
8.2.1. Movimiento del servo PWM . . . . .	67
8.2.2. Movimiento de los actuadores Dynamixel . . . . .	68
8.2.3. Movimiento sincronizado de las articulaciones	70
8.2.4. Funciones de movimientos combinados TODO	73
8.2.5. Creación de movimientos completos TODO .	73
8.3. Comunicación serie TODO . . . . .	74
8.3.1. Comunicación serie en OpenCM TODO . . . . .	74
8.3.2. Comunicación serie en BeagleBone . . . . .	76
8.3.3. Comunicación con módulo Bluetooth TODO .	76
8.4. Programación de sensores TODO . . . . .	77
8.4.1. Infrarrojos TODO . . . . .	77
8.4.2. IMU TODO . . . . .	77
8.4.3. Brújula TODO . . . . .	77
8.5. Algoritmos de visión TODO . . . . .	77
8.5.1. Análisis de trayectorias en navegación TODO	77
8.5.2. Búsqueda de líneas TODO . . . . .	77
8.5.3. Lectura de códigos QR TODO . . . . .	77

<b>9. Aplicaciones</b>	<b>79</b>
9.1. CEABOT 2014 . . . . .	79
9.2. Spain Experience . . . . .	79
<b>10. Presupuesto</b>	<b>81</b>
<b>11. Evaluación de resultados</b>	<b>83</b>
11.1. Pruebas de funcionamiento . . . . .	83
11.2. Conclusión . . . . .	83
11.3. Situación y desarrollos futuros . . . . .	83
<b>Bibliografía</b>	<b>85</b>

# Índice de figuras

1.1.	Robots Nao y DARwIn-OP . . . . .	2
1.2.	Robot Sylar de la Asociación de Robótica . . . . .	3
1.3.	Prueba de la carrera de obstáculos . . . . .	4
1.4.	Prueba de la escalera . . . . .	5
1.5.	Prueba de sumo . . . . .	6
1.6.	Marcador de la prueba de visión . . . . .	7
4.1.	Hitec Robonova ©Hitec RCD . . . . .	16
4.2.	Kondo KHR-3HV . . . . .	17
4.3.	Vstone Robovie-X . . . . .	18
4.4.	Bioloid Premium . . . . .	19
4.5.	Microservo PWM . . . . .	21
4.6.	Primer experimento. . . . .	23
4.7.	Segundo experimento. . . . .	23
4.8.	Tercer experimento. . . . .	24
4.9.	Sensor inercial MPU9150 . . . . .	25
4.10.	Brújula magnética CMPS03 . . . . .	26
4.11.	Microsoft LifeCam Cinema . . . . .	27
4.12.	Placa Arbotix . . . . .	29
4.13.	Placa CM900 . . . . .	30
4.14.	Placa OpenCM9.04 . . . . .	31
4.15.	SBC BeagleBone Black . . . . .	32
4.16.	Diagrama de conexiones . . . . .	34
4.17.	Batería LiPo . . . . .	36
4.18.	Convertidor DC-DC comercial, UBEC . . . . .	39
5.1.	Pantalla del editor de OpenSCAD . . . . .	42
5.2.	Impresora 3D Prusa i2 Air . . . . .	43
5.3.	KiCad . . . . .	44
5.4.	Qt Creator . . . . .	45
5.5.	CM9 IDE . . . . .	46
7.1.	Gráfica de tensión entre salida para un sensor infrarrojo	56

7.2.	Esquema de entradas analógica de una BeagleBone Black . . . . .	57
7.3.	Circuito de adecuación de un sensor infrarrojo . . . . .	58
7.4.	Esquema de puertos I2C de una BeagleBone Black . . . . .	58
7.5.	Circuito del bus I2C . . . . .	59
8.1.	Salida del comando lsusb . . . . .	63
8.2.	Comprobación de conexión para la webcam . . . . .	64
8.3.	Comprobación de compatibilidad para la webcam . . . . .	64
8.4.	Parámetros leídos de la cámara . . . . .	65
8.5.	Puertos serie en una BeagleBone Black . . . . .	66
8.6.	Algunas capas de Device Tree Overlays . . . . .	66
8.7.	Habilitación de capas . . . . .	67
8.8.	Lectura analógica del pin 4 . . . . .	67
8.9.	Amplitud de giro de un AX-12A . . . . .	70
8.10.	Esquema de puertos serie en una OpenCM 9.04 . . . . .	75

# Índice de cuadros

4.1. Especificaciones Raspberry Pi B . . . . .	31
4.2. Especificaciones BeagleBone Black . . . . .	33
4.3. Consumo medio . . . . .	37
6.1. piezas . . . . .	51
6.2. piezas . . . . .	52
6.3. piezas . . . . .	53
8.1. Movimientos programados . . . . .	74



# Capítulo 1

## Introducción

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 inquietudes e interés por el campo de la robótica.

A día de hoy, la asociación cuenta con mas de cien miembros activos repartidos en cinco líneas 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 línea de Robots Mini-Humanoides, están 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 ( ↴ TODO referencia ), en el que se propone la construcción de robots mini-humanoides compuestos íntegramente con piezas impresas y replicables.

### 1.1. Línea de investigación de robots Mini-Humanoides

La sección de la asociación que enmarca este trabajo es la línea de investigación de Robots Mini-Humanoides. Los robots mini-humanoides son robots antropomórficos con una altura menor 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 de la división Kid-Size como son el DARwIn-OP de Robotis o el Nao (figura 1.1) de Aldebaran no entrarían dentro de la definición de mini-humanoide, ya que sobrepasan las dimensiones máximas estipuladas.



Figura 1.1: Robots Nao y DARwIn-OP

El objetivo anual de este grupo de la asociación es la participación en el campeonato nacional para robots mini-humanoides CEABOT.

## 1.2. Marco de trabajo

Desde el año 2006, la Asociación de Robótica ha trabajado con robots humanoides destinados a la competición e investigación. Durante su actividad, se han utilizado diferentes plataformas robóticas y modificaciones que permitían a los robots del grupo ampliar sus capacidades y su competitividad. A lo largo del tiempo, pueden diferenciarse tres etapas caracterizadas por la plataforma robótica que fué empleada.

Entre 2006 y 2010, la plataforma empleada sobre la que se centraron los estudios y desarrollos fue el Robonova de Hitec (figura 1.2). En esta primera etapa se realizaron mejoras en el robot para alojar sensores adicionales, como brújulas y sensores infrarrojos. El funcionamiento del sistema pudo testarse en competiciones como el CEABOT y el RobotChallenge con excelentes resultados. En posteriores modificaciones, se sustituyó la placa de control del Robonova

por una placa Arduino, que permitiría una mayor libertad a la hora de programar tanto la locomoción del robot como su sensorización.



Figura 1.2: Robot Sylar de la Asociación de Robótica

En 2010, la asociación adquirió un kit de Bioloid. Esta plataforma, más moderna que el Robonova, permitiría realizar un mejor control de los movimientos del robot. Dentro de la universidad, se han realizado multiples proyectos basados en este robot. En competición, los Bioloids de la Asociación de Robótica llegaron a conseguir el segundo puesto en la edición de 2012 del campeonato Ceabot. Sin embargo, las capacidades de esta plataforma, en lo que a sensorización y programación se refiere, no son demasiado altas y con sus componentes de serie existe una gran limitación.

Por esta razón, desde el año 2013, se ha estudiado el modo de modificar estos robots con el objetivo de obtener una plataforma que permita explotar todo el potencial de sus actuadores y al mismo tiempo, facilitar la inclusión de nuevos componentes. Con este objetivo, se ligó fuertemente el estudio de la fabricación de piezas mediante técnicas de impresión 3d con su aplicación en robots Mini-Humanoides. Con piezas mecánicas nuevas y electrónicas libres, se prevee que las capacidades del robot aumenten exponencialmente. Esto convierte a los robots de la asociación en una base muy adecuada para la investigación.

### 1.3. Campeonato CEABOT

El campeonato nacional CEABOT reúne cada año a robots mini-humanoides procedentes de universidades españolas y de equipos independientes. Durante tres días, los equipos tienen la posibilidad de

presentar a sus robots a diferentes pruebas de habilidad en las que pueden 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, 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.

A continuación se muestran las pruebas previstas para la edición de 2014.

### 1.3.1. 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 1.3) 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

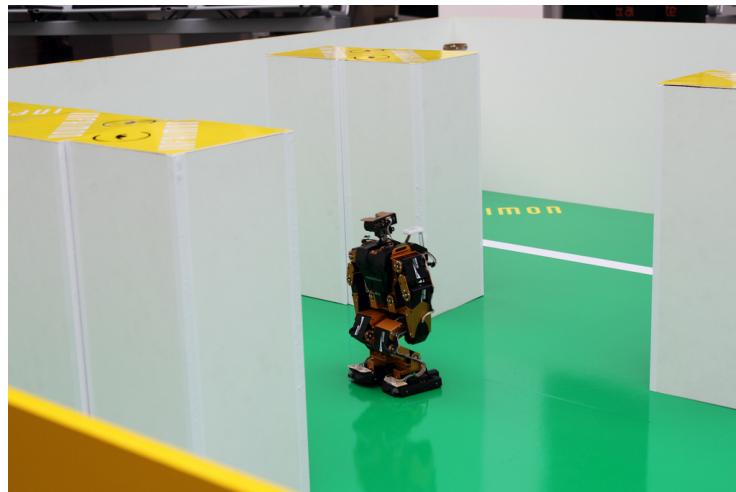


Figura 1.3: Prueba de la carrera de obstáculos

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 puntuará 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.

### 1.3.2. 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 1.4).

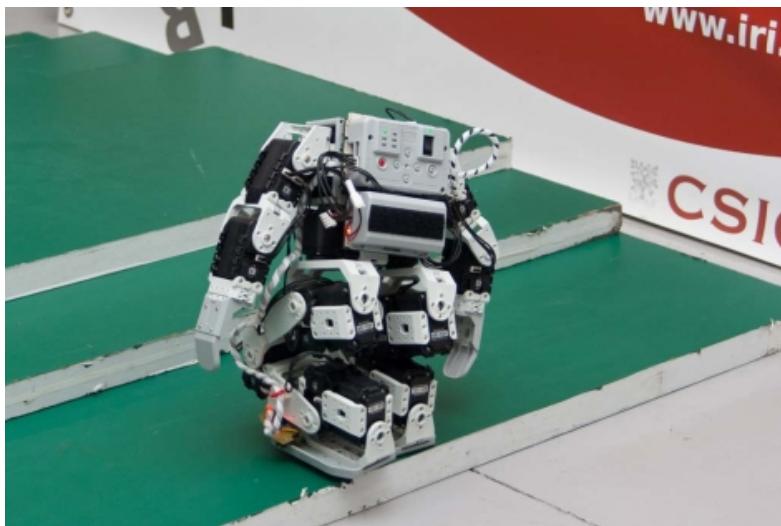


Figura 1.4: Prueba de la escalera

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.

### 1.3.3. Sumo

La prueba de sumo (figura 1.5) 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 1.5: Prueba de sumo

#### 1.3.4. 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  $45^\circ$ . En la parte superior de los obstáculos se colocará un rectángulo rojo con un código QR en su interior (figura 1.6). 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 superar la prueba.

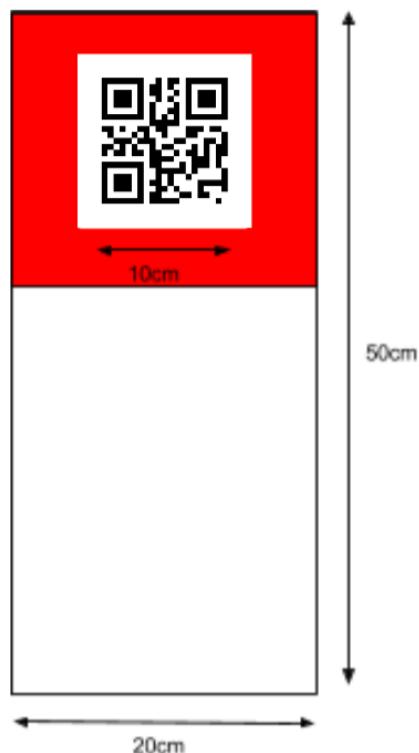


Figura 1.6: Marcador de la prueba de visión

#### 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 blablablabla



# **Capítulo 2**

## **Estado del arte**

### **2.1. Plataformas robóticas mini-humanoides**

Robots

### **2.2. Competiciones**

#### **2.2.1. RoboGames**

Robots estructuralmente molones multidisciplinares

#### **2.2.2. Robo-One**

Robots estructuralmente molones, fuertes, estables, radiocontrolados. Compiten cara a cara muchos equipos

#### **2.2.3. Robocup**

Enjambres, vision, proramación coordinada. Entornos dinámicamente VARIABLES.

#### **2.2.4. Ceabot**

Robots individuales, compiten solos en entornos "complejos" pero FIJOS

### **2.3. Locomoción**

hablar del motion, del pypose... contra un zmp y cosas asi

**2.4. Visión**

# **Capítulo 3**

## **Objetivos**

### **3.1. Desarrollar una plataforma robótica mini-humanoide**

El primer objetivo de este proyecto es el desarrollo de una plataforma robótica mini-humanoide de propósito general. Para llegar a ello será necesario estudiar los componentes que forman un robot humanoide. Sus capacidades deben ser al menos suficientes para desarrollar sobre la plataforma los objetivos que se listan a continuación. Además, se pretende que la plataforma sea lo suficientemente robusta y fácil de usar como para poder servir de base para proyectos futuros.

#### **3.1.1. Estudiar los componentes que necesita un robot humanoide**

Se realizará un estudio de qué elementos son necesarios para formar parte de un robot mini-humanoide. Entre ellos, se evaluará su funcionamiento en base a las necesidades del proyecto y se analizará qué componentes son adecuados para integrarse en el robot. En resumen, se diseñará un sistema completo y funcional de controlador, alimentación, sensores y actuadores.

#### **3.1.2. Integrar una cámara USB**

Otro objetivo será integrar una cámara con la que se realizarán algoritmos de visión. Para ello se requiere que la cámara quede integrada físicamente en el robot. Adicionalmente, la cámara deberá poder moverse de forma independiente al cuerpo del robot, por lo que se construirá una base móvil que otorgue a la cámara capacidad para rotar e inclinarse.

### 3.1.3. Integrar un controlador

Integrar en el robot un controlador que tenga el potencial necesario para controlar la locomoción del robot, comunicarse con los diversos sensores que se monten, y procesar algoritmos de visión. También se buscará que dicho controlador permita programarse en diferentes lenguajes sin tener que depender de un entorno de desarrollo fijo. Por último, el controlador que se escoja debe tener las capacidades de procesamiento suficientes para permitir un funcionamiento fluido y, al mismo tiempo, permitir una autonomía de funcionamiento del robot razonable.

En el caso de que el controlador requiera conexiones o conectores especiales, se deberán realizar las modificaciones necesarias para que el sistema sea limpio y fiable. Dicho de otro modo, si por ejemplo los servos necesitan conectores especiales que el controlador no posea, se buscará una solución, como por ejemplo, diseñar una placa de expansión.

### 3.1.4. Agregar sensores y actuadores

Otro objetivo será incluir sensores y actuadores para aumentar las capacidades tanto de sensorización como de locomoción del robot. Aunque el componente principal para tomar datos del entorno será la cámara, no estará de más incluir algunos sensores que apoyen y complementen la información recogida por la parte de visión. De este modo, obtendremos un sistema versátil que podrá seguir funcionando cuando el procesamiento de imágenes no sea suficiente, ya sea por un entorno complejo o con características inadecuadas para la visión (escenarios con poca luz, humo... etc).

Así mismo, también se incluirán actuadores adicionales que amplien las posibilidades del robot, ya sea para soportar el movimiento de nuevos componentes (como la cámara) o simplemente para aumentar las capacidades de locomoción.

### 3.1.5. Diseñar un sistema de alimentación

Dado que se prevee la inclusión de numerosos componentes nuevos, se deberá rediseñar el sistema de alimentación de manera que cumpla dos condiciones. Por una parte, debe ser capaz de alimentar con diferentes tensiones a todos los miembros del sistema, adaptándose a los requerimientos eléctricos de cada uno. Por otra parte, debe ofrecer una autonomía total al robot razonable y suficiente para realizar pruebas de competición.

También se procurará que las baterías sean fácilmente intercambiables, sin necesidad de montar y desmontar partes mecánicas. Este objetivo se propone a partir de la necesidad de hacer cambios veloces de baterías en las competiciones de mini-humanoides.

### 3.1.6. Realizar las modificaciones estructurales que sean pertinentes

Se tendrá como objetivo adecuar la estructura del robot para el montaje de nuevos dispositivos. Además, secundariamente se mejorarán diversos puntos de la estructura para mejorar las capacidades generales de la plataforma.

Las piezas deberán diseñarse siguiendo dos reglas. La primera es que puedan imprimirse en una impresora 3D casera. La segunda regla es que soporten los esfuerzos necesarios y no se rompan durante la operación del robot.

## 3.2. Puesta en marcha y programación

El robot debe diseñarse, montarse, programarse y testarse. Quizás éste sea el objetivo al que se la dará mayor peso en este proyecto, y no es otro que obtener al final una plataforma mini-humanoide fiable y probada para su uso. Este proyecto no debe quedar en un simple estudio teórico, sino que debe llevarse a la práctica y obtener resultados positivos.

### 3.2.1. Programar la locomoción bípeda

Se deberá diseñar un sistema de locomoción completo del robot. En este punto se unen diversos objetivos, desde controlar las articulaciones por separado hasta la realización de programas de caminata. Se deberá conseguir al menos la realización de los siguientes movimientos:

- Avance frontal.
- Rotación a la izquierda.
- Rotación a la derecha.
- Paso lateral a la izquierda.
- Paso lateral a la derecha.
- Reincorporación tras una caída frontal

**■ Reincorporación tras una caída trasera**

Junto a esto, se deberá crear una librería en C++ que permita realizar movimientos de una forma cómoda y sencilla.

**3.2.2. Programar sensores**

Los sensores que se seleccionen deberán adecuarse para poder operar adecuadamente con la placa de control que se utilice. No solo deberán adecuarse electrónicamente hablando, sino que también deberán interpretarse su datos de forma correcta para extraer información útil durante el funcionamiento del robot.

**3.2.3. Desarrollar algoritmos de visión**

El comportamiento del robot se basará en la información tomada por la cámara. Para procesar esta información se utilizarán las librerías de OpenCV, por lo que será necesario familiarizarse con ellas y estudiar las diversas posibilidades que ofrecen en el campo de la visión por computador. Con ello, se desarrollarán programas adaptados a la plataforma robótica del proyecto. Éstos, deberán optimizarse de forma que funcionen correctamente con las capacidades de procesamiento del controlador.

**3.2.4. Desarrollar aplicaciones de competición**

Se prevee la presentación del robot a competiciones. Esto significa que como parte del proyecto se desarrollarán aplicaciones de competición de carácter diverso. Para la realización de los programas, se tendrá en cuenta principalmente el reglamento concreto de cada prueba, con sus objetivos y limitaciones.

Estas aplicaciones se apoyarán en todo lo desarrollado anteriormente, y constituirán la mejor prueba experimental del funcionamiento del robot.

## Capítulo 4

# 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.

### 4.1. Plataforma robótica

El primer paso para la realización de este proyecto fue el estudio y selección de las plataformas robóticas que se encuentran en el mercado 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. Buscamos una plataforma que cumpla los siguientes requisitos:

- **Programable en C/C++**

Se requiere que sea programable en C/C++ y que además no dependa de una IDE concreta.

- **Posibilidad de conectarle una cámara**

Se necesita que permita conectarle una cámara y realizar programas con OpenCV.

- **Expandible con sensores y actuadores**

El sistema debe permitir la adición de nuevos sensores y actuadores, sin verse limitado por hardware o software.

- **Servos de al menos 12kg/cm**

Ya que se van a incluir nuevas partes, es absolutamente necesario que los servos puedan soportar carga adicional colocada en el robot.

#### ■ Chasis reconfigurable

Aunque no es tan importante como el resto, es posible que el robot requiera modificaciones, por lo que una base reconfigurable y versátil será apreciada.

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

#### 4.1.1. 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 más accesibles y extendidos.

##### Hitec Robonova

El Robonova (figura 4.1) es uno de los mini-humanoides más 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 4.1: Hitec Robonova ©Hitec RCD

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 comunicarselas 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 4.2), del fabricante japonés Kondo, es uno de los mini-humanoides más avanzados actualmente. Puede presumir de ser el modelo de serie más utilizado en el campeonato RoboOne, siendo seleccionado por los equipos por su gran agilidad y tamaño compacto.



Figura 4.2: 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 analógicas 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 gra-

tuitamente 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 servicios que montan, manteniendo comunes el resto de partes del robot. El modelo Standard (figura 4.3) 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 4.3: 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 4.4) incluye 18 servos Dynamixel, modelo AX-12A o AX-18A, dependiendo de la versión del kit. Los actuadores 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.



Figura 4.4: Bioloid Premium

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.

- **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

Realizando un primer análisis, ninguno de los robots candidatos cumple los requisitos. Todos ellos obligan a utilizar entornos de desarrollo y lenguajes propios para su programación. El Robonova y el Robovie tienen unos servos demasiado débiles, lo que dificultaría mucho añadir mas peso al robot. Entre el Kondo y el Bioloid, se ha elegido el Bioloid por tres razones: Es mas fácil de modificar, el kit contiene mas servos y es más barato. También, previendo las modificaciones futuras, se contempló la idea de comprar únicamente los servos Dynamixel que usa el Bioloid por separado. Sin embargo, resultó mas económico adquirir el kit completo de Bioloid Comprehensive.

## 4.2. Modificaciones estructurales

El Bioloid Comprehensive es un buen punto de partida, sin embargo tiene algunos puntos débiles que conviene revisar. Además, para poder implantar en el robot los dispositivos que requiere este proyecto se necesitarán mejorar las capacidades de la plataforma.

### 4.2.1. Cabeza móvil TODO IMAGEN

Un requisito importante del proyecto es permitir que la cámara que vamos a montar pueda moverse con libertad para enfocar a diferentes zonas de su entorno. Dado que en CEABOT la mayoría de los datos que aporta el entorno están situados en el suelo, necesitamos que la cámara al menos pueda dirigirse hacia el frente y hacia el suelo. Esto lo conseguiremos con la adición de un microservo PWM y una plataforma articulada para la cabeza. Dadas las características de este movimiento, no necesitamos un servo con grandes capacidades, ya que su rango de acción estará muy limitado y su efecto será despreciable en el reparto de pesos del robot.

Figura 4.5: Microservo PWM

Se ha elegido un microservo PWM por su bajo tamaño y peso, su bajo precio y su sencillez a la hora de montarlo y programarlo. El principio de funcionamiento de un servo PWM es muy simple. De las tres patillas de su conector, dos son de alimentación y la tercera se encarga de recibir una señal PWM que, variando la amplitud de su pulso, ordena al servo a moverse a una posición fijada.

Particularmente, se ha escogido un servo Tower Pro MG90s (figura 4.5) cuyas especificaciones presentan un torque de  $2.4=\text{kg}/\text{cm}$ , y una transmisión metálica soportada por rodamientos. Este último dato es muy importante si tenemos en cuenta que el sistema de cabeza móvil se situará en una zona extrema del robot, y que un golpe provocado por una caída forzará de forma directa el eje del servo de la articulación. Este servo proporcionará a la cabeza la robustez necesaria para salir indemne en este tipo de accidentes. Accidentes que por otra parte son muy comunes teniendo en cuenta que el robot estará destinado a la realización de pruebas de competición.

### 4.2.2. Cintura móvil TODO IMAGEN

Otra de las modificaciones básicas a realizar sobre la paltaforma robótica, ha sido la inclusión de un servo adicional Dynamixel AX-12A para articular la cintura. A parte de la mejora de capacidades que se produce en los movimientos del robot, servirá para girar la dirección de la cámara radialmente. Podría decirse que entre el servo de la cabeza y el de la cintura, se ha creado un sistema distribuido de pan-tilt que permitirá mover la cámara en todas las direcciones manteniendo fija la base del robot, es decir, sus piernas.

### 4.3. Sensorización TODO

El funcionamiento del robot y su control va a estar basado principalmente en la cámara y los algoritmos de visión que se programarán. No obstante, la cámara consume muchos recursos del procesador y existen algunas tareas sencillas que es más fácil programar con sensores más simples. A continuación se muestra un estudio de sensores que pueden ser útiles en el proyecto.

#### 4.3.1. Sensores de distancia

Existen diferentes tipos de sensores de distancia. El propósito de estos sensores no es otro que la medición de longitudes utilizando diferentes principios físicos. Se ha planteado el uso de sensores infrarrojos y/o de sensores de ultrasonidos.

##### Sensores de infrarrojos

Los sensores de infrarrojos basan su funcionamiento en la reflexión de luz infrarroja sobre superficies. El sensor está formado por dos LEDs infrarrojos, uno emisor y otro receptor. El emisor emite de forma continua una luz infrarroja dirigida hacia un punto fijo. Si en ese punto fijo se encuentra un objeto físico, la luz se reflejará y será captada por el diodo receptor. La salida de estos sensores se mide mediante la diferencia de potencial que se produce en el diodo receptor. Se ha estudiado la inclusión en el proyecto de sensores infrarrojos Sharp, especialmente de su modelo GP2Y0A21YK, que tiene un rango de operación de entre 4 y 150cm.

##### Sensores de ultrasonidos

Los sensores de ultrasonidos detectan distancias basándose en el tiempo en el que un sonido de alta frecuencia recorre el espacio. Normalmente, los sensores de ultrasonidos tienen dos focos, uno emisor y otro receptor. Una de las diferencias de este tipo de sensores con los sensores infrarrojos es que mientras los sensores infrarrojos toman medidas de un punto, el rango de operación de los sensores de ultrasonidos se abre en un cono de 60° de amplitud desde su emisor. Se ha analizado el sensor de ultrasonidos HC-SR04 como posible candidato para formar parte del robot por su fácil accesibilidad. Otro punto importante es que en estos sensores la salida es proporcional a la distancia medida, es decir, se adecúa a una recta. Es por esto por lo que podremos realizar medidas reales directamente midiendo la salida del sensor.

### Experimentos de medida

Para poner en práctica y observar las diferencias teóricas que existen entre el funcionamiento de ambos sensores, se ha realizado un pequeño experimento en el que se han realizado medidas de longitud entre el sensor y un obstáculo. Para ello, se han analizado tres configuraciones posibles. En cada imagen, se ha marcado en azul el rango de acción del sensor. La linea roja representa la distancia que medirá el sensor en cada situación.

En el primer experimento (figura 4.6) se ha colocado un obstáculo como los que se utilizan en la prueba de navegación de CEABOT en posición frontal y recta. Puede observarse que en esta situación ambos sensores medirían de forma similar la distancia entre el sensor y el obstáculo.

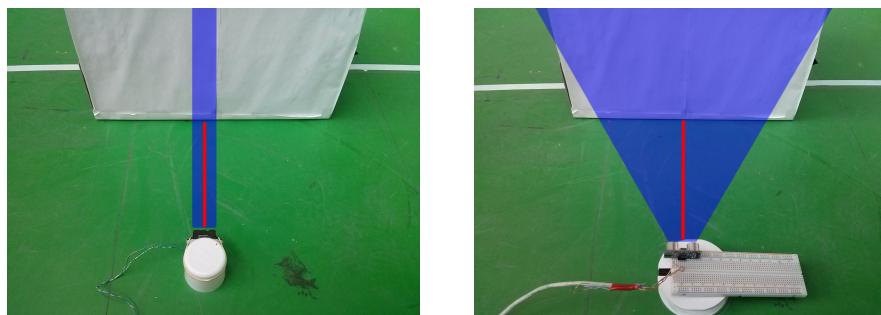


Figura 4.6: Primer experimento.

En el segundo experimento (figura 4.7) se ha desplazado el objeto ligeramente hacia la derecha. Puede observarse cómo el rango de acción del sensor infrarrojo, al tener una forma lineal, no detecta el obstáculo. Mientras tanto, el sensor de ultrasonidos es capaz de detectarlo, ya que en esa posición sigue estando dentro de su rango de acción.

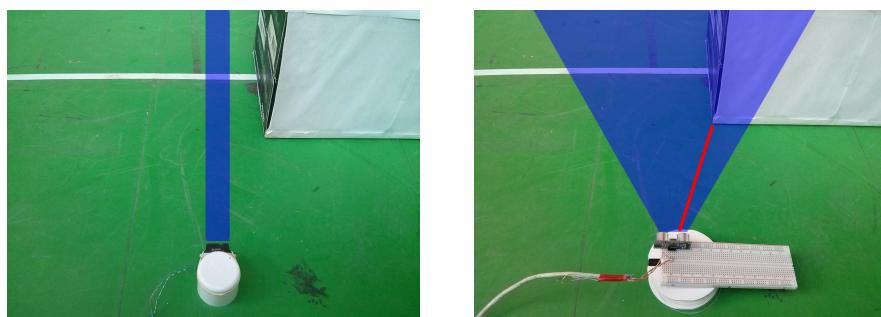


Figura 4.7: Segundo experimento.

Por último, el tercer experimento (figura 4.8) muestra el comportamiento de ambos sensores al enfrentarse a una superficie oblicua. Tal y como puede observarse en la imagen, el sensor infrarrojo realiza una medida puntual a la zona del obstáculo que se interpone en su rango de acción. Por otra parte, el sensor de ultrasonidos nos devuelve la distancia del punto del objeto que, encontrándose dentro de su rango, constituye la distancia mínima entre ambos.

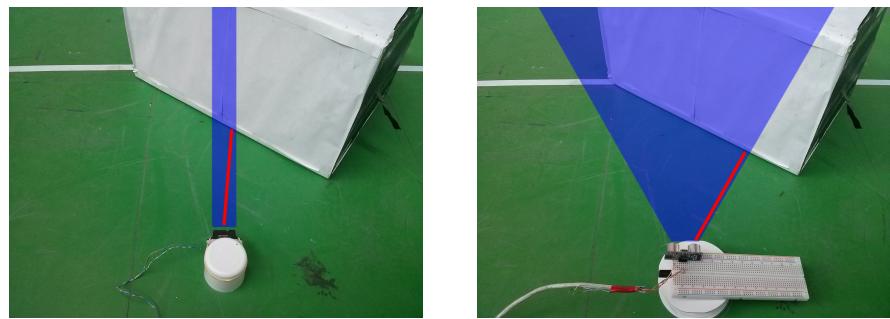


Figura 4.8: Tercer experimento.

De este experimento pueden sacarse varias conclusiones. La primera de ellas es que no existe un sensor mejor que el otro, cada uno destaca en una funcionalidad. Un sensor infrarrojo nos dará una mejor precisión a la hora de realizar medidas de distancia en un punto concreto, sin embargo, su rango de acción es limitado y será importante controlar donde está apuntando exactamente en cada momento. El sensor de ultrasonidos por su parte, detectará un obstáculo con mayor probabilidad, pero siempre existirá un incertidumbre respecto al posicionamiento exacto del obstáculo que estamos midiendo. De esta forma, un sensor infrarrojo será útil cuando necesitemos precisión en la posición del objeto y por este motivo, será el seleccionado para montarse en el robot. De todos modos, no se descarta la idea de montar también sensores de ultrasonidos en un futuro.

#### 4.3.2. Sensores iniciales TODO

El control del equilibrio del robot es muy importante en configuraciones bípedas, ya que estos robots son conocidos por su facilidad para tropezar y caer al suelo. Con el objetivo de analizar la posición del robot respecto al suelo se ha requerido la inclusión de diferentes sensores iniciales.

### Acelerómetro y giroscopio

En el mercado existen varios modelos económicos. En este proyecto se ha utilizado un sensor MPU9150 (figura 4.10), que es un sensor inercial compuesto de acelerómetro de 3 ejes, giroscopio de 3 ejes y brújula de 3 ejes. Este sensor combina dos sensores, un MPU6050 (incluye el acelerómetro y el giroscopio) y un AK8975 (incluye la brújula). Sin embargo, en la práctica se observó que la brújula incluida en el sensor era muy propensa a sufrir interferencias, por lo que se decidió montar una brújula adicional que no tuviese estos problemas.



Figura 4.9: Sensor inercial MPU9150

### Brújula TODO IMAGEN

La inclusión de una brújula es indispensable cuando se requiere conocer la dirección en la que se desplaza un robot. En este proyecto se montará una brújula para evaluar si el desplazamiento del robot se produce de forma recta y efectuar los redireccionamientos necesarios. Existen diferentes modelos de brújulas digitales en el mercado. Todas ellas, basan su funcionamiento en la detección y medición de campos magnéticos. Para lograr conocer la orientación del robot, la brújula deberá apoyarse en su posición respecto a la del campo magnético terrestre. El gran problema de estos dispositivos es su facilidad para modificar sus medidas cuando existe un campo magnético fuerte en su entorno. Esta interferences pueden ser causadas por aparatos electrónicos, estructuras metálicas o incluso por los propios componentes del robot.

Para este proyecto se ha utilizado una brújula CMPS03 (figura ??). El motivo de haber seleccionado este modelo frente a otros más comunes, como por ejemplo la HMC5883L, se debe a que ya ha sido utilizada en anteriores proyectos robóticos en la asociación con



Figura 4.10: Brújula magnética CMPS03

muy buenos resultados. Esto se debe en parte a que la placa del sensor incluye un PIC que se encarga de calibrar y comunicar los datos medidos, de forma que no será necesario que el controlador del robot realice estos cálculos y podrá utilizar directamente la medida extraída.

#### 4.3.3. Cámara TODO

La elección de la cámara ha sido condicionada principalmente por su compatibilidad con el driver de Linux v4l2. Existe una amplia variedad de cámaras válidas en el mercado. Dado el carácter de este proyecto, se ha optado por utilizar una webcam, ya que son mas accesibles que las cámaras avanzadas de investigación y los algoritmos de visión que serán programados no requieren imágenes de alta calidad.



Figura 4.11: Microsoft LifeCam Cinema

El modelo seleccionado ha sido una Microsoft LifeCam Cinema (figura 4.11). Entre sus características destacan su posibilidad de grabar video en 720p HD, enfoque automático y la regulación de brillos en tiempo real. Adicionalmente, se trata de una cámara de fácil desmontaje y dimensiones contenidas, por lo que será sencillo integrarla en el robot.

Cabe remarcar que antes de utilizar esta cámara se realizaron pruebas con una Hama Pocket, pero dada su baja calidad de construcción presentó diversos problemas de fiabilidad y fue despreciada en favor de la Microsoft.

#### 4.4. Elección del controlador TODO

Llegamos a un punto crítico del proyecto, y es la elección de un controlador para nuestro sistema. La controlador CM-5 contenida en el kit original de Bioloid Comprehensive se encuentra muy lejos de poder soportar el conjunto de nuevos dispositivos que se usarán en el proyecto. Necesitaremos reemplazarla por un sistema mas complejo que nos permita conectar y programar los sensores seleccionados, programar algoritmos de visión y comunicarse con los servos Dynamixel.

Para solucionar esto se decide colocar un SBC (Single Board Computer, ordenador de placa única), que dadas sus capacidades de procesamiento se utilizará como controlador principal. Así mismo, será el encargado de realizar el procesamiento de imágenes. Junto al SBC, se conectará un controlador mas sencillo basado en un microcontrolador para controlar los Dynamixel AX-12A por separado.

De esta forma, el controlador del robot estará formado por dos

elementos, el SBC que se encargará del procesamiento de imágenes y sensores, y un microcontrolador se encargará de la locomoción del robot.

#### 4.4.1. Controlador de locomoción TODO

El controlador de locomoción debe encargarse de mover los 20 servos del robot y de comunicarse con el controlador de visión para recibir instrucciones. A continuación se realiza un estudio de posibles placas que podrían utilizarse.

##### Arduino TODO

Las placas Arduino se han hecho famosas por su fácil programación y puesta en marcha. Dada su popularidad, existe una gran comunidad de usuarios que generan documentación de forma constante. Con una placa Arduino disponemos de una amplia colección de bibliotecas para interactuar con sensores y actuadores. Sin embargo, estas placas por si solas no soportan la comunicación con los servos Dynamixel. Por esta razón, se ha desestimado su uso en el proyecto.

##### Arbotix

La placa Arbotix (figura 4.12) está basada, al igual que las placas Arduino, en un microcontrolador AVR. No obstante, esta placa de desarrollo está orientada principalmente a la construcción de pequeños robots, por lo que posee capacidades mejores que las de las placas Arduino. Entre sus especificaciones destacan: 2 puertos serie (uno de ellos reservado para el bus Dynamixel), 32 pines de entrada/salida, 8 entradas analógicas, 2 drivers de 1A para motores de continua y zócalos para la conexión de módulos XBEE.



Figura 4.12: Placa Arbotix

El problema de estas placas es que no existe un distribuidor que las venda en Europa. Eso no solo significa que sean difíciles de conseguir, sino tambien que no tienen demasiados usuarios, y por tanto la documentación es escasa y heterogenea.

#### Serie OpenCM TODO

Recientemente la empresa Robotis ha sacado a la venta una nueva linea de placas de desarrollo, las placas OpenCM (figura 4.13). Estas placas de desarrollo están basadas en las placas Arduino, copiando sus funcionalidades y añadiendo al sistema la posibilidad de comunicarse con actuadores Dynamixel. Su entorno de programación está basado en Arduino IDE y continene bibliotecas similares a las de Arduino para la comunicación con sensores y actuadores. Su alta compatibilidad con Arduino proporciona al usuario una mayor facilidad a la hora de buscar documentación sobre cómo utilizar la placa.

Si bien es cierto, es necesario comentar que la primera edición, la CM900, presentó problemas en su diseño que la convirtieron en una placa muy poco fiable y propensa a deteriorarse prematuramente. En este proyecto se utilizará la OpenCM9.04, sucesora de la CM900, con similares capacidades y tamaño reducido. Entre sus especificaciones destacan: Un microcontrolador ARM Cortex-M3, 26 pines de entrada/salida, 10 entradas analogicas de 12 bits y 3 puertos serie (uno de ellos reservado para el bus Dynamixel).



Figura 4.13: Placa CM900

#### 4.4.2. Controlador de visión TODO

Como controlador principal, se utilizará un SBC ( TODO revisar las siglas ) . Por tamaño y capacidades existen dos alternativas directas: La Raspberry Pi B y la BeagleBone Black.

##### Raspberry Pi TODO

La Raspberry Pi es un ordenador completo del tamaño de una tarjeta de crédito, dirigida a formar parte de proyectos de electrónica, informática y robótica. Dado su bajo coste y fácil adquisición, la Raspberry Pi cuenta con una comunidad de usuarios muy extensa repartida a lo largo del mundo. En internet existen colecciones de tutoriales educativos para formar al usuario y dar soporte a sus proyectos. En el cuadro 4.1 se describen las especificaciones técnicas de la placa.



Figura 4.14: Placa OpenCM9.04

Especificaciones Rasp	
CPU	ARM 1176JZFS 700 MHz
Memoria (SDRAM)	512 MB (compartidos con la GPU)
Puertos USB 2.0	2 (vía hub USB integrado)
Almacenamiento integrado	SD / MMC / ranura para SDIO
Periféricos de bajo nivel	8 x GPIO, SPI, I2C, UART
Consumo energético	700 mA (3.5 W)
Fuente de alimentación	5 V vía Micro USB o GPIO header
Dimensiones:	85.60mm × 53.98mm
Sistemas operativos soportados:	GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux. RISC OS2

Cuadro 4.1: Especificaciones Raspberry Pi B

En el caso que nos ocupa, sería una perfecta candidata a ocuparse del procesamiento de imágenes del robot, sin embargo hay algunos detalles que es importante conocer. El primer problema de

la Raspberry Pi es que la lógica de sus pines trabaja a 5V, mientras que la placa que hemos seleccionado para la parte de locomoción, la OpenCM, funciona a 3.3V. Esto significa que para comunicar ambas placas sería necesario incluir un convertidor lógico en el sistema. Por otra parte, sus GPIOs tienen un funcionamiento puramente binario, sin entradas analógicas ni generación de señales PWM. Por último, como puede observarse en la figura ?? aunque tiene un tamaño razonable, la placa cuenta con varios conectores de video, audio, conexión en red... etc que aunque podrían resultar útiles en otro tipo de proyectos, en nuestro caso estorbarán a la hora de integrar el dispositivo en la espalda del robot.

[FOTO REAL DE UNA RASPBERRY] TODO

#### **BeagleBone Black TODO**

La BeagleBone Black (figura 4.15), de Texas Instruments no es tan popular como la Raspberry Pi, sin embargo posee algunas características muy interesantes.



Figura 4.15: SBC BeagleBone Black

Como puede observarse en el cuadro 4.2, sus características son ligeramente superiores a las de la Raspberry. Principalmente, el hecho de que tenga tal variedad de pines la hace muy adecuada para usarse como controlador principal de un robot. La BeagleBone Black actualmente no cuenta con una comunidad tan extensa como la Raspberry Pi, pero cada vez se ve mas en proyectos. El problema fundamental de la BeagleBone Black es la falta de bibliotecas en

C/C++ que permitan programar sus periféricos de bajo nivel. Esto significa que en el caso de utilizar esta placa, se deberán programar bibliotecas propias para estos propósitos.

Especificaciones BBB	
CPU	AM335x 1GHz ARM Cortex-A8
Memoria (SDRAM)	512 MiB
Puertos USB 2.0	1x Standard A y 1x mini B
Almacenamiento integrado	4GB 8-bit eMMC / microSD
Periféricos de bajo nivel	4xUART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2x CAN bus
Consumo energético	1200 mA (6 W)
Fuente de alimentación	5 V vía Micro USB, jack de 5.5mm a 5 V o GPIO header
Dimensiones:	86.40 mm × 53.30 mm
Sistemas operativos soportados:	Fedora, Android, Ubuntu, Debian, openSUSE, Ångström, FreeBSD, NetBSD, OpenBSD, QNX, MINIX 3, RISC OS, y Windows Embedded.

Cuadro 4.2: Especificaciones BeagleBone Black

Por otro lado, su lógica funciona a 3.3V y es mas compacta que la Raspberry Pi. Por todo lo comentado, y aunque trabajar con ella pueda resultar mas tedioso que con la Raspberry Pi, se ha seleccionado la Beaglebone Black como controlador principal del robot.

#### 4.4.3. Diseño del sistema TODO

Una vez se ha seleccionado todos los elementos que formarán parte del robot, se ha diseñado el conexionado que llevarán. Tal y como se ha ido viendo a lo largo de este apartado, por un lado tendremos la OpenCM que se encargará de controlar el servo PWM y los servos Dynamixel. La BeagleBone en cambio, se encargará de recibir los datos de todos los sensores y de la cámara. Las dos placas se conectarán en serie. El diagrama de la figura 4.16 representa

las conexiones de todos los dispositivos, a excepción de la parte de alimentación, que se verá en la próxima sección.

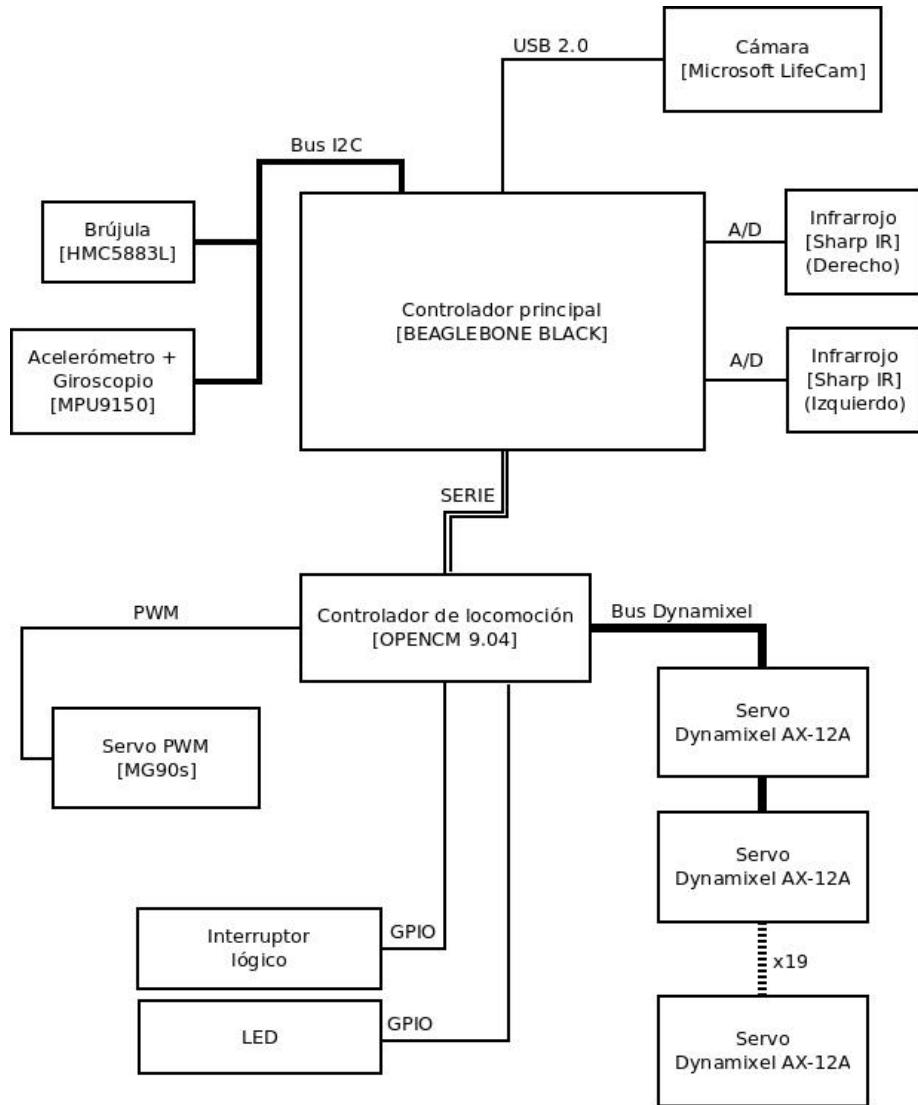


Figura 4.16: Diagrama de conexiones

TODO Revisar que los componentes sean los buenos ( especialmente TODO la brujula )

## 4.5. Alimentación

La batería contenida en el kit es una batería NiMH de 8 elementos AA Sanyo Eneloop que desarrolla 9.6V y 1900mAh. Para el consumo

de este robot es una opción aceptable que proporciona cerca de 10 minutos de autonomía. No está mal si se tiene en cuenta que pruebas con el CEABOT tiene una duración máxima limitada a 5 minutos. No obstante, tiene un inconveniente, su peso y su volumen es muy alto. Al estar situada en la espalda, el centro de gravedad del robot sube en altura de forma considerable. Esto causa inestabilidades y complica bastante la programación de desplazamientos.

#### 4.5.1. Batería

Por las razones descritas, se ha decidido buscar una alternativa. Los requisitos para seleccionar una batería los marcan tres factores: La tensión de funcionamiento de los servos, la de las placas de control y la de los sensores. Pero dado que los servos tienen una tensión de operación en un rango de 9V a 12V, respecto al resto de elementos que serán alimentados a 5V o 3.3V, nos centraremos principalmente en encontrar una batería adecuada para los servos.

Hoy en dia, las baterías ne NiMH están empezando a caer en desuso a favor de la baterías de litio. Se barajaron dos posibilidades.

##### Baterías de polímero de litio

Las baterías de polímero de litio, mas conocidas como baterías LiPo, destacan por desarrollar una tasa de descarga bastante alta y por tener un tamaño y peso reducido. Dado su uso en automodelismo y aeromodelismo, son unas baterías muy comunes que en los últimos años han ido bajando su precio. Actualmente existe una gran variedad de baterías LiPo de diversos tamaños y precio asequible. Cada célula LiPo ofrece 3.7V, por lo que la elección correcta para este robot sería una batería de 3S, o lo que es lo mismo, 11,1V.

Sin embargo, las baterías LiPo tienen una desventaja importante, su fragilidad. El proceso de carga y descarga de una batería de este tipo no es un asunto trivial, una sobredescarga de la batería provoca un deterioro inmediato de la misma, calentando sus elementos e incluso pudiendo llegar a incendiarse. Es por ello que se necesita un cargador con capacidad para balancear las células, de forma que las tensiones de cada una de ellas se tengan bajo control durante el proceso.

##### Baterías de litio fosfato de hierro

Las batería de litio fosfato de hierro, es decir, LiFe, son conocidas por su gran resistencia y larga vida útil. Cada célula de LiFe desa-

rrolla 3.3V y no presenta problemas por sobredescargas. De hecho, estas baterías ni siquiera requieren de un cargador balanceador para su carga, sino que pueden cargarse sin problemas con un cargador normal. El gran problema de estas baterías es que sus elementos son bastante voluminosos, y, al no ser tan populares como las LiPo, tampoco es demasiado fácil encontrar modelos de tamaño y capacidad adecuados en el mercado.

#### Elección y diseño del sistema

Por su tamaño y prestaciones se ha elegido una batería LiPo Rhino de 3S y 1750mAh. Esta batería tiene unas dimensiones perfectas para montarse en la cintura del robot y su peso es 50 ( TODO pesarla, que ahora no la tengo a mano ) gramos menor al de la batería original (215 gramos).

A continuación se presenta una tabla (cuadro 4.3) con una estimación del consumo medio de todos sus elementos.



Figura 4.17: Batería LiPo

Dispositivo	Número	Consumo unitario	Consumo total
Dynamixel AX-12A	x19	$11,1V \cdot 600mA = 6660mW$	$126540mW$
TowerPro MG90s	x1	$5V \cdot 120mA = 600mW$	$600mW$
BeagleBone Black	x1	$5V \cdot 1200mA = 6000mW$	$6000mW$
OpenCM 9.04	x1	$11,1V \cdot 90mA \simeq 1000mW$	$1000mW$
Sharp IR	x2	$5V \cdot 30mA = 150mW$	$300mW$
MPU9150	x1	$3,3V \cdot 15mA \simeq 50mW$	$50mW$
HMC5883L	x1	$3,3V \cdot 12mA \simeq 40mW$	$40mW$
Microsoft LifeCam	x1	$5V \cdot 200mA = 1000mW$	$1000mW$
Total			$135530mW \simeq 136W$

Cuadro 4.3: Consumo medio

Por tanto, la estimación de autonomía del robot en funcionamiento es la siguiente:

$$\frac{11,1V \cdot 1750mAh}{135530mW} \cdot \frac{60min}{h} \simeq 8,6min$$

Entre 8 y 9 minutos de autonomía teóricos es una buena cifra contando con el tamaño del robot y sus prestaciones. Como conclusión podemos remarcar el hecho de que gracias al nuevo sistema de alimentación hemos podido introducir los dispositivos necesarios para cumplir los requerimientos de este proyecto sin comprometer la autonomía del robot.

#### 4.5.2. Regulador de tensión

Hasta aquí parece que es suficiente con la elección de la nueva batería, sin embargo, aún falta un detalle que deberemos resolver. La mayor parte de los dispositivos que forman el conjunto están alimentados a 5V. La batería que hemos elegido tiene una tensión

de 11.1V, y se ha observado, que cuando está cargada al máximo desarrolla hasta 12.6V. No podemos alimentar la BeagleBone Black y el resto de componentes a una tensión tan alta, por lo que será necesario colocar un regulador que nos proporcione 5V.

### Regulador lineal

La primera idea fue utilizar un pequeño circuito con un regulador lineal LM7805. El LM7805 es un componente muy común en los circuitos electrónicos, ya que ofrecen una salida de 5V a partir de tensiones de entrada de hasta 35V. En principio no parecería muy descabellado conectar los servos AX-12A y la placa OpenCM directamente a la batería, y conectar el resto de componentes a través del regulador.

No obstante existen dos problemas para realizar este montaje. El primero es que la eficiencia de este componente es muy pobre, cercana al 60 % según su hoja de características. Esto provocaría una disminución drástica en el tiempo de operación del robot y una disipación de energía tan alta que sería necesario incluir un disipador. Pero el segundo problema es más importante, el componente está diseñado para aportar una corriente máxima al circuito de 1 amperio. Teniendo en cuenta los datos del cuadro 4.3, solo la BeagleBone Black ya requiere una corriente de 1.2A, y junto al resto de componentes, asciende hasta 1.7A . Un consumo superior al marcado por el fabricante puede producir fallos de funcionamiento en el componente. En este caso particular, podría producirse un calentamiento excesivo, que a su vez puede llegar a provocar cortocircuitos internos entre la patilla de entrada y la de salida. Las consecuencias de un fallo así serían nefastas, ya que deteriorarían los dispositivos conectados a él.

### Convertidor UBEC TODO

Un UBEC (figura 4.18) es un convertidor DC-DC de tipo reductor. Los UBEC se utilizan normalmente con baterías de litio cuando el sistema necesita una tensión de alimentación de 5V o 6V. Es por ello que se utilizan mucho en receptores de emisoras de radiocontrol, y gracias a esto existe una amplia variedad de UBECs asequibles en el mercado. La mayor ventaja de estos circuitos convertidores es su eficiencia superior al 90 %. Gracias a esto nuestro sistema tendrá unas perdidas menores, no se calentará en exceso y gozará de una autonomía superior.



TODo

Figura 4.18: Convertidor DC-DC comercial, UBEC

#### Elección final TODO

Tal y como se comentaba anteriormente, nuestro sistema tiene un consumo estimado de unos 1.7A en la salida de 5V, por lo que un UBEC de 3A sería suficiente para proporcionar al circuito la corriente necesaria para su funcionamiento. Adicionalmente se han incluido unos condensadores en la entrada y salida del circuito para ... como se verá en el apartado ( TODO capítulo 7 )



## **Capítulo 5**

# **Descripción de las herramientas a utilizar**

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

### **5.1. 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.

#### **5.1.1. 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 5.1). 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

## 42 CAPÍTULO 5. 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.

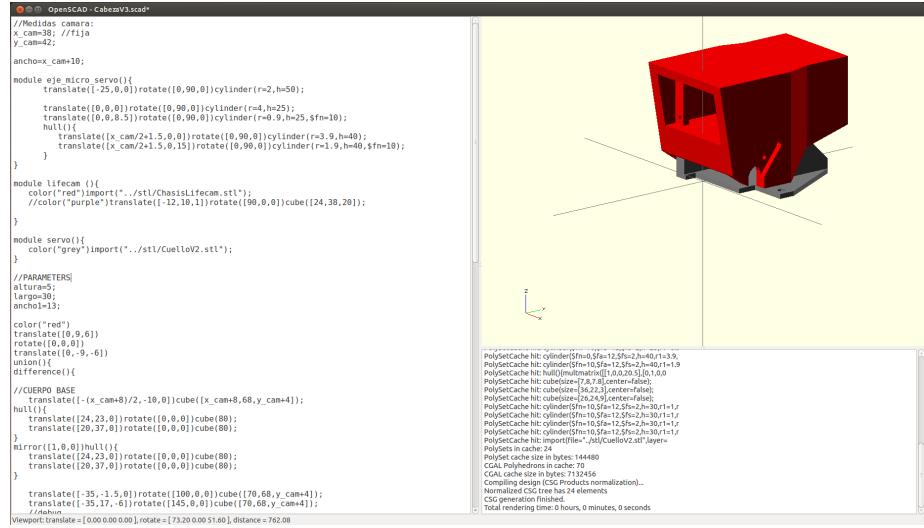


Figura 5.1: Pantalla del editor de OpenSCAD

### 5.1.2. 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 5.2).

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.

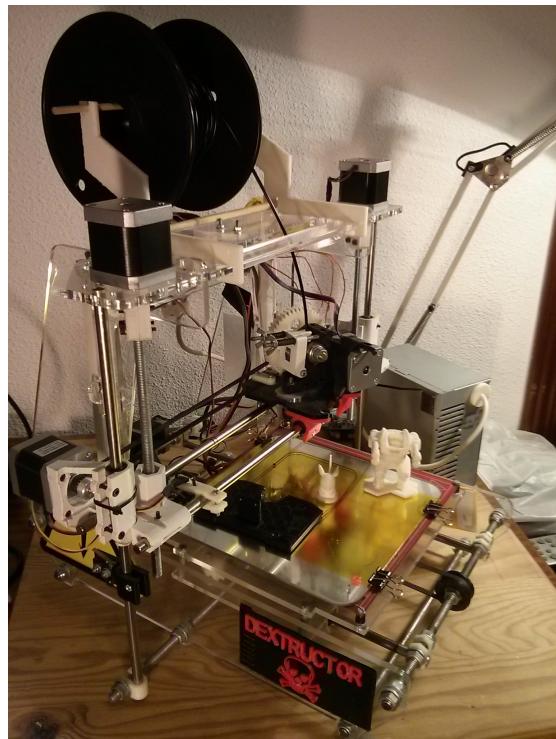


Figura 5.2: Impresora 3D Prusa i2 Air

## 5.2. 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.

### 5.2.1. 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

## 44 CAPÍTULO 5. DESCRIPCIÓN DE LAS HERRAMIENTAS A UTILIZAR

simulación 3D de la placa. El entorno (figura 5.3) 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.

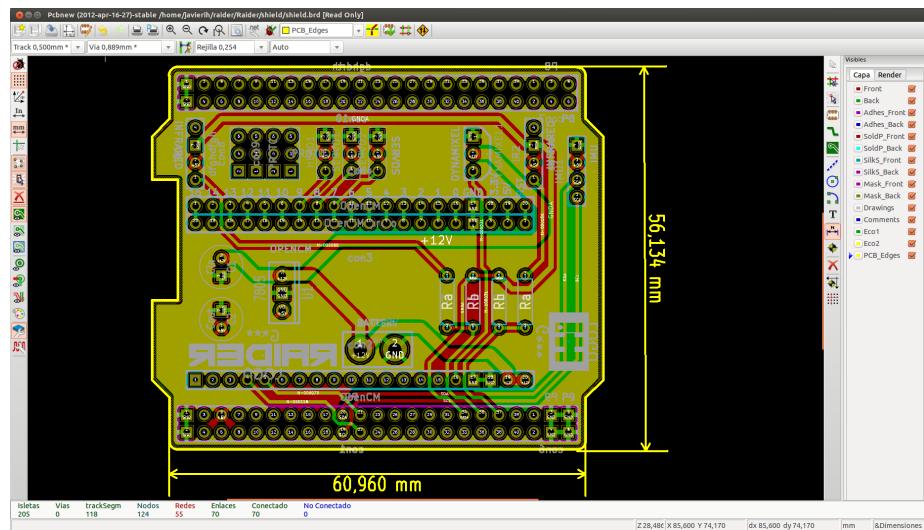


Figura 5.3: KiCad

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.

### 5.3. 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.

#### 5.3.1. 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.

### 5.3.2. Qt Creator

Qt Creator (figura 5.4) 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.

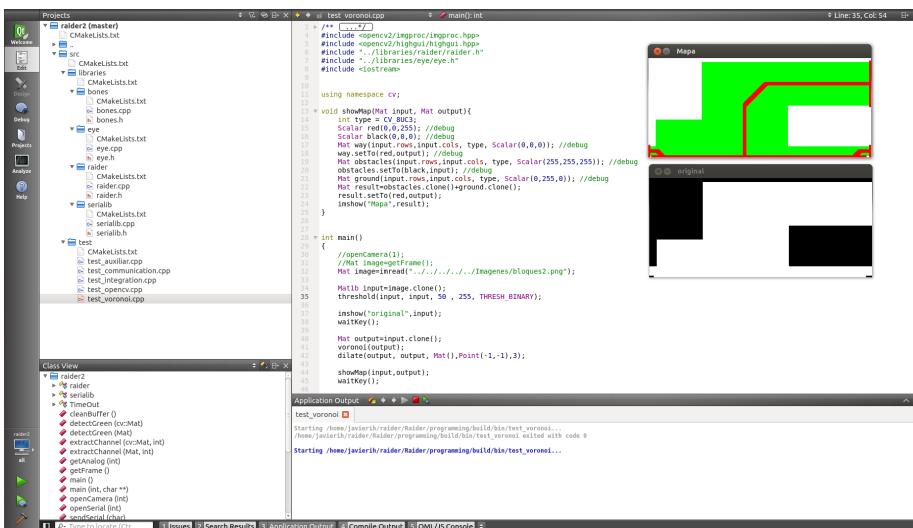


Figura 5.4: Qt Creator

### 5.3.3. 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.

### 5.3.4. 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 5.5) 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.

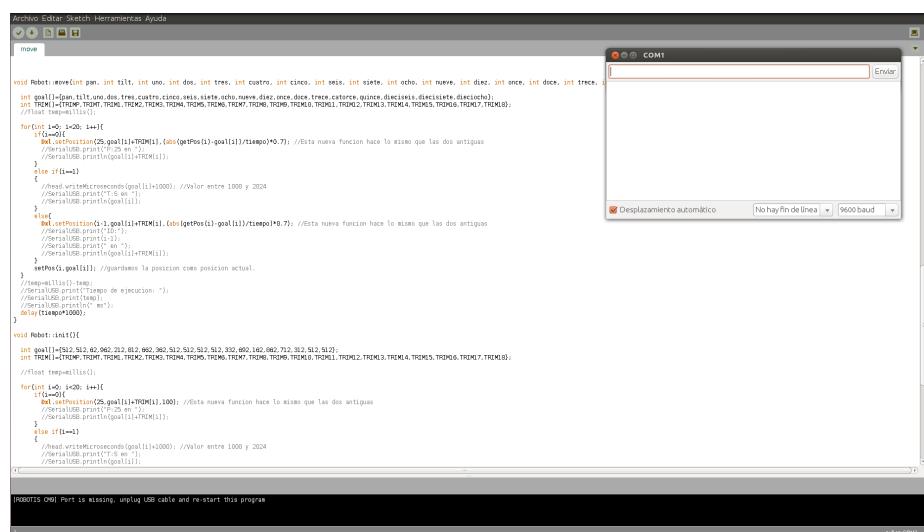


Figura 5.5: CM9 IDE

### 5.3.5. 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?)



## **Capítulo 6**

# **Diseño de las partes mecánicas TODO**

La mayor parte de las piezas que monta Raider han sido rediseñadas para mejorar su función y/o permitir el alojamiento de nuevos dispositivos. Además, al haber sido diseñadas con OpenSCAD, ha sido posible parametrizarlas para facilitar modificaciones futuras. Acciones como agregar a una pieza un soporte para un sensor determinado son ahora mucho mas rápidas.

### **6.1. Cabeza**

Uno de los principales objetivos del proyecto, el tomar datos del entorno con una cámara, requiere el diseño y montaje se un soporte móvil en el que poder albergar la webcam. Se ha decidido colocar la webcam en la cabeza del robot por ser la zona mas alta del robot y por tener espacio para su libre movimiento. La cabeza estará formada por cuatro piezas: Bancada del servo, chasis, tapa con ranura para el cable y soporte de la cámara.

### **6.2. Cintura**

Como se comentó en el apartado TODO , ha sido necesario incluir un servo adicional en la cintura que permita realizar movimientos horizontales del tronco del robot respecto a la piernas. Consta de dos piezas que aprisionan el servomotor y otra mas que se utilizará como soporte de la batería.

### 6.3. Tronco

Para diseñar el tronco existen varias premisas. Necesitamos un cuerpo que permita albergar el nuevo controlador, soportar los servos de los hombros, tener una zona adecuada para montar la cabeza y una base firme que permita atornillarlo al nuevo servo de la cintura.

### 6.4. Brazos

En este apartado se han diseñado diferentes tramos tramos del brazo. Con ello se ha dotarle de una total modularidad, separando uniones entre servos, soportes para sensores y protectores. Gracias a esto, es muy sencillo sustituir unas piezas por otras. En este caso se han incluído sensores infrarrojos en los antebrazos, sin embargo, si se necesitase cambiarlos por sensores de ultrasonidos (o cualquier otro sensor) solo sería necesario modificar y reimprimir la bancada del sensor.

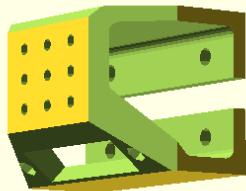
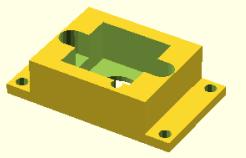
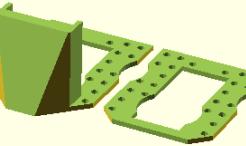
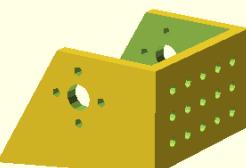
### 6.5. Piernas

En cuanto a las piernas, se han modificado con dos objetivos: Rigidificar sus tramos y acortar levemente su longitud para descargar peso en las articulaciones.

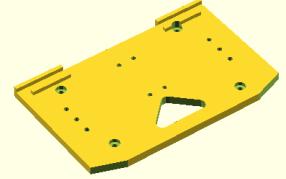
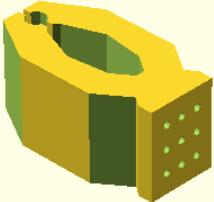
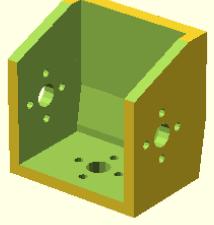
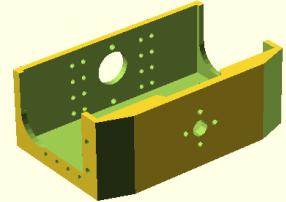
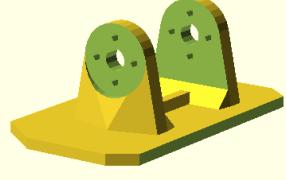
### 6.6. Pies y tobillos

El diseño de los pies se ha planteado de forma que cumpla las especificaciones del reglamento del campeonato CEABOT y al mismo tiempo tenga la mayor superficie posible. A priori, parece sencillo suponer que cuanto mayor sea la extensión del pie mayor será la estabilidad del robot, sin embargo, también es muy importante controlar como se distribuye el peso en la planta. Con el objetivo de centrar el peso en el centro del pie, se ha realizado un rediseño completo del tobillo.

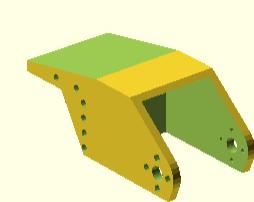
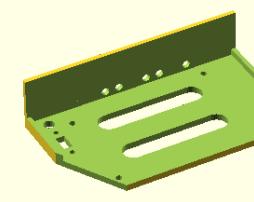
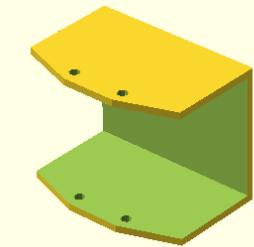
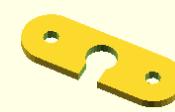
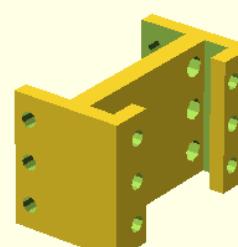
### 6.7. Lista de piezas

Imagen	Archivo	Cantidad	Observaciones
	antebrazo.stl	x1	
	cabeza.stl	x1	
	chasislifecam.stl	x1	
	cintura.stl	x1	
	codo.stl	x1	
	cuello.stl	x1	

Cuadro 6.1: piezas

Imagen	Archivo	Cantidad	Observaciones
	espalda.stl	x1	
	gripper.stl	x1	
	hombro.stl	x1	
	pecho.stl	x1	
	pie.stl	x1	
	pierna1.stl	x1	

Cuadro 6.2: piezas

Imagen	Archivo	Cantidad	Observaciones
	pierna2.stl	x1	
	protector.stl	x1	
	soporteBateria.stl	x1	
	tapaCabeza.stl	x1	
	tobillo.stl	x1	

Cuadro 6.3: piezas



## Capítulo 7

# Desarrollo TODO MONTAJES

En este capítulo se mostrarán los procedimientos que se han seguido para montar algunos componentes. Hasta este punto hemos definido una colección de sensores, actuadores y piezas; el siguiente paso será integrar y conectarlos todos en el robot.

### 7.1. Adecuación de sensores

Los sensores utilizados, tal y como se detalló en la selección de componentes, serán: un acelerómetro y giroscopio MPU9150, una brújula CMPS03 y dos infrarrojos Sharp. A continuación se muestra cómo han sido conectados al controlador principal, la BeagleBone Black.

#### 7.1.1. Infrarrojos Sharp

Los sensores infrarrojos modelo GP2Y0A21YK0F de la marca Sharp, permiten variar la tensión de alimentación entre -0.3 y 7V, pero experimentalmente se ha observado que la mayor precisión se alcanza a 5V. A esta tensión, podemos observar en el datasheet del sensor una gráfica (figura 7.1) en la que se muestra el valor de la salida del sensor respecto a la distancia medida.

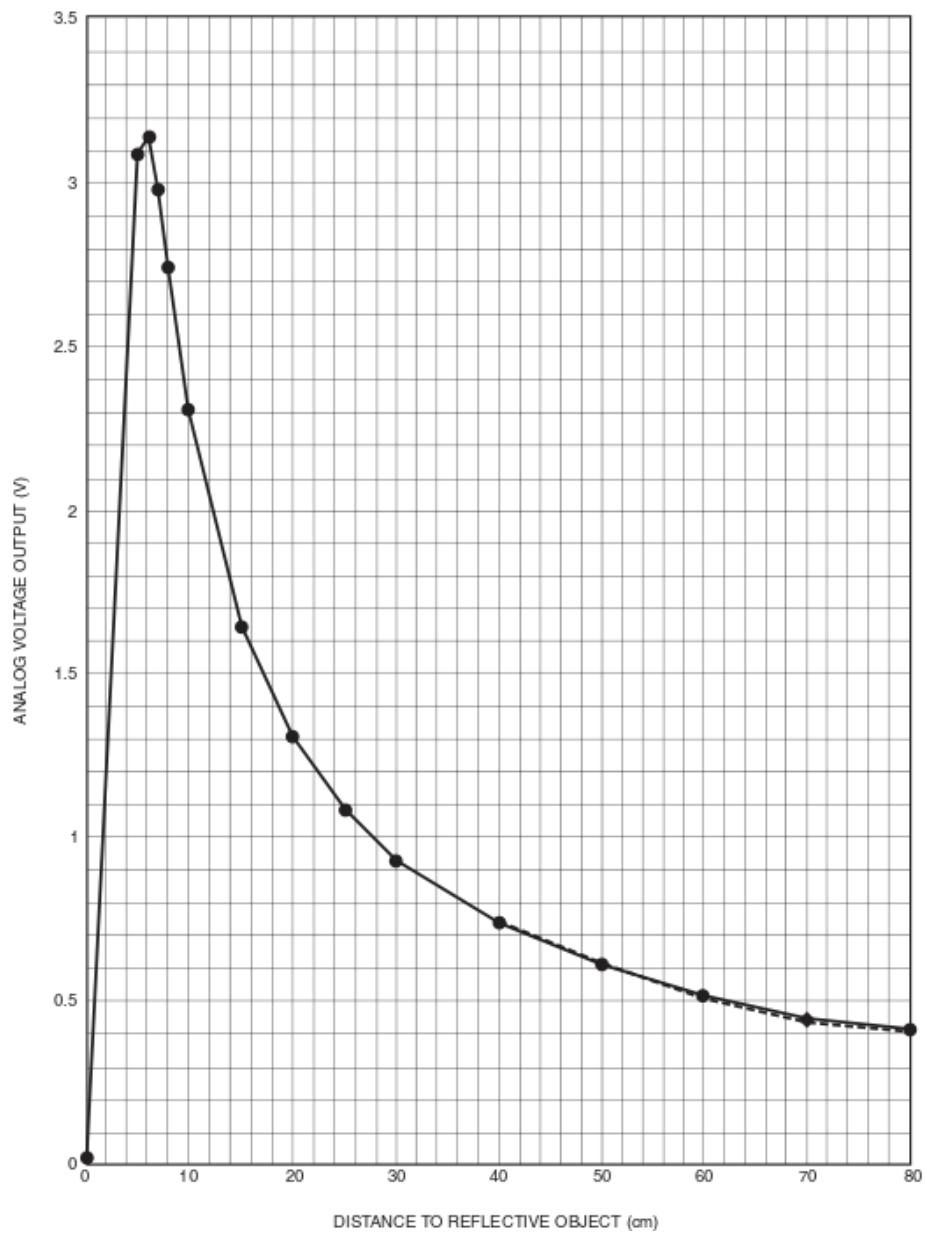


Figura 7.1: Gráfica de tensión entre salida para un sensor infrarrojo

De esta gráfica pueden sacarse dos conclusiones. La primera, ya conocida, es que la variación de la salida del sensor y su medida no son proporcionales, por lo que si fuese necesaria extraer medidas en unidades reales habría que calcular una adecuación compleja. Sin embargo, esto no será necesaria ya que la programación del sensor se basará en umbrales fijos que serán marcados de forma empírica. La segunda conclusión es que el sensor ofrecerá una tensión de salida

máxima de 3.3V.

Dado que la BeagleBone Black dispone de hasta siete entradas analógicas (figura 7.2), conectaremos las salidas de los dos sensores en dos de estos pines. Sin embargo, leyendo el manual de la BeagleBone Black ( TODO manual ) observamos que cada entrada analógica soporta un máximo de 1.8V. Si alimentando el sensor a 5V lo conectásemos directamente, un pico de 3.3V podría deteriorar la placa. Por ello, se ha decidido aplicar una simple adecuación de la salida añadiendo un divisor resistivo. De esta forma, nos cercioraremos de que los 3.3V máximos que podrían salir del sensor se conviertan en 1.8V.

## 7 analog inputs (1.8V)

P9			P8		
DGND	1	2	DGND	1	2
VDD_3V3	3	4	VDD_3V3	3	4
VDD_5V	5	6	VDD_5V	5	6
SYS_5V	7	8	SYS_5V	7	8
PWR_BUT	9	10	SYS_RESETN	9	10
GPIO_30	11	12	GPIO_60	11	12
GPIO_31	13	14	GPIO_40	13	14
GPIO_48	15	16	GPIO_51	15	16
GPIO_4	17	18	GPIO_5	17	18
I2C2_SCL	19	20	I2C2_SDA	19	20
GPIO_3	21	22	GPIO_2	21	22
GPIO_49	23	24	GPIO_15	23	24
GPIO_117	25	26	GPIO_14	25	26
GPIO_125	27	28	GPIO_123	27	28
GPIO_121	29	30	GPIO_122	29	30
GPIO_120	31	32	VDD_ADC	31	32
AIN4	33	34	GNDA_ADC	33	34
AIN6	35	36	AIN5	35	36
AIN2	37	38	AIN3	37	38
AIN0	39	40	AIN1	39	40
GPIO_20	41	42	GPIO_7	41	42
DGND	43	44	DGND	43	44
DGND	45	46	DGND	45	46

*Make sure you don't input more than 1.8V to the analog input pins.*

This is a single 12-bit analog-to-digital converter with 8 channels, 7 of which are made available on the headers.

Figura 7.2: Esquema de entradas analógica de una BeagleBone Black

En el esquema de la figura 7.3 podemos observar las conexiones que se han realizado entre la placa y el sensor. El sensor es alimentado a 5V a partir del convertidor DC-DC y puesto a tierra común con el controlador. La salida del sensor se aplicará sobre el divisor resistivo, siendo la salida del divisor resistivo la entrada a la BeagleBone Black. Se ha colocado una resistencia de  $33k\Omega$  en  $R_a$  y una resistencia de  $66k\Omega$  en  $R_b$ .

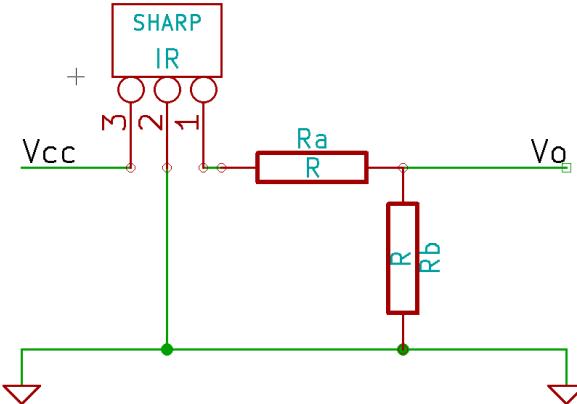


Figura 7.3: Circuito de adecuación de un sensor infrarrojo

### 7.1.2. Brújula y sensor inercial

Tanto la b?ujula CMPS03 como el sensor inercia MPU9150 han sido conectados por I2C, ambos en el mismo bus. La BeagleBone Black cuenta con dos buses I2C independientes (7.4).

## 2 I2C ports

P9			P8		
DGND	1	2	DGND	1	2
VDD_3V3	3	4	VDD_3V3	GPIO_38	3
VDD_5V	5	6	VDD_5V	GPIO_34	5
SYS_5V	7	8	SYS_5V	GPIO_66	7
PWR_BUT	9	10	SYS_RESETN	GPIO_69	9
GPIO_30	11	12	GPIO_60	GPIO_45	11
GPIO_31	13	14	GPIO_40	GPIO_23	13
GPIO_48	15	16	GPIO_51	GPIO_47	15
I2C1_SCL	17	18	I2C1_SDA	GPIO_27	17
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19
I2C2_SCL	21	22	I2C2_SDA	GPIO_62	21
GPIO_49	23	24	I2C1_SCL	GPIO_36	23
GPIO_117	25	26	I2C1_SDA	GPIO_32	25
GPIO_125	27	28	GPIO_123	GPIO_86	27
GPIO_121	29	30	GPIO_122	GPIO_87	29
GPIO_120	31	32	VDD_ADC	GPIO_10	31
AIN4	33	34	GNDA_ADC	GPIO_9	33
AIN6	35	36	AIN5	GPIO_8	35
AIN2	37	38	AIN3	GPIO_78	37
AIN0	39	40	AIN1	GPIO_76	39
GPIO_20	41	42	GPIO_7	GPIO_74	41
DGND	43	44	DGND	GPIO_72	43
DGND	45	46	DGND	GPIO_70	45

Figura 7.4: Esquema de puertos I2C de una BeagleBone Black

De este modo el sistema ha quedado diseñado tal y como se muestra en la figura 7.5. Cabe destacar que los dos sensores se han alimen-

tado a 3.3V, ya que el MPU9150 los requiere y el CMPS03, aunque se recomienda conectarlo a 5V, funciona perfectamente a 3.3V. Esta tensión se ha conseguido desde el pin P9-3 del controlador.

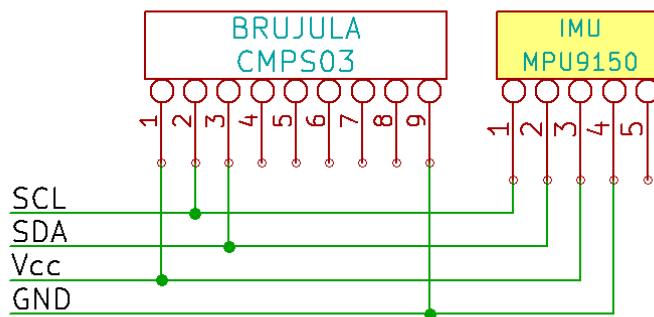


Figura 7.5: Circuito del bus I2C

## 7.2. Desarrollo de una placa de expansión TODO

Para montar el esquema que se diseñó anteriormente en la figura 4.16 será necesario cablear un gran número de conexiones entre dispositivos. Para ello, se ha optado por diseñar y fabricar una placa de expansión que permita conectar todos los dispositivos de una forma limpia y ordenada. Esta placa servirá para comunicar y alimentar las dos placas controladoras, sensores, actuadores y alimentación de todos ellos.

( TODO referencia a un anexo, foto o algo de los planos )

## 7.3. Montaje

Una vez preparados todo los componentes se ha procedido a su montaje. En los siguientes dos apartados se muestra cómo se ha montado la placa de expansión y cómo se ha integrado la webcam en la cabeza del robot.

### 7.3.1. Montaje de la placa de expansión

//Aquí meter hasta los separadores

**7.3.2. Integración de la cámara en la cabeza**

//fotos del montaje y esas cosas

# **Capítulo 8**

# **Programación TODO**

En este capítulo se presenta el desarrollo de todo el software que ha sido requerido preparar para la puesta en marcha del robot.

## **8.1. Configuración de la BeagleBone Black**

El controlador principal, la BeagleBone Black, trae de fábrica una instalación de una distribución Ångström basada en Linux. El sistema operativo incluye: entorno gráfico, escritorio, OpenCV 2.2 y otros programas como navegadores o reproductores de archivos multimedia. Esto se debe a que la BeagleBone Black no es una placa que de fábrica esté pensada para su uso en robots, sino que constituye un ordenador completo que puede adquirir la función de centro multimedia, equipo de ofimática o incluso servidor de una red.

### **8.1.1. Sistema operativo**

La distribución Ångström no es una de las más populares entre los sistemas operativos basados en Linux, no cuenta con una comunidad de usuarios tan amplia como Debian o Fedora. Por ello, las actualizaciones no suelen ser muy frecuentes y, en muchos casos, el sistema presenta errores de funcionamiento a la hora de realizar tareas básicas, como la instalación de nuevo software.

Por estas razones se ha decidido instalar un nuevo sistema operativo. Se pretende conseguir una distribución de Linux que solo tenga lo mínimo necesario que requiere el robot para su funcionamiento. Con este objetivo, se instala en la BeagleBone Black una distribución Debian.

### 8.1.2. Instalación de librerías

Para la realización del proyecto, se ha requerido instalar Git, CMake, Video4Linux2 y las librerías de OpenCV 2.4.8 y ZBar 0.1.

Antes de comenzar con la instalación de software, es recomendable realizar una actualización del sistema ejecutando los siguientes comandos:

```
sudo apt-get update
sudo apt-get upgrade
```

Lo primero que instalaremos será Git, ya que lo necesitaremos para instalar algunas librerías posteriormente. Git se instala ejecutando el siguiente comando:

```
sudo apt-get install git-core
```

Para instalar CMake lo haremos de la siguiente forma:

```
sudo apt-get install cmake
sudo apt-get install cmake-curses-gui
```

Para el control y configuración de la webcam se necesitará el driver Video4Linux. Se instalará como se muestra a continuación:

```
sudo apt-get install v4l-utils
```

Una vez hecho esto, se pasará a instalar las librerías que se utilizarán en el código del robot. La forma mas sencilla de instalar Zbar será hacerlo desde los repositorios de Debian. De este modo, ejecutaremos los siguientes comandos:

```
sudo apt-get install libzbar0
sudo apt-get install libzbar-dev
```

Por último, instalaremos OpenCV. Se comenzará por descargar todas las librerías que requiere OpenCV. Se instalarán una a una de la siguiente forma:

```
sudo apt-get install build-essential
sudo apt-get install libgtk2.0-dev
sudo apt-get install pkg-config
sudo apt-get install python-dev
```

```
sudo apt-get install python-numpy
sudo apt-get install libavcodec-dev
sudo apt-get install libavformat-dev
sudo apt-get install libswscale-dev
```

Hecho esto, se procederá a descargar la última versión estable del código, que en este momento es la 2.4.9.

```
wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.9/ope
```

Y posteriormente se instalará de este modo:

```
unzip opencv-2.4.9.zip
cd opencv-2.4.9
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
make
sudo make install
```

### 8.1.3. Configuración de la cámara

A continuación se configurará la webcam en la beaglebone. Para esto se conectará por USB la Microsoft Lifecam a la BeagleBone Black tal y como puede observarse en la figura ???. A partir de aquí, se iniciará sesión en la BeagleBone Black y se procederá a configurar la cámara.

Lo primero que se hará será comprobar que la cámara se ha reconocido correctamente. Para ello ejecutaremos el comando:

```
lsusb
```

Y deberá proporcionarnos una salida parecida a la que puede observarse en la figura 8.1. La salida indica que la webcam está conectada en el bus 001

```
debian@arm:~$ lsusb
Bus 001 Device 002: ID 045e:075d Microsoft Corp. LifeCam Cinema
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Figura 8.1: Salida del comando lsusb

Para cerciorarnos de que la cámara se ha reconocido correctamente podemos buscarla en el directorio /dev. En la siguiente imagen

```
debian@arm:~$ ls /dev/video0
/dev/video0
```

Figura 8.2: Comprobación de conexión para la webcam

(figura 8.2) se observa el archivo `/dev/video0`, que está vinculado a la webcam.

A continuación nos ayudaremos de las utilidades de Video4Linux (una interfaz para la programación de aplicaciones con captura de imágenes para Linux) para la webcam. OpenCV se apoya en Video4Linux, por tanto, lo primero que se hará será comprobar que la cámara es compatible con Video4Linux. Para ello, se ejecutará el siguiente comando:

```
v4l2-ctl --list-devices
```

Tras ejecutarlo se deberá observar una salida similar a la de la figura 8.3

```
debian@arm:~$ v4l2-ctl --list-devices
Microsoft® LifeCam Cinema(TM) (usb-musb-hdrc.1.auto-1):
    /dev/video0
```

Figura 8.3: Comprobación de compatibilidad para la webcam

Ésto nos confirma que la cámara funciona correctamente. Para proseguir con la configuración vamos a ejecutar el comando siguiente:

```
v4l2-ctl --all
```

En la figura 8.4 podemos observar la salida del comando. En ella se pueden observar diferentes parámetros como el formato de video, la resolución o los fotogramas por segundo.

Por último, se bajará la resolución de la cámara a un valor que nos permita tratar las imágenes con mayor rapidez cuando se programen algoritmos de visión con OpenCV desde la BeagleBone Black. Se ha elegido una resolución de 320x240, mas adelante se ajustará este valor empíricamente en función del rendimiento del robot. Para ello ejecutamos este comando de configuración:

```
v4l2-ctl --set-fmt-video=width=320,heigth=240
```

Tras estos pasos la cámara ya está preparada y lista para usarse.

```
debian@arm:~$ v4l2-ctl --all
Driver Info (not using libv4l2):
    Driver name      : uvcvideo
    Card type       : Microsoft® LifeCam Cinema(TM)
    Bus info        : usb-musb-hdrc.1.auto-1
    Driver version: 3.8.13
    Capabilities   : 0x84000001
                      Video Capture
                      Streaming
Format Video Capture:
    Width/Height   : 640/480
    Pixel Format  : 'YUYV'
    Field          : None
    Bytes per Line: 1280
    Size Image    : 768000
    Colorspace     : SRGB
Crop Capability Video Capture:
    Bounds         : Left 0, Top 0, Width 640, Height 480
    Default        : Left 0, Top 0, Width 640, Height 480
    Pixel Aspect: 1/1
Video input : 0 (Camera 1: ok)
Streaming Parameters Video Capture:
    Capabilities   : timeperframe
    Frames per second: 312500.000 (312500/1)
    Read buffers   : 0
Priority: 2
```

Figura 8.4: Parámetros leídos de la cámara

#### 8.1.4. Configuración de pines

Para finalizar esta sección, se muestra el desarrollo de un script para configurar los pines del controlador.

La Beaglebone Black posee 92 pines configurables de entrada y salida. Entre ellos, 7 pueden funcionar como entradas analógicas y 5 parejas de pines están preparadas como puertos serie (figura 8.5).

## 4.5 serial UARTs

P9			P8			
DGND	1	2	DGND	1	2	
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4
VDD_5V	5	6	VDD_5V	GPIO_34	5	6
SYS_5V	7	8	SYS_5V	GPIO_66	7	8
PWR_BUT	9	10	SYS_RESETN	GPIO_69	9	10
UART4_RXD	11	12	GPIO_60	GPIO_45	11	12
UART4_TXD	13	14	GPIO_40	GPIO_23	13	14
GPIO_48	15	16	GPIO_51	GPIO_47	15	16
GPIO_4	17	18	GPIO_5	GPIO_27	17	18
UART1_RTSN	19	20	UART1_CTSN	GPIO_22	19	20
UART2_RXD	21	22	UART2_RXD	GPIO_62	21	22
GPIO_49	23	24	UART1_RXD	GPIO_36	23	24
GPIO_117	25	26	UART1_RXD	GPIO_32	25	26
GPIO_125	27	28	GPIO_123	GPIO_86	27	28
GPIO_121	29	30	GPIO_122	GPIO_87	29	30
GPIO_120	31	32	VDD_ADC	UART5_CTSN+	31	32
AIN4	33	34	GNDA_ADC	UART4_RTSN	33	34
AIN6	35	36	AIN5	UART4_CTSN	35	36
AIN2	37	38	AIN3	UARR5_RXD+	37	38
AIN0	39	40	AIN1	GPIO_76	39	40
GPIO_20	41	42	UART3_RXD	GPIO_74	41	42
DGND	43	44	DGND	GPIO_72	43	44
DGND	45	46	DGND	GPIO_70	45	46

There is a dedicated header for getting to the UART0 pins and connecting a debug cable. Five additional serial ports are brought to the expansion headers, but one of them only has a single direction brought to the headers.

Figura 8.5: Puertos serie en una BeagleBone Black

Para configurar los pines de la placa se utilizarán overlays (capas). Un overlay es un archivo que indica al sistema operativo cómo debe configurarse un pin. La BeagleBone Black se configura mediante Device Tree Overlays, un mecanismo de descripción del hardware que forma parte de un sistema. Para configurar las entradas analógicas necesarias para los sensores infrarrojos, y el puerto serie que comunicará la placa con el controlador de locomoción; se utilizarán los archivos *.dtbo* que pueden encontrarse en */lib/firmware* (figura 8.6)

```
debian@arm:/lib/firmware$ ls *.dtbo
am33xx_pwm-00A0.dtbo          BB-BONE-PRU-01-00A0.dtbo      BB-UART1-00A0.dtbo
BB-ADC-00A0.dtbo               BB-BONE-PRU-02-00A0.dtbo      BB-UART2-00A0.dtbo
BB-BONE-AUDI-01-00A0.dtbo       BB-BONE-PRU-03-00A0.dtbo      BB-UART2-RTSCTS-00A0.dtbo
BB-BONE-BACON-00A0.dtbo        BB-BONE-PRU-04-00A0.dtbo      BB-UART4-00A0.dtbo
BB-BONE-BACONE-00A0.dtbo       BB-BONE-PWMT-00A0.dtbo       BB-UART4-RTSCTS-00A0.dtbo
BB-BONE-BACONE2-00A0.dtbo      BB-BONE-RS232-00A0.dtbo      BB-UART5-00A0.dtbo
BB-BONE-CAM3-01-00A2.dtbo       BB-BONE-RST-00A0.dtbo       bone_pwm_P8_13-00A0.dtbo
BB-BONE-CAM-VVDN-00A0.dtbo     BB-BONE-RST2-00A0.dtbo      bone_pwm_P8_19-00A0.dtbo
BB-BONE-eMMC1-01-00A0.dtbo     BB-BONE-RTC-00A0.dtbo       bone_pwm_P8_34-00A0.dtbo
BB-BONE-GPEVT-00A0.dtbo        BB-BONE-SERL-01-00A1.dtbo    bone_pwm_P8_36-00A0.dtbo
BB-BONE-GPS-00A0.dtbo          BB-BONE-SERL-03-00A1.dtbo    bone_pwm_P8_45-00A0.dtbo
BB-BONE-LCD4-01-00A0.dtbo      BB-GPIOHELP-00A0.dtbo       bone_pwm_P8_46-00A0.dtbo
BB-BONE-LCD4-01-00A1.dtbo      BB-I2C1-00A0.dtbo         bone_pwm_P9_14-00A0.dtbo
BB-BONE-LCD7-01-00A2.dtbo      BB-I2C1A1-00A0.dtbo       bone_pwm_P9_16-00A0.dtbo
BB-BONE-LCD7-01-00A3.dtbo      BB-SPIDEVO-00A0.dtbo       bone_pwm_P9_21-00A0.dtbo
BB-BONE-LCD7-01-00A4.dtbo      BB-SPIDEV1-00A0.dtbo       bone_pwm_P9_22-00A0.dtbo
BB-BONELT-BT-00A0.dtbo         BB-SPIDEV1A1-00A0.dtbo     bone_pwm_P9_28-00A0.dtbo
```

Figura 8.6: Algunas capas de Device Tree Overlays

En concreto se utilizarán dos capas, la que activará el puerto serie número 2: *ttyO2\_armhf.com* y esta otra que nos permitirá activar las entradas analógicas: *cape-bone-iio*. Se habilitarán de la siguiente manera (figura 8.7):

```
debian@arm:~$ sudo su
root@arm:/home/debian# echo ttyO2_armhf.com > /sys/devices/bone_capemgr.9/slots
root@arm:/home/debian# echo cape-bone-iio > /sys/devices/bone_capemgr.9/slots
```

Figura 8.7: Habilitación de capas

Puede realizarse una comprobación simple de que los pines se han configurado correctamente realizando una lectura analógica. Para ello, se leerá el archivo */sys/bus/platform/drivers/bone-iio-helper/helper.15/AIN4*. En la figura 8.8 se ha realizado una lectura del pin 4, el valor de 570 indica que se está realizando una lectura de  $570mV$ .

```
debian@arm:~$ cat /sys/bus/platform/drivers/bone-iio-helper/helper.15/AIN4
570
```

Figura 8.8: Lectura analógica del pin 4

## 8.2. Sistema de locomoción TODO

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.

### 8.2.1. Movimiento del servo PWM

El control del servo PWM de Raider, situado en su cabeza, se realiza con la biblioteca Servo.h orginalmente desarrolladas para Arduino y que posteriormente ha sido portada para su utilización en OpenCM. Dado el cometido 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(uint8 pin)**

Con esta función configuramos un pin de la OpenCM para que actúe como emisora de señales PWM. A partir de esta inicialización ya puede moverse el servo.

#### **Servo::writeMicroseconds(uint16 pulseWidth)**

Para mover el servo se utiliza la función writeMicroseconds, cuyo parámetro define la amplitud del pulso de la onda PWM. Si bien es cierto, existe otra función paralela llamada write(int angle) que directamente utiliza como parámetro la posición en grados, se ha utilizado writeMicroseconds por su mayor precisión. Acepta valores de entre 1000 y 2000 microsegundos, pero para mantener un formato constante con el resto de articulaciones, se ha realizado una conversión matemática a un rango de 0 a 1023.

#### **8.2.2. 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 transmision y recepción de paquetes de forma simultanea. Esto la convierte en una conexión bastante típica en los sistema que utilizan un solo bus de comunicación. Como en el mismo bus existe mas 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 envio mientras manda un paquete. (TODO revisar) Los Dynamixel AX-12A poseen una tabla de registros (tabla TODO ) sobre la cual podemos modificar 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 instantanea, 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  $3,5rad/s$  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  $114rpm$ . Por tanto, se realizaría la siguiente conversión:

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

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 8.9

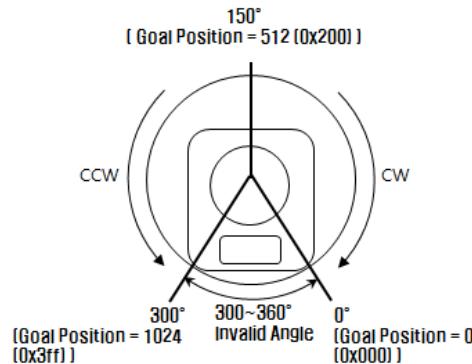


Figura 8.9: Amplitud de giro de un AX-12A

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:

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

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

### 8.2.3. 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 mas 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 raider TODO 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 targetPostion[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 volcar 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 más 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 más sencillo de las inercias entre movimientos consecutivos.

### 8.2.4. Funciones de movimientos combinados TODO

Llegado este punto se ha 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 instantánea del robot.

**void movVertical(int,int) TODO**

Meter fotos del robot con una pierna levantada

**void movLateral(int,int) TODO**

**void movFrontal(int,int) TODO**

### 8.2.5. Creación de movimientos completos TODO

Encontrándonos en este punto, la programación de desplazamientos, giros y otros movimientos complejos, se ha realizado mediante la combinación de las funciones anteriormente descritas. A cada movimiento se le asignará un valor hexadecimal en forma de carácter, de forma que los caracteres comunicados a la OpenCM sirvan como identificador de el movimiento que el controlador de visión está ordenando.

Se ha creado la función void controller(char), dentro de la clase Robot, con el objetivo de indexar todos los movimientos que realizará Raider. En el cuadro 8.1 se presenta una tabla con los movimientos programados y su comando asignado.

Comando	Función	Descripción
W	walk(3)	Caminar 3 pasos cortos
A	turnL()	Rotación a la izquierda
D	turnR()	Rotación a la derecha
S	run(3)	Caminar 3 pasos largos y rápidos
Q	stepL()	Paso lateral a la izquierda
E	stepR()	Paso lateral a la derecha
K	kick()	Patada (para golpear una pelota)
Y	yes()	Movimiento intermitente de la cabeza
G	getUp()	Levantamiento desde una caída frontal
R	roll()	Rodar, se utiliza cuando el robot cae de espaldas
H	hello()	Saludo
q	miniTurnL()	Giro leve hacia la izquierda
e	miniTurnR()	Giro leve hacia la derecha
Z	punchL()	Puñetazo con el brazo izquierdo
B	punchR()	Puñetazo con el brazo derecho
C	crab()	Ataque de sumo con dos brazos
V	miniPunchR()	Puñetazo leve derecho
X	miniPunchL()	Puñetazo leve izquierdo
w	defense()	Posición defensiva
a	lookL()	Mirar hacia la izquierda
d	lookR()	Mirar hacia la derecha
L	look()	Mirar de frente
l	lookUp()	Mirar hacia arriba
f	endLookUp()	Volver a la posición de reposo tras lookUp()

Cuadro 8.1: Movimientos programados

### 8.3. Comunicación serie TODO

En la sección anterior hemos completado la programación de movimientos sobre la placa OpenCM, sin embargo, el control principal del robot se realiza desde la BeagleBone. En este apartado se comunicará la BeagleBone con la OpenCM de forma que adopten una configuración de maestro y esclavo. La estrategia consistirá en el envío de comandos hexadecimales desde la BeagleBone a la OpenCM. Cada comando servirá de identificador para un movimiento completo. Los comandos serán los descritos en el cuadro 8.1.

#### 8.3.1. Comunicación serie en OpenCM TODO

La OpenCM posee 3 puertos serie, de los cuales uno de ellos está reservado para el control del bus Dynamixel. En el esquema de la figura ?? se presenta la configuración de puertos serie de la OpenCM.

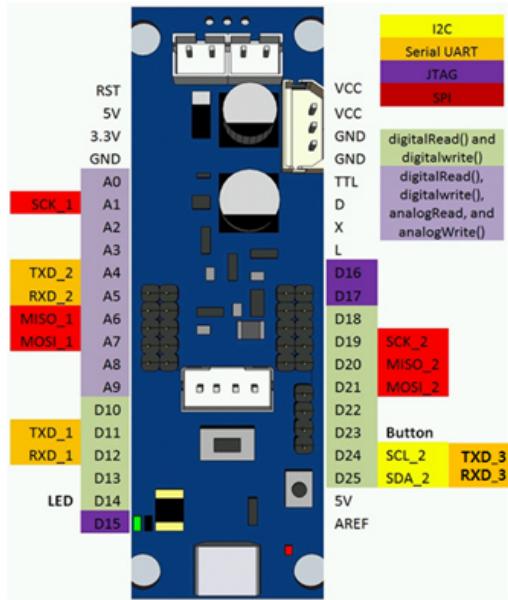


Figura 8.10: Esquema de puertos serie en una OpenCM 9.04

Para la programación de la comunicación serie en la placa OpenCM se ha utilizado la biblioteca `HardwareSerial.h`, contenida en CM9 IDE. La documentación de la biblioteca está disponible en ( TO-DO), sin embargo, a continuación se listan las funciones que se han utilizado junto a una breve explicación de su funcionamiento. ( TO-DO )

```
void HardwareSerial::begin(unsigned int baud)
```

Esta función habilita los pines como puerto serie. Como parámetro se le pasará la velocidad en baudios, que deberá coincidir con la del controlador maestro. En este caso, se ha configurado el puerto 3 con una velocidad de 9600 baudios

```
unsigned int HardwareSerial::available()
```

El objetivo de esta función es consultar si existe algún mensaje en el buffer de lectura. En caso afirmativo, podremos proceder a leer el mensaje.

```
unsigned char HardwareSerial::read()
```

La lectura de mensajes nos devolverá un valor hexadecimal. En nuestro caso, será el emitido por la BeagleBone Black y corresponderá a la ejecución de un movimiento.

**void HardwareSerial::flush()**

Con esta función vaciaremos el buffer de entrada. Se ejecutará cada vez que leamos un comando.

**8.3.2. Comunicación serie en BeagleBone**

Para poner en marcha la comunicación serie en la BeagleBone se ha utilizado como base la librería open-source Serialib ( TODO referencia gorda). Esta librería nos permite administrar un puerto serie, y está preparada para funcionar tanto en Windows como en Linux.

Se han realizado algunas modificaciones leves para ajustar su funcionamiento al requerido. Los métodos que se han utilizado ( TODO pueden encontrarse en el repo ) se detallan a continuación.

**open TODO nombre**

Esta función abrirá el puerto serie. En el primer parámetro se le pasará la cadena `/dev/ttyO2`, es decir, la dirección de nuestro puerto serie. Como segundo parámetro se le pasará la velocidad del puerto, que coincidiendo con la de la OpenCM será de 9600 baudios.

**writchar TODO nombre**

Para mandar los comandos utilizaremos esta función. Ya que el objeto de la clase conserva las características del puerto serie, solo será necesario pasarle el valor hexadecimal que queramos transmitir al controlador de locomoción.

**8.3.3. Comunicación con módulo Bluetooth TODO**

Adicionalmente, se ha incluido la posibilidad de utilizar un control auxiliar por Bluetooth desde un dispositivo externo. Para ello se ha conectado un modulo Bluetooth (figura ??) TODO directamente a la OpenCM. La función de esta otra vía de control no es otra que la realización de pruebas experimentales controladas, ya que nos permite modificar el comportamiento del robot en los momentos en los que sea necesario. Por supuesto, de cara a su funcionamiento autónomo, el módulo Bluetooth se inutilizará. Sin embargo, será necesario dejar preparada su conexión y programación. ( TODO foto de mexico )

Se ha utilizado un módulo Bluetooth hc05 conectado al puerto 3 del controlador de locomoción. Con esto, podremos mandar mensajes desde un teléfono móvil, un ordenador portátil o cualquier otro dispositivo que soporte conexión Bluetooth. Para utilizar el control del robot por Bluetooth se sustituirá el puerto de lectura de la OpenCM reservado al controlador de visión (el puerto 2) por el puerto 3.

## **8.4. Programación de sensores TODO**

### **8.4.1. Infrarrojos TODO**

Lecturas analogicas

### **8.4.2. IMU TODO**

Con todo el tostonazo de I2C

### **8.4.3. Brújula TODO**

## **8.5. Algoritmos de visión TODO**

### **8.5.1. Análisis de trayectorias en navegación TODO**

### **8.5.2. Búsqueda de lineas TODO**

### **8.5.3. Lectura de códigos QR TODO**



# **Capítulo 9**

# **Aplicaciones**

## **9.1. CEABOT 2014**

## **9.2. Spain Experience**

**Palabras clave:** palabraclave1, palabraclave2, palabraclave3.



# **Capítulo 10**

# **Presupuesto**

Se pres



## **Capítulo 11**

# **Evaluación de resultados**

Se presentan a continuación las conclusiones...

### **11.1. Pruebas de funcionamiento**

### **11.2. Conclusión**

Una vez finalizado el proyecto...

### **11.3. 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.