Máster en Informática y Automática Industrial

Trabajo Final de Master

Diseño y desarrollo de un módulo de gestión de IMU para Bioloid sobre Beaglebone Black

Alumno: Antoni Benavent Puertos
Director: D. Juan Francisco Blanes Noguera

València, Septiembre 2016

Índice general

Índice de ilustraciones

Índice de Tablas

Resumen

El presente proyecto, nace de la necesidad de dotar al robot humanoide Bioloid de la capacidad de obtener información acerca de su orientación, de modo que sea posible diseñar distintos bucles de control que permitan al robot, alcanzar o mantener una orientación concreta.

La herramienta principal que permitirá llevar a cabo la tarea de la obtención de la información acerca del estado del robot, será la I.M.U (Unidad de medición inercial). Para poder acceder a la información de la I.M.U, se utilizará como plataforma de desarrollo la BeagleBone Black que además permitirá la comunicación tanto con los actuadores del robot como con el controlador propio del robot según se necesite.

Esta posibilidad de acceso a la información de la I.M.U desde ambas plataformas, permite que el bucle de control se pueda definir en una u otra dependiendo de la aplicación. Originalmente no existía tal opción, por lo que se optó por diseñar un método de comunicación serie entre ambas plataformas.

La necesidad de variar la plataforma donde se implementa el bucle de control, se debe a que para la aplicación de control de estabilidad, donde se tiene que realizar cálculos para solucionar la cinemática inversa del robot, el controlador que lleva incorporado el robot no contaba con la suficiente potencia de cómputo. Por lo cual, y aprovechando que ya se utilizaba la BeagleBone Black para la conexión con la I.M.U, se utilizará dicha plataforma que si permite realizar todos los cálculos necesarios para llevar a cabo el control.

Finalmente, se realizarán los ensayos pertinentes a cada una de las aplicaciones diseñadas, para comprobar que los bucles de control funcionan correctamente y de acuerdo con lo esperado.

Antecedentes

El campo de la robótica avanza rápidamente, con mejoras continuas tanto a nivel de software como de hardware. Cada vez, son más las aplicaciones de robots diseñadas para prestar servicio directo a los humanos, borrando el concepto de robot solamente como herramienta industrial.

Ante este aumento de robots diseñados para interactuar con humanos, siempre destacan los robots humanoides, cuyo objetivo es emular comportamientos humanos del modo más fiel posible, ya sea caminar erguido o poder obtener información de una imagen.

Para conseguir que los robots realicen comportamientos motrices similares a los humanos correctamente, es necesario poder controlar variables como la posición o velocidad del robot. Es aquí donde aparece la importancia de una buena sensorización que garantice unos valores fiables, a partir de los cuales se pueda implementar y diseñar dicho control, y unos actuadores adecuados que permitan que las acciones de control se realicen correctamente. Afortunadamente, en la actualidad se dispone de una gran variedad de sensores y actuadores donde elegir el que mejor se adapte a cada aplicación.

El presente trabajo se centra en la orientación del robot, de modo que el propio robot sea capaz de orientarse o posicionarse según le convenga. En la actualidad, la mayoría de robots humanoides ya suelen contar con sistemas que les permiten obtener información acerca de su orientación y controlarla.

No obstante para robots humanoides más comerciales, como el Bioloid, estas características no se encuentran demasiado desarrolladas, la mayoría se basan en sensores de presión en las plantas de los pies, giroscopios o acelerómetro. Es en este punto donde aparece la importancia de la I.M.U, sensor que comprende acelerómetro, giróscopo y magnetómetro, cuyo uso se está expandiendo gracias a la calidad y gran variedad de datos que puede proporcionar.

Haciendo uso de la I.M.U y de la plataforma de desarrollo BeagleBone Black, se diseñarán distintos bucles de control que permitan al robot, controlar su orientación. Procurando ofrecer un diseño sencillo para que se pueda adaptar fácilmente a futuras aplicaciones.

1 Introducción y objetivos

1.1 Objetivos

El objetivo principal de este trabajo es desarrollar el software necesario para gestionar la información proporcionada por la I.M.U (Unidad de medición inercial), que permita su integración en bucles de control tanto locales como en un entorno distribuido, con el fin de mejorar el comportamiento motriz del robot humanoide Bioloid.

Para poder diseñar un bucle de control, es necesario definir una variable a controlar, en este caso: Yaw, Pitch y Roll, calculados a partir de la información proporcionada por la I.M.U, y un modo de actuar para poder corregir dicha variable, en este caso: funciones de movimiento y posicionamiento del robot.

Los comportamientos a mejorar son:

- Conseguir que el robot alcance una posición angular respecto su eje vertical, con una determinada con exactitud.
- Conseguir que el robot se estabilice en una superficie con inclinaciones variables.

Para conseguirlo se definen los siguientes subobjetivos:

- Desarrollo de la comunicación de la BeagleBone Black con la I.M.U.
- Análisis y estudio del comportamiento de la I.M.U.
- Análisis y estudio de la motricidad del robot humanoide Bioloid.
- Definición de las opciones para mejorar el comportamiento motriz del robot.
- Validación y test del trabajo realizado.

1.2 Introducción

El componente principal que posibilita la realización del proyecto, es la I.M.U. A continuación se explicará con más detalle su composición y funcionamiento.

1.2.1 I.M.U

Una unidad de medida inercial o IMU es un componente electrónico basado en sensores de aceleración (acelerómetro), velocidad angular (giróscopos) y campo magnético (magnetómetro), la cual aporta información acerca del movimiento y orientación que sufre dicha unidad. Es el componente principal de sistemas de guía inercial usados en vehículos aéreos, espaciales, marinos y aplicaciones robóticas.

También podemos encontrar unidades que incorporan un microprocesador que se encarga de recoger los datos de dichos sensores y enviarlos de forma ordenada al usuario mediante el protocolo de comunicación incorporado en la IMU.

A continuación se detallarán los componentes nombrados:

- **Acelerómetro**: instrumento capaz de medir aceleración en uno, dos o tres ejes. Existen varios tipos de acelerómetros, dependiendo de su fabricación y funcionamiento. Las I.M.Us incorporan acelerómetros integrados en silicio, utilizando la tecnología llamada MEMS6, debido a la necesidad de reducir el tamaño total de la unidad. La mayoría de éstos son capacitivos, y calculan la aceleración mediante el voltaje obtenido entre dos placas una de las cuales varía su posición dependiendo del movimiento del acelerómetro. Se caracterizan por ser muy precisos en situaciones estables y tener un gran error en situaciones vibratorias o movimientos muy inestables.

- **Giróscopo**: dispositivo que mide la orientación, basándose en los principios de la conservación del momento angular. Las unidades de medida inercial utilizan giróscopos MEMS, es decir, integrados y de tamaño reducido. La salida de dicho sensor es un voltaje, la variación del cual nos indica en grados por segundo (V/º/s) la velocidad angular sufrida por el sensor. Se caracterizan por tener un error constante y lineal llamado bias el cual debemos tener en cuenta.

- **Magnetómetro**: algunas unidades de medida inercial incluyen también sensores magnetómetros. Estos dispositivos miden la fuerza y/o dirección de los campos magnéticos que los afectan respecto el campo magnético terrestre. Aunque cabe la posibilidad de que se vean afectados por variación de otros campos magnéticos en algunas zonas.

- **Microprocesador**: algunas unidades de medida inercial, como ya se ha comentado, incorporan un microprocesador. Su principal función es recoger los datos entregados por los sensores, procesarlos según desee el usuario y enviarlos. La mayoría de los microprocesadores incorporan un conversor analógico-digital, para así dado por el sensor en una muestra.

- **Protocolo de comunicación**: los protocolos de comunicación alámbricos típicos en las I.M.Us son el UART, I2C, el RS-232 o el USB7. Algunas unidades incluyen protocolos inalámbricos, siendo los más utilizados ZigBee y Bluetooth.

2 Hardware utilizado y revisión de la tecnología actual

2.1 Alternativas placa computadora

Respecto a la placa computadora, las principales placas de bajo coste que se pueden encontrar en el mercado son: **BeagleBone y Raspberry Pi**. No obstante el proyecto estaba orientado al desarrollo de una aplicación para **BeagleBone**, de modo que se descartan las otras alternativas.

2.1.1 Solución adoptada, placa computadora

Ilustración 1 - BeagleBone Black Rev C

BeagleBone Black: El desarrollo de la BeagleBone se inicia con un Objetivo, diseñar un hardware abierto de bajo coste para desarrolladores no profesionales y aficionados a la electrónica. Gracias a su gran potencia y precio, facilita al máximo el desarrollo de aplicaciones. Además tiene como característica principal, el uso de **Capes**, que permite añadir funcionalidades extras (sensores, buses…) según la necesidad.

Desde Julio de 2011 han ido apareciendo distintas versiones, para esta aplicación en concreto se utilizará la **BeagleBone Black Rev C**.

Especificaciones técnicas:

- Procesador AM335x 1GHz ARM® Cortex-A8
- 512MB DDR3 RAM
- Almacenamiento interno de 4GB
- Acelerador de gráficos 3D
- NEON floating-point accelerator
- 2x PRU 32-bit microcontroladores (Para manejo de aplicaciones de Tiempo real)
- Conectividad:
    - Ethernet
    - USB host
    - HDMI
    - 2x46 pines de conexión
- Software
    - Debian
    - Ubuntu
    - Android

2.2 Alternativas I.M.U

Existen numerosas alternativas a la hora de seleccionar una I.M.U. Encontrando diferentes productos con calidades y precios variados. Para las aplicaciones a desarrollar los principales candidatos son los siguientes.

2.2.1 I.M.U Stick 9 DOF

Ilustración 2 - I.M.U Stick

La **I.M.U Stick 9DOF** es una pequeña placa con sensores que permite obtener 9 grados de libertad. Incluye un acelerómetro **ADX345**, un magnetómetro **HMC5883L** y un giróscopo **ITG-3200**, todos ellos de 3 ejes. El stick dispone de una sencilla interfaz I2C que permite comunicarse con los sensores. Esta placa sólo incorpora los sensores sin ninguna electrónica adicional.

2.2.2 I.M.U Razor

Ilustración 3 - I.M.U Razor

El sistema de medición inercial **9 DOF Razor I.M.U** dispone de 3 sensores, un giróscopo de tres ejes **ITG3200**, un acelerómetro **ADXL345** de 3 ejes, y un magnetómetro **HMC5883L** de 3 ejes. El conjunto proporciona 9 grados de libertad para medición inercial. Todas las salidas de los sensores son procesadas por un ATmega328 que envía y recibe la información por su puerto serie. La placa viene programada con el *bootloader* de **Arduino** (8MHz).

2.2.3 MPU 9150

Ilustración 4 - MPU 9150

La MPU-9150 se trata de un "*System in Package*" (**SiP**), el cuál combina dos chips: El **MPU-6050**, que contiene: un giroscopio de tres ejes, un acelerómetro de tres-ejes y un procesador interno "*Digital Motion Processor™*" (**DMP™**) capaz de realizar complejos algoritmos de fusión sensorial. Y la **AK8975**, un magnetómetro digital de tres-ejes. El fabricante, Invensense, proporciona unas librerías para poder adaptar la IMU a la aplicación deseada. Cuenta con una interfaz I2C que permite comunicarse.

2.2.4 Solución adoptada, IMU

La MPU-9150 es la seleccionada ya que proporciona 9 grados de libertad e incorpora el DMP para ofrecer unos resultados más robustos. Además se trata de la opción más económica.

A continuación se detallan sus especificaciones técnicas:

- Giróscopo de tres ejes con una sensibilidad hasta de 131 LSB/dps y con un fondo de escala variable entre: ±250, ±500, ±1000, y ±2000dps
- Acelerómetro de tres ejes con un fondo de escala variable entre: ±2g, ±4g, ±8g y ±16g
- Magnetómetro de tres ejes con un fondo de escala de ±1200μT
- Rango de tensión de alimentación: 2.4V–3.46V
- Corriente nominal del Giróscopo: 3.6mA
- Corriente nominal del Giróscopo + Acelerómetro: 3.8 mA
- Corriente nominal del Giróscopo + Acelerómetro + Magnetómetro + DMP: 4.25 mA
- Interfaz de comunicación I2C, modo rápido (400kHz)
- Reloj On-chip con una variación de ±1% de la frecuencia respecto a todo el rango de temperatura de operación

2.3 Robot humanoide Bioloid

El robot utilizado es el Bioloid, Fabricado por la casa Sur-Coreana **Robotis**, se trata de un kit muy similar al del Robonova-1. Pensado para gente que quiera iniciarse en la robótica, el fabricante ofrece el software **RoboStudio**, mediante el cual se puede programar de un modo muy visual, el comportamiento del robot. Así mismo, ofrece la posibilidad de programar el controlador del robot directamente en C, de modo que quien esté interesado pueda optar a una configuración personalizada.

Entre sus especificaciones se encuentran:

- Controlador CM530
  - ARM Cortex STM32F103RE
  - 6 botones
  - 7 leds
  - 6 puertos para sensores OLLO (Del mismo fabricante)
  - 5 puertos para motores Dynamixel AX/MX
  - Micro
  - Puerto USB
  - Puerto de conexión módulo Zigbee (UART)
- 18 servomotores Dynamixel
- Sensores de distancia IR
- Leds externos
- Mando a distancia

Ilustración 5 - Robot Bioloid

A destacar, los servomotores Dynamixel, los cuales presentan características muy interesantes:

Ilustración 6 - Servomotor Dynamixel AX

- Incorporan un controlador con el protocolo Dynamixel de comunicación implementado, lo cual permite la lectura y escritura de diversos parámetros (Par, Temperatura, Posición, Velocidad, Intensidad de corriente, Slope y Margin).
- Comunicaciones a gran velocidad (Hasta 1Mbps)
- Resolución de 0.29º
- Giro de 0º a 300, centrado en 150º

La alimentación del Robot la proporciona una batería recargable de Li-Po de dos Celdas de 12V y capacidad de 1000mAh. En el kit se proporciona un acople para poder cargarlas con el mismo cargador para alimentar al robot.

Ilustración 7 - Batería y adaptador

Un elemento importante para poder comunicar la BeagleBone Black con los servos Dynamixel es el USB2Dynamixel, el cual cuenta con una conexión de BUS dynamixel.

Ilustración 8 - USB2Dynamixel

2.4 Alternativas para la comunicación BeagleBone Black y CM-530

Como ya se ha especificado, el controlador del robot Bioloid, es el CM-530, el cual cuenta con diversos puertos de conexión tanto para servos Dynamixel como para sensores OLLO. Además de un puerto mini USB y un Puerto de comunicación (UART).

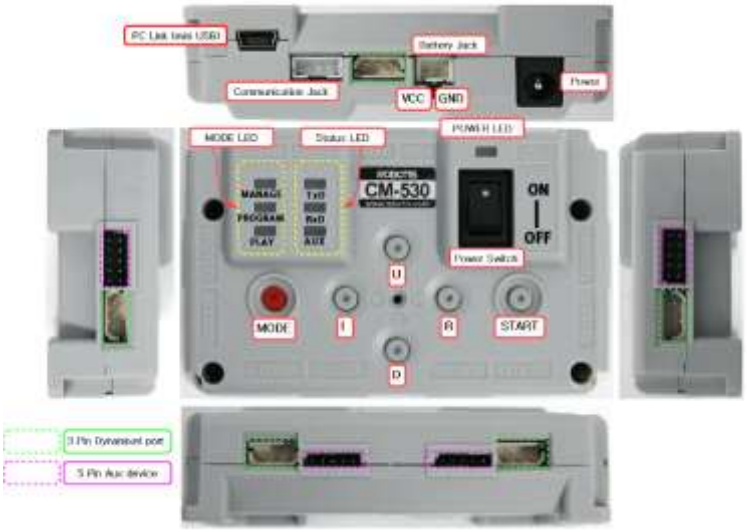


Ilustración 9 - Controlador CM-530

Surge la necesidad de poder comunicar la CM-530 con la BeagleBone Black ya que para la aplicación de corrección de la orientación en el eje Z del robot (ilustración 34), el acceso a los datos de la I.M.U se hace desde la Beagle mientras que las funciones de movimiento del robot se realizan desde la CM-530. Para conseguir la comunicación entre ambas plataformas, existen dos opciones que se muestran a continuación.

2.4.1 Comunicación por USB

Existe la opción de utilizar un cable USB a mini USB, de modo que el cabezal USB se conecta al puerto USB de la BeagleBone Black y el extremo mini USB a la CM-530, comunicándose en serie por dicho puerto.

Ilustración 10 - Cable USB a mini USB

2.4.2 Comunicación por UART

La siguiente opción era utilizar el puerto de comunicación de la CM-530 el cual presenta los siguientes pines, de modo que se pudiera comunicar con uno de los puertos serie de la BeagleBone Black:

1. GND : Ground Level (0v)
2. VDD : Supply Voltage (2.7~3.6V)
3. RXD : Receive Signal Terminal
4. TXD : Transmit Signal Terminal

Ilustración 11 - Puerto de comunicación de la CM-530

2.4.3 Solución adoptada, comunicación

La opción del cable USB a mini USB no resultaba factible ya que en una de las aplicaciones era necesario la conexión de una cámara USB a la BeagleBone Black de modo que el puerto ya se encuentra ocupado. Por lo tanto, la comunicación debía realizarse por el puerto de comunicación.

Para conseguirlo, se modificó uno de los cables que conectan a dicho puerto, eliminando el cable que conectaba la alimentación, ya que este puerto tiene un pin de alimentación para el caso en el que se conecta un módulo de comunicación Zigbee. En este caso no es necesario, solamente interesan los pines de RXD, TXD y GND. El resultado del cable es el siguiente:

Ilustración 12 - Cable de comunicación entre la CM-530 y la BeagleBone Black

Los extremos con los conectores macho se conectan a una de las UART de la BeagleBone Black, de modo que las conexiones quedarían del siguiente modo:

CM - 530	BeagleBone Black
GND	Pin 2
VDD	No conectado
RXD	20
TXD	21

Tabla 1 - Conexiones entre BeagleBone Black y CM-530 (Comunicación)

Con esta conexión, ya es posible transmitir información entre los puertos serie de la BeagleBone Black y la CM-530.

2.5 Alternativas para la alimentación de la BeagleBone Black

Debido a la necesidad de alimentar la BeagleBone Black, se barajó la posibilidad de utilizar una batería externa. No obstante esto representaba una mayor complicación a la hora de montar el conjunto entero en el robot. Por ello se optó por aprovechar la batería que ya incorpora el Robot.

2.5.1 Solución adoptada, Alimentación BeagleBone Black

2.5.1.1 Solución, Alimentación desde la CM-530

Se han adoptado dos soluciones, la primera consiste en alimentar la BeagleBone Black aprovechando la tensión que ofrece la CM-530 en los puertos destinados a los Sensores OLLO. Estos puertos presentan la siguiente configuración de pines:

1. OUT : 3.3V- Torque Possible (Maximum Allowed Current 0.3A)
2. VCC (5V)
3. ADC : The analog signals from the sensor made by the user can be read.
4. GND
5. OUT2 : 3.3V- Torque Possible (Maximum Allowed Current 0.3A)

Ilustración 13 - Puerto de conexión de un sensor OLLO en la CM-530

La alimentación de los sensores OLLO cuando se conectan a la CM-530 es de 5V, de modo que a priori sirve para alimentar a la BeagleBone Black. Para conseguirlo, se modificó uno de los cables con cabezal para el puerto del sensor, de modo que solo conectara los pines de alimentación y tierra. En el otro extremo del cable se soldó un cabezal para poder conectarlo a la alimentación de la Beagle. El resultado es el siguiente:

Ilustración 14 - Cable para la alimentación de la BeagleBone Black desde la CM-530

CM - 530	BeagleBone Black
OUT	No conectado
VCC	+
ADC	No conectado
GND	-
OUT2	No conectado

Tabla 2 - Conexiones entre BeagleBone Black y CM-530 (Alimentación)

Esta solución sirve para aquellas aplicaciones donde el robot tenga la CM-530, nos permite alimentar la BBB sin necesidad de reguladores de tensión externos.

2.5.1.2 Solución, Regulador de tensión

En aquellas aplicaciones donde se controlen los sensores y actuadores de Bioloid desde la BeagleBone Black, será necesario la instalación de un regulador de tensión de modo que se pueda utilizar la batería que acompaña al robot.

Se ha seleccionado un regulador genérico, de pequeño tamaño para facilitar su ensamblaje en el robot. El regulador soporta tensiones de entrada de 4.5V a 28 V y la salida se puede regular para conseguir tensiones de entre 0.8V y 20V.

El ajuste de la regulación de la tensión de salida se realiza mediante el potenciómetro que incorpora el regulador.

Ilustración 15 - Regulador de tensión

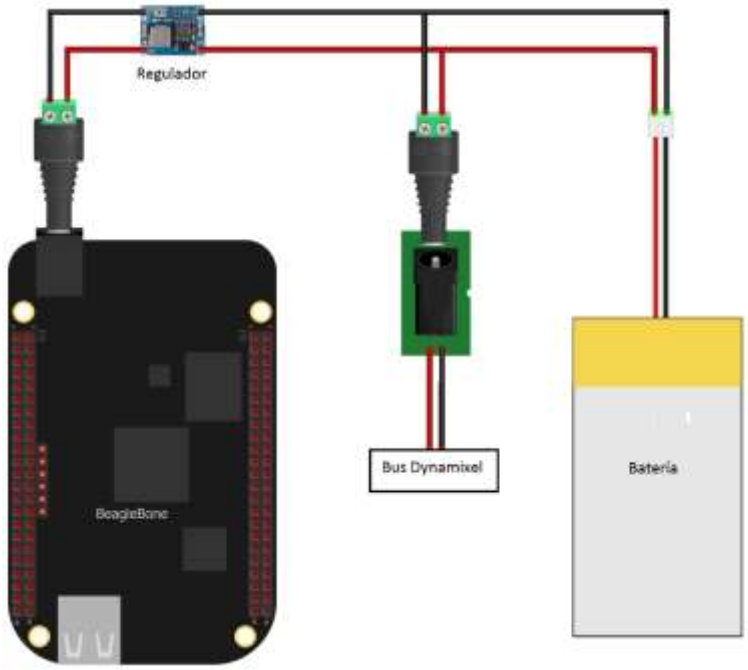


Ilustración 16 - Esquema de conexiones del regulador de tensión

Para alimentar el bus de Dynamixel es necesario proporcionar una tensión de 12V, como la batería ya proporciona dicha tensión, solamente con una conexión directa ya bastaría, pero se ha optado por añadir un conector macho y otro hembra de modo que se pueda desconectar la alimentación del bus si fuera necesario sin la necesidad de desconectar la batería.

3 Comunicación de la BeagleBone Black con la I.M.U.

Para poder comunicar la I.M.U con la BeagleBone Black y utilizar las funciones incluidas en las librerías de Invensense (Fabricante), hay que realizar una tarea de portabilidad (porting). Invensense desarrolla toda la librería utilizando unas funciones genéricas. La portabilidad consistirá en definir las funciones equivalentes de nuestra plataforma, respecto a las genéricas del fabricante.

3.1 Portabilidad

El grupo de desarrolladores **Pansenti** realizó la portabilidad para el modelo anterior I.M.U MPU-6090 en una **Raspberry Pi,** definiendo las funciones necesarias de modo genérico para sistemas basados en Linux. Ajustar dicha portabilidad requirió un par de ajustes, configurar correctamente el bus **i2c** que se iba a utilizar y la dirección de la I.M.U (0x69). Así como a la definición correcta de la plataforma a la hora de desarrollar en el entorno de programación **Eclipse**.

Los pasos seguidos para realizar la portabilidad se describen a continuación. Los ficheros proporcionados por el fabricante y Pansenti, son los siguientes:

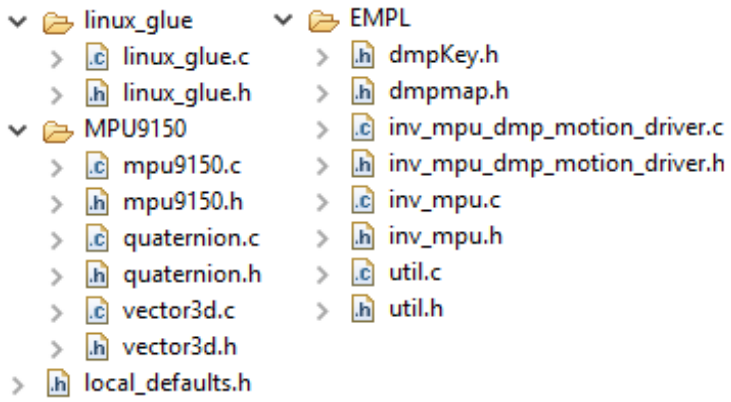


Ilustración 17 - Ficheros de la librería proporcionada por INVENSENSE y Pansenti

En el fichero **inv_mpu.c** encontramos las funciones que debemos definir:

```
/* The following functions must be defined for this platform:
*  i2c_write(unsigned char slave_addr, unsigned char reg_addr,*unsigned char
* length, unsigned char const *data)
* i2c_read(unsigned char slave_addr, unsigned char reg_addr,*unsigned char
* length, unsigned char *data)
* delay_ms(unsigned long num_ms)
* get_ms(unsigned long *count)
* reg_int_cb(void (*cb)(void), unsigned char port, unsigned char pin)
* labs(long x)
* fabsf(float x)
* min(int a, int b)
*/
```

En los ficheros de Pansenti encontramos los siguientes: **linux_glue.h** y **linux_glue.c.**
En el primero encontramos las declaraciones de los prototipos de las funciones:

```
void linux_set_i2c_bus(int bus);
int  linux_i2c_write(unsigned char slave_addr, unsigned char reg_addr,
unsigned char length, unsigned char const *data);
int linux_i2c_read(unsigned char slave_addr, unsigned char reg_addr,
unsigned char length, unsigned char *data);
int linux_delay_ms(unsigned long num_ms);
int linux_get_ms(unsigned long *count);
```

Y las definiciones de equivalencia entre estas y las del fabricante:

```
#define i2c_write       linux_i2c_write
#define i2c_read        linux_i2c_read
#define delay_ms        linux_delay_ms
#define get_ms          linux_get_ms
#define log_i           printf
#define log_e           printf
#define min(a, b)       ((a < b) ? a : b)
```

En **linux_glue.c**, se encuentran definidos los cuerpos de las funciones. A parte de las funciones necesarias para la portabilidad, también se encuentran definidas otras funciones para el correcto desarrollo de la comunicación i2c.

Una vez definidas las funciones y sus equivalencias, se realizan en los archivos **inv_mpu.c**, **inv_mpu.h** e **inv_mpu_dmp_motion_driver.c** las siguientes modificaciones:

Fichero	Línea	Modificaciones
inv_mpu.c	104	`#elif defined EMPL_TARGET_LINUX` `#include "linux_glue.h"`
inv_mpu.h	41	`#elif defined EMPL_TARGET_LINUX` `unsigned int pin;`
inv_mpu_dmp_motion_driver.c	65	`#elif defined EMPL_TARGET_LINUX` `#include "linux_glue.h"`

Tabla 3 - Modificaciones realizadas para la portabilidad

Finalmente, en el fichero principal se deberán incluir las siguientes definiciones, para que en el momento de acceder a la librería, la configuración sea la correcta:

- Plataforma a utilizar: `#define EMPL_TARGET_LINUX`
- I.M.U que se utiliza: `#define MPU9150`
- Modelo del Magnetómetro: `#define AK8975_SECONDARY`

I los correspondientes #include:

```
#include "util.h"
#include "mpu9150.h"
#include "linux_glue.h"
#include "local_defaults.h"
#include "inv_mpu_dmp_motion_driver.h"
#include "inv_mpu.h"
```

Al realizar estos cambios, ya se podrán utilizar las funciones de la librería proporcionada por el fabricante.

3.2 Definición local_defaults.h

En el fichero **local_defaults.h** se definen los valores de la frecuencia de muestreo y de yaw_mix_factor (utilizado para aplicaciones de aviónica), así como el bus i2c que se utilizará en la aplicación. La razón de definirlos en un fichero aparte, es para poder variar los valores de manera cómoda. Se configura una frecuencia de muestreo a su valor máximo de 100 Hz y seleccionamos el bus i2c 1.

3.3 Inicialización de la I.M.U y el D.M.P

En los ficheros del fabricante se proporcionan los siguientes archivos: **mpu9150.c** y **mpu9150.h**. En ellos se definen nuevas funciones para interactuar con el D.M.P™ y calibrar los sensores.

```
void mpu9150_set_debug(int on);
int  mpu9150_init(int i2c_bus, int sample_rate, int yaw_mixing_factor);
void mpu9150_exit();
int  mpu9150_read(mpudata_t *mpu);
int  mpu9150_read_dmp(mpudata_t *mpu);
int  mpu9150_read_mag(mpudata_t *mpu);
void mpu9150_set_accel_cal(caldata_t *cal);
void mpu9150_set_mag_cal(caldata_t *cal);
```

La función que nos interesa, es la **mpu9150_init(…)** puesto que se encarga de la inicialización de la comunicación i2c, de los sensores y del D.M.P™. Los pasos que sigue la función son:

	Tarea	Función
1	Inicializa la matriz de transformación (varía según la aplicación)	`signed char gyro_orientation[9] = {0, -1, 0, -1, 0, 0, 0, 0, 1 };`
2	Comprueba que el bus i2c definido es válido	`if (i2c_bus < MIN_I2C_BUS || i2c_bus > MAX_I2C_BUS) { printf("Invalid I2C bus %d\n", i2c_bus); return -1;}`
3	Comprueba que la frecuencia de muestreo es válida	`if (sample_rate < MIN_SAMPLE_RATE || sample_rate > MAX_SAMPLE_RATE){ printf("Invalid sample rate %d\n", sample_rate); return -1;}`
4	Comprueba que el mix_factor es válido	`if (mix_factor < 0 || mix_factor > 100) { printf("Invalid mag mixing factor %d\n", mix_factor); return -1;}`
5	Inicializa el bus i2c definido	`linux_set_i2c_bus(i2c_bus);`
8	Inicializa la MPU	`mpu_init(NULL)`
9	Inicializa el acelerómetro, el giroscopio y el magnetómetro	`mpu_set_sensors(INV_XYZ_GYRO|INV_XYZ_ACCEL| INV_XYZ_COMPASS)`
10	Configura la cola de salida de la MPU	`mpu_configure_fifo(INV_XYZ_GYRO | INV_XYZ_ACCEL)`
11	Configura la frecuencia de muestreo de la MPU	`mpu_set_sample_rate(sample_rate)`
12	Configura la frecuencia de muestreo del magnetómetro	`mpu_set_compass_sample_rate(sample_rate)`
13	Carga el firmware en el DMP™	`dmp_load_motion_driver_firmware()`
14	Carga la orientación en el DMP™	`dmp_set_orientation(inv_orientation_matrix_to_scalar(gyro_orientation))`
15	Activa las características del DMP™	`dmp_enable_feature(DMP_FEATURE_6X_LP_QUAT|DMP_FEATURE_SEND_RAW_ACCEL |DMP_FEATURE_SEND_CAL_GYRO | DMP_FEATURE_GYRO_CAL)`
16	Configura la frecuencia de la cola de salida del DMP™	`dmp_set_fifo_rate(sample_rate))`
17	Activa el DMP™	`mpu_set_dmp_state(1)`

Tabla 4 - Secuencia de funciones para la inicialización de la I.M.U

Una vez realizada la llamada a la función **mpu9150_init(…)**, disponemos de la función **dmp_read_fifo(…)** para obtener los datos provenientes de la IMU acorde a la configuración establecida.

3.4 Conexión con la BeagleBone Black

Para conectar la MPU 9150 al Bus I2C 1 de la BeagleBone Black hay que conectar los siguientes pines:

MPU 9150	BeagleBone Black
VDD	Pin 3
GND	Pin 1
SDA	Pin 20
SCL	Pin 19

Tabla 5 - Conexiones entre BeagleBone Black y MPU 9150

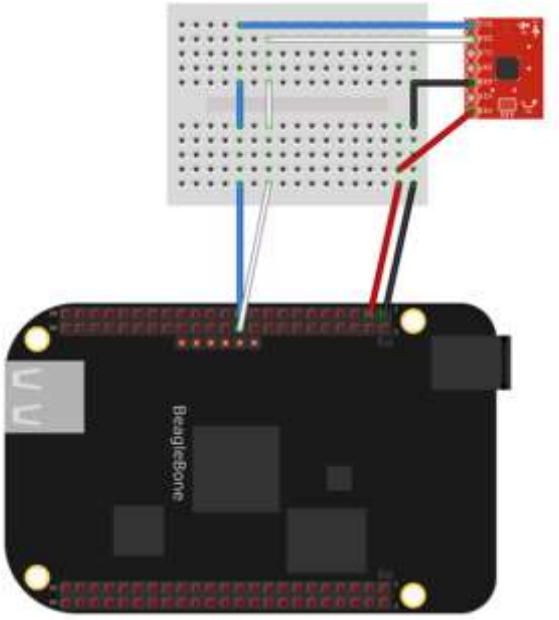


Ilustración 18 - Conexiones entre BeagleBone Black y MPU 9150

Para comprobar que la conexión es correcta, se puede ejecutar el comando: i2cdetect –y –r 1, donde obtendremos información acerca de los elementos conectados al bus 1 de la BeagleBone Black:

```
root@beaglebone:~# i2cdetect -y -r 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: 60 -- -- -- -- -- -- -- -- 69 -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@beaglebone:~#
```

Ilustración 19 - Resultado del comando i2cdetect

De este modo comprobamos que la conexión se ha realizado correctamente ya que aparece un elemento conectado en la dirección 0x69 (Dependiendo del cableado de la I.M.U también podría aparecer conectada en la dirección 0x68).

3.5 Calibración

El giroscopio lleva integrado un mecanismo de auto-calibración, en cambio la calibración del acelerómetro y del magnetómetro se ha de realizar de manera manual. En los ficheros de **Pansenti** se encuentra el archivo **imucal.c** el cual servirá para realizar la calibración.

Para consultar los distintos argumentos que pueden acompañar a la ejecución. Se ejecuta:

```
root@beaglebone:~/linux-mpu9150$ ./imucal –h.

Usage: ./imucal <-a | -m> [options]
      -b <i2c-bus>          The I2C bus number where the IMU is. The default is
1 for /dev/i2c-1.
      -s <sample-rate>      The IMU sample rate in Hz. Range 2-50, default 10.
      -a                    Accelerometer calibration
      -m                    Magnetometer calibration
                            Accel and mag modes are mutually exclusive, but one
must be chosen.
      -f <cal-file>         Where to save the calibration file. Default
./<mode>cal.txt
      -h                    Show this help

    Example: ./imucal -b3 -s20 –a
```

Será necesario ejecutarlo dos veces, una para el acelerómetro y otra para el magnetómetro. Al ejecutarlo en la **BeagleBone Black**, aparecerá lo siguiente:

```
root@beaglebone:~/linux-mpu9150$ ./imucal -a

    Initializing IMU .......... done
    Entering read loop (ctrl-c to exit)

    X -16368|16858|16858    Y -16722|-2490|16644    Z -17362|-562|17524
```

Los valores que se muestran, son mínimo | actual | máximo para cada eje.

Para la correcta calibración, se ha de girar la IMU respecto a cada uno de los ejes (XYZ). Los giros se han de realizar muy lentamente, debido a que se pretende medir únicamente la gravedad y los movimientos rápidos añadirían aceleraciones no deseadas.

Los valores se irán actualizando mientras se produzcan cambios en los valores de máximo o mínimo. Por lo tanto, cuando los valores se mantengan constantes habrá finalizado la calibración. Para finalizar el programa basta con pulsar Ctrl-C. Una vez finalizado, se creará un archivo accelcal.txt donde se guardarán los valores mínimo/máximo.

```
 root@beaglebone:~/linux-mpu9150$ cat accelcal.txt
    -16368  16858     -16722  16644    -17362  17524
```

Se realizará el mismo proceso para calibrar el magnetómetro, en este caso se pueden realizar movimientos más rápidos debido a que no se están midiendo aceleraciones.

```
root@beaglebone:~/linux-mpu9150$ ./imucal -m

    Initializing IMU .......... done
    Entering read loop (ctrl-c to exit)
    X -179|-54|121    Y -154|199|199    Z -331|-124|15
```

Una vez finalizada la calibración se finaliza el programa mediante ctrl-c y se generará el archivo magcal.txt que al igual que en el archivo anterior, es donde se guardarán los valores mínimo/máximo.

```
root@beaglebone:~/linux-mpu9150$ cat magcal.txt
    -179  121    -154  199    -331  15
```

Ambos archivos accelcal.txt y magcal.txt se han de guardar en el mismo directorio que el programa que contenga las funciones de calibración, ya que los utilizarán por defecto. De no existir dichos ficheros, se mostraría un mensaje por pantalla indicando que los ficheros no existen y no se realizaría la calibración. No obstante la I.M.U funcionaría sin problemas.

3.6 Comunicación y representación.

Para obtener una representación más visual de los datos proporcionados por la I.M.U y así poder comprobar también su correcto funcionamiento. Para ello se ha adaptado un Cliente escrito en Phyton, diseñado para la MPU-6050, que recibe los datos de la I.M.U por un socket U.D.P. Dicho Cliente representa la orientación espacial del sensor mediante un prisma rojo, así como sus valores de Roll, Pitch & Yaw.

Para la comunicación entre la BeagleBone Black y el computador anfitrión, hay que inicializar el socket U.D.P, para ello ejecutamos el programa **UDP_socket_init.c** en la BeagleBone Black**,** el cual creará e inicializará el socket

Una vez inicializada la configuración de la comunicación, se ejecuta en la BeagleBone Black el programa **imu_com.c**, para empezar a transmitir paquetes que contienen los datos obtenidos de la I.M.U (Yaw, Pitch y Roll) por el socket.

A su vez se inicia desde el computador anfitrión se inicia el programa **bbb_comm.py**. Deberán aparecer las siguientes ventanas:

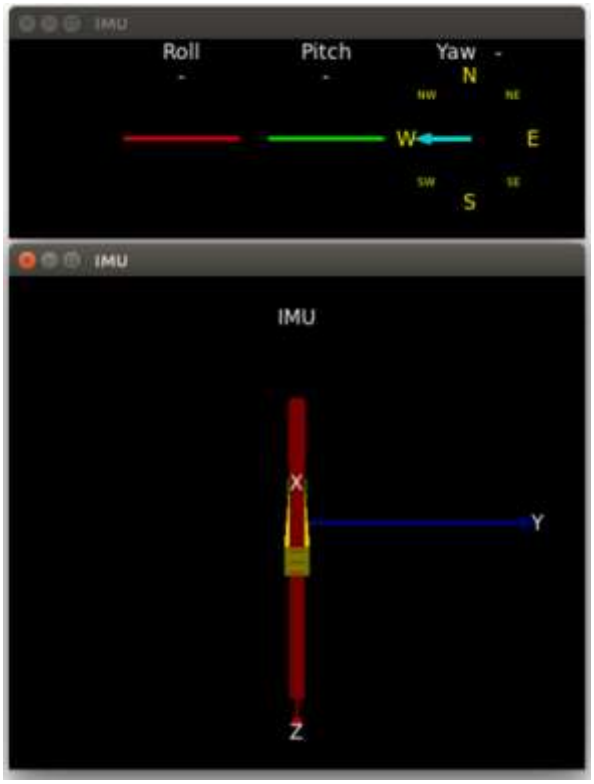


Ilustración 20 - Representación de la orientación espacial de la I.M.U (Al iniciar el programa)

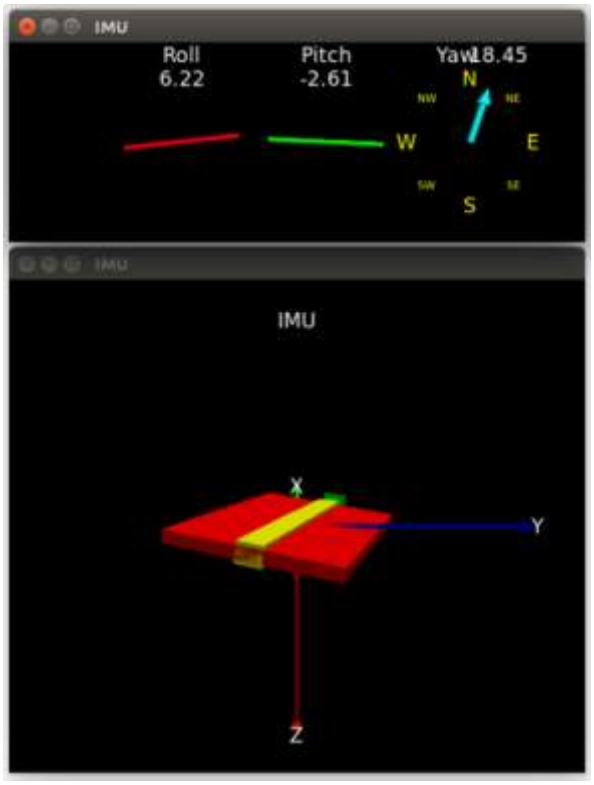


Ilustración 21 - Representación de la orientación espacial de la I.M.U (Recibiendo datos

4 Configuración del software

En general se realizarán dos configuraciones generales. El modelo 1, para la aplicación de corrección del Yaw y el modelo 2 para las aplicaciones de corrección del pitch y el roll.

En ambos modelos la configuración de la I.M.U será la misma.

4.1 BeagleBone Black

A continuación se explicará la estructura del proyecto base utilizado para el desarrollo de las aplicaciones.

4.1.1 Proyecto_base (Modelo 1)

Este proyecto base debe contar con las librerías Serialib y la de Invensense/Pansenti.

4.1.1.1 Serialib

Esta librería es necesaria en el modelo 1 para comunicarse por la UART con la CM530.

Para poder utilizar esta librería basta con añadirla a la carpeta del proyecto y en el fichero principal añadir su include correspondiente:

```
#include "serialib.h"
```

- En el caso del modelo 1 habrá que añadir las siguientes líneas en el fichero principal:
  - Habilitar la UART-4 de la BeagleBone Black

    ```
    system("echo BB-UART4 > /sys/devices/bone_capemgr.9/slots");
    ```

  - Definir el Puerto serial que se va a utilizar:

    ```
    #define          DEVICE_PORT_1              "/dev/ttyO4"
    ```

  - Se crea un objeto de la clase Serialib:

    ```
    serialib LS1;
    ```

  - Abrir el puerto serial, seleccionando la velocidad de 57600 baudios. En caso de error se mostrará un mensaje en pantalla:

    ```
    Ret=LS1.Open(DEVICE_PORT_1,57600);
    if (Ret!=1) {
          printf ("Error while opening port. Permission problem ?\n");
    }
    ```

Una vez el puerto está habilitado y abierto, se utilizan las funciones Write y Read para recibir y enviar información por el puerto serial:

`Char Write(const void *Buffer, const unsigned int NbBytes);`
Escribir un conjunto de bytes.
`int Read(void *Buffer,unsigned int MaxNbBytes, const unsigned int TimeOut_ms=NULL);`
Leer un conjunto de bytes, con la opción de añadir un timeout a la lectura.

Tabla 6 - Funciones principales utilizadas de Serialib

4.1.1.2 Invensense / Pansenti

Librería necesaria para poder acceder a la I.M.U. Una vez la librería está adaptada a la plataforma BeagleBone Black (Apartado 3), para poder utilizarla, en el fichero main.c solamente será necesario añadir las siguientes líneas:

- Definición de la plataforma, la I.M.U y el magnetómetro utilizado:

```
#define EMPL_TARGET_LINUX
#define MPU9150
#define AK8975_SECONDARY
```

- #Includes necesarios para acceder a los ficheros de la librería:

```
#include "util.h"
#include "mpu9150.h"
#include "linux_glue.h"
#include "local_defaults.h"
#include "inv_mpu_dmp_motion_driver.h"
#include "inv_mpu.h"
```

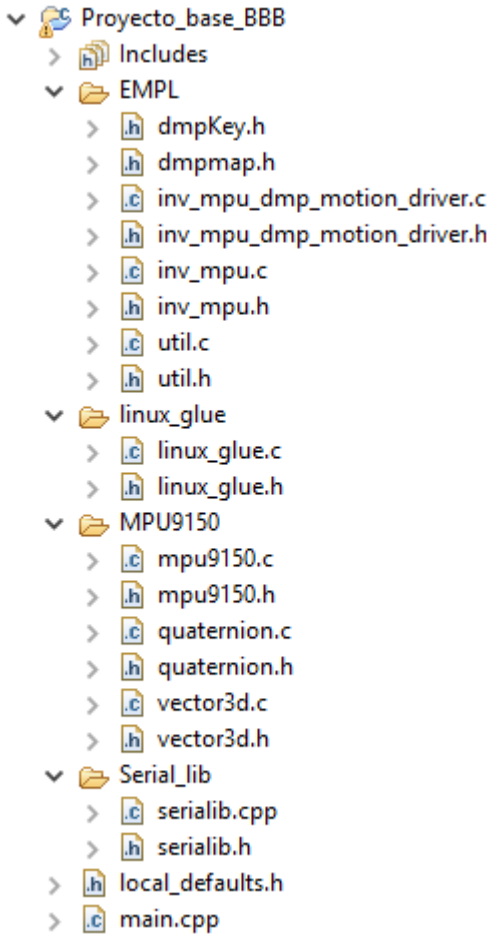


Ilustración 22 - Ficheros del Proyecto_base (Modelo 1)

4.1.2 Proyecto_base (Modelo 2)

Este proyecto debe contar con la **API para interacción de sensores y actuadores de robot Bioloid, basado en BeagleBone Black** y la **API de locomoción para robot Bioloid** así como de la librería Invensense/Pansenti.

4.1.2.1 API para interacción de sensores y actuadores de robot Bioloid, basado en BeagleBone Black

Esta API, desarrollada por David Sisternes Roses, permite la comunicación de la BeagleBone Black con los servomotores Dynamixel. Es necesaria para poder utilizar la librería API de locomoción para robot Bioloid

La configuración a realizar es sencilla, basta con añadir los ficheros a la carpeta del proyecto.

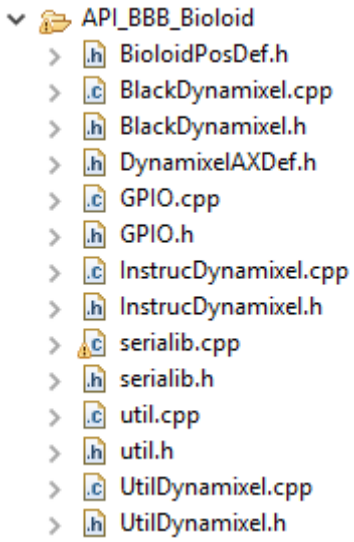


Ilustración 23 - Ficheros API BBB-Bioloid

Primero, se añaden al fichero principal los Includes correspondientes:

```
#include "BlackDynamixel.h"
#include "InstrucDynamixel.h"
#include "UtilDynamixel.h"
#include "DynamixelAXDef.h"
#include "BioloidPosDef.h"
```

A continuación se configura el puerto que se utilizará para la comunicación. En este caso se trata del puerto USB de la BeagleBone Black, donde se conectará el dispositivo USB2Dynamixel, el cual ya cuenta con conexión al bus Dynamixel de los motores, de modo que servirá de puente entre la placa y los actuadores.

- Se define el Puerto serial que se va a utilizar:

```
#define             DEVICE_PORT             "/dev/ttyUSB0"
```

- La velocidad de comunicación del Puerto:

```
#define             BAUD_RATE               115200
```

- Finalmente se abre el puerto para poder empezar la comunicación:

```
Open_Connection(DEVICE_PORT,BAUD_RATE);
```

Con esto, ya hay acceso a las funciones necesarias para poder utilizar la API de locomoción. Principalmente se trata de la función **setPosVel(),** la cual permite la escritura de valores de posición y velocidad a los servos Dynamixel.

```
uint8_t set_PosVel(uint8_t id, uint16_t pos, uint16_t vel,
                   uint8_t type_pos, uint8_t type_vel)
```

4.1.2.2 Invensense / Pansenti

Sigue la misma configuración que el apartado 4.1.1.2.

4.1.2.3 API de locomoción para robot Bioloid

Esta API, desarrollada por Juan Carlos Brenes, permite el cálculo de la cinemática inversa del robot Bioloid, de modo que se puede modificar la posición del robot dependiendo de donde se quiere situar las caderas en un espacio XYZ definido.

En la versión utilizada, las funciones no se encuentran en distintos ficheros. De modo que la configuración se reduce en copiar todo el código y pegarlo en el fichero principal, para así poder utilizar las funciones necesarias.

En las aplicaciones diseñadas, desde la función **main** se llama a las funciones:

`void IK6ServoLegtoHip(float xH, float yH, float zH, float pitchH, float rollH, float yawH, float legLen[5], int ang[6])`
Función que calcula la cinemática inversa de la pierna del robot, toma como referencia el pie y ubica la cadera. Recibe los puntos x, y, z de la ubicación deseada de la cadera. También los Ángulos pitch, roll y yaw de la orientación deseada. Modifica por referencia un vector con el valor de los Ángulos en grados. Estos ángulos toman el eje x del servo como cero
`void IKAdjust6ServoLeg (int ang[6], const bool isRightLeg)`
Función que recibe 6 ángulos calculados de la cinemática inversa de una pierna y los ajusta a las especificaciones del robot Bioloid y los servos Dynamixel. Diferencia entre pierna derecha o izquierda. Modifica por referencia un vector con el valor de los Ángulos en grados

Tabla 7 - Funciones principales de la API de locomoción

4.2 CM-530

El fabricante proporciona dos posibilidades para su programación. Mediante el software **RoboTask**, cuyo objetivo es permitir la programación del robot a aquellas personas que no estén familiarizadas con entornos de programación más avanzados. Y la programación en C del **ARM Cortex STM32F103RE** mediante el entorno **Eclipse**.

En este proyecto se utilizará la opción de programar en C, ya que permite una mayor libertad de operaciones y configuraciones. No obstante, esta opción también supone un incremento considerable de tiempo, ya que la correcta configuración del ARM es costosa si no se conoce el entorno.

En este apartado no se hará hincapié en la configuración de todas las funciones necesarias para un correcto funcionamiento, ya que sería extenderse en un tema que no es el principal. Todo el código se añadirá en los anexos, para poder analizarlo si fuera necesario.

Para configurar correctamente la CM-530, hay que ejecutar las siguientes funciones al inicio de la función **main**:

```
/* ---------------------- Configuration: ------------------------------ */
      /* Configura los relojes del sistema */
      RCC_Configuration();
      /* NVIC configuration */
      NVIC_Configuration();
      /* GPIO configuration */
      GPIO_Configuration();
      SysTick_Configuration();
      /* Configura los diferentes timers que se hayan definido */
      Timer_Configuration();
      /* Configuración del convertidor ADC */
      ADC_Configuration();
      GPIO_ResetBits(PORT_SIG_MOT1P,PIN_SIG_MOT1P);
      GPIO_ResetBits(PORT_SIG_MOT1M,PIN_SIG_MOT1M);
      /* Inicialización del puertode comunicación (Puerto Zigbee) */
      zgb_initialize(0);
      EnableZigbee();
      /* Configuración de la UART (Bus dynamixel) */
      USART1_Configuration(Baudrate_DXL);
      /* Inicialización de la comunicación con los servomotores */
      dxl_initialize( 0, 1 );
      /* Configuración de la UART del Puerto USB */
      USART_Configuration(USART_PC, Baudrate_PC);
      /* Vector con los identificadores de cada motor */
      for( i=0; i<NUM_ACTUATOR; i++ )
      { id[i] = i+1;
      }
      /* Led de POWER encendido */
      GPIO_ResetBits(PORT_LED_POWER, PIN_LED_POWER);
      /* Robot en posicion de espera */
      Robot_ready();
      mDelay(500);
```

De este modo se configura la CM-530 para:

- Enviar y recibir datos a los servomotores Dynamixel.
- Comunicarse por el puerto de comunicación y USB.
- Controlar los leds, botones, micro y zumbador incorporados.
- Realizar lecturas de los puertos para sensores OLLO.

4.3 Comunicación BeagleBone Black y CM-530

Primero hay que definir la velocidad a la que se van a comunicar ambas partes, esto será a 57600 baudios. A continuación se debe inicializar el puerto de comunicación de la CM-530 (El puerto serie de la BeagleBone Black ya se inicializa en el apartado 4.2.1.1), esta tarea se realiza al ejecutar la función **`zgb_initialize(),`** la cual ya tiene predefinidos los parámetros de velocidad y puerto que ha de configurar.

Una vez solucionado el problema de conexión entre los dos controladores, surge la necesidad de definir un protocolo para que los datos que se transmitan puedan entenderse por ambos. Para ello se enviará la información dentro de un paquete con la siguiente estructura:

FF	55	Data_L	~Data_L	Data_H	~Data_H

Tabla 8 - Estructura del paquete de comunicación entre BBB y CM-530

La elección de esta estructura no es arbitraria, se trata de la estructura utilizada para enviar y recibir información del módulo Zigbee disponible para el robot Bioloid. Al utilizar la misma, se aprovechan las funciones proporcionadas por el fabricante para la comunicación mediante Zigbee. De modo que cuando se desee enviar un dato a la BeagleBone Black, bastará con utilizar la función: **`zgb_tx_data(int data).`**

Para la recepción de información se dispone de la función: **`zgb_rx_check().`** La cual retorna un 1 si el paquete recibido cumple con la estructura definida.

Si el paquete es correcto, la función **`zgb_rx_data(),`** retorna el valor en hexadecimal de los datos recibidos, de modo que para convertirlo al valor decimal hay que realizar los siguientes pasos:

```
lowbyte = dxl_get_lowbyte(RcvData);
highbyte = dxl_get_highbyte(RcvData);

DATA  = (unsigned short)((highbyte << 8) & 0xff00);
DATA += lowbyte;
```

De este modo se consigue recibir en el CM-530 los datos enviados desde la BeagleBone Black.

En cuanto a la BeagleBone Black, las funciones **Read** y **Write** incluidas en la Serialib, permiten la lectura y escritura de un número de bytes determinado. Por lo tanto, solo faltan por implementar las funciones encargadas de codificar y decodificar el paquete.

La función **leer_puerto_serie**, se encarga de leer una cantidad determinada de bytes y a continuación buscar la cabecera del paquete entre los bytes recibidos. Finalmente comprueba que el paquete esté construido correctamente:

```
int Leer_puerto_serie(){

	char Buffer[128];
	int  Numero_bytes;
	int  timeout_ms = 1000;
	int  ret,i;

	ret=LS1.Read(Buffer,Numero_bytes,timeout_ms);
	if(ret == 1){
		for (i=0; i<25;i++){ // Búsqueda de la cabecera
			if (Buffer[i]==0xFF && Buffer[i+1]==0x55) {
				aux=i;
				i=30;
			}
		}
		if(Buffer[aux]==0xFF && Buffer[aux+1]==0x55){
			checksum_1 = ~Buffer[aux+2];
			if(Buffer[aux+3]==checksum_1){
				checksum_2 = ~Buffer[aux+4];
				if(Buffer[aux+5]==checksum_2){
		data = (unsigned short int)(Buffer[aux+4] << 8|Buffer[aux+2]);
					linux_delay_ms(10);
				}
			}
		}
	}
	return data;
}
```

La función **enviar_paquete**, se encarga de codificar el valor deseado para que cumpla la estructura del paquete.

```
void Enviar_paquete(unsigned short int data){

	unsigned char SndPacket[6];
	unsigned short word = (unsigned short)data;
	unsigned char lowbyte = (unsigned char)(word & 0xff);
	unsigned char highbyte = (unsigned char)((word >> 8) & 0xff);

	SndPacket[0] = 0xff;
	SndPacket[1] = 0x55;
	SndPacket[2] = lowbyte;
	SndPacket[3] = ~lowbyte;
	SndPacket[4] = highbyte;
	SndPacket[5] = ~highbyte;

	LS1.Write(SndPacket,6);
	LS1.FlushReceiver();
}
```

Una vez se tienen todas las herramientas para codificar y decodificar los paquetes desde ambos extremos, ya se puede garantizar una comunicación fluida entre la BeagleBone Black y la CM-530.

5 Montaje del sistema

Se habla de dos modelos ya que se han utilizado dos robots diferentes. En el primer modelo, el robot Bioloid cuenta con el controlador propio del robot CM-530, el cual se encarga de ejecutar las funciones para mover el robot (los servos Dynamixel van conectados a los puertos de la CM-530 correspondientes). Por otro lado se encuentra la BeagleBone Black con la I.M.U conectada. Ambos controladores se comunican utilizando el protocolo explicado en el apartado 4.3.

En el modelo 2 todo el proceso se realiza desde la BeagleBone Black, de modo que será necesario la colocación del USB2Dynamixel y el regulador de tensión. En los siguientes apartados se mostrarán los modelos con todos sus componentes ensamblados.

5.1 Modelo 1

El resultado general del modelo 1 es el siguiente:

Ilustración 24 – Parte frontal del robot, Modelo 1

Ilustración 25 - Parte trasera del robot, Modelo 1

Tanto la BeagleBone Black como la I.M.U se han colocado atornilladas en la parte delantera del tronco del robot, de modo que al colocar la carcasa del pecho queden protegidas

Ilustración 26 - I.M.U y BeagleBone Black (Con pechera)

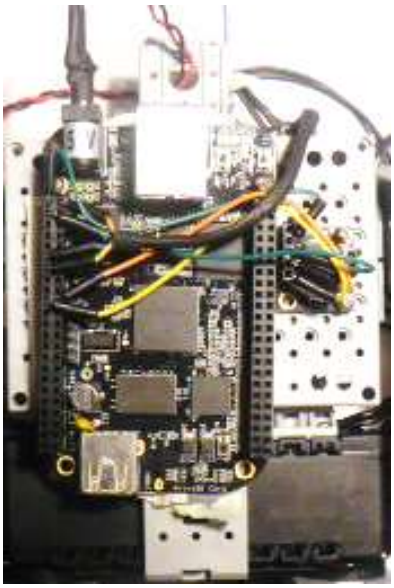


Ilustración 27 - I.M.U y BeagleBone Black (Sin pechera)

5.2 Modelo 2

El resultado general del modelo 2 es el siguiente:

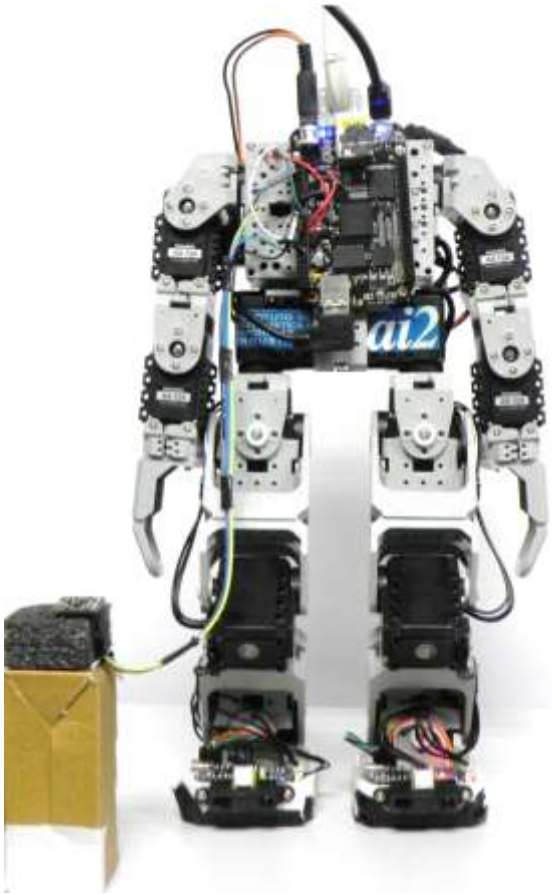


Ilustración 28 - Parte frontal del robot, Modelo 2

Ilustración 29 - Parte trasera del robot, Modelo 2

La BeagleBone Black se ha colocado del mismo modo que en el modelo 1, no obstante para este modelo la I.M.U se ha colocado en un soporte externo al robot. Esto se debe a que se necesitaba poder manipularla libremente en los ensayos para comprobar el efecto de distintos comportamientos.

En la parte trasera del robot se coloca el USB2Dynamixel, el regulador de tensión (Se encuentra justo debajo del USB2Dynamixel), además de un conjunto de pines para conectar los motores al bus Dynamixel.

Ilustración 30 - Colocación de la I.M.U en el Modelo 2

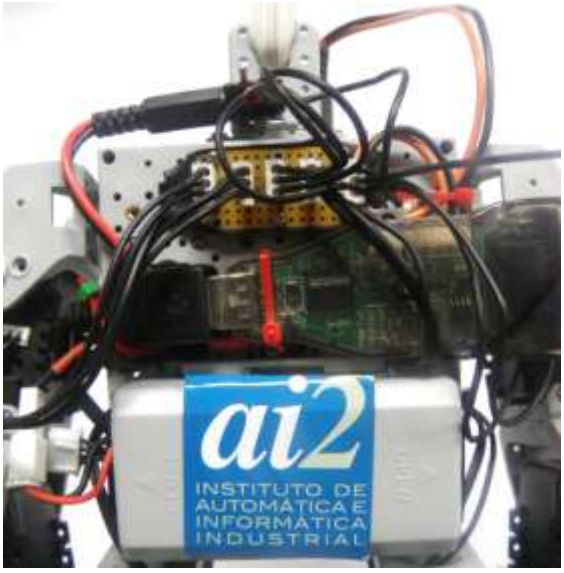


Ilustración 31 - Colocación USB2Dynamixel, Regulador de tensión y pines de conexión

6 Análisis y estudio del comportamiento de la I.M.U.

La I.M.U MPU-9150, puede proporcionar gran cantidad de información, ya sea respecto a la orientación (cuaterniones), valores de aceleraciones o giros realizados en los tres ejes XYZ. En este proyecto, cuyo objetivo es la mejora de la orientación las variables a utilizar son el Yaw, Pitch y Roll. A continuación se profundizará en los conceptos mencionados.

6.1 Ángulos de Euler y Ángulos de Tait-Bryan (Yaw, Pitch & Roll)

Para entender el funcionamiento de las aplicaciones diseñadas, es necesario entender los conceptos de Ángulos de Euler y Yaw, Pitch & Roll.

Los ángulos de Euler (introducidos por Leonhard Euler) constituyen un conjunto de tres coordenadas angulares (α, β, γ) que sirven para especificar la orientación de un sistema de referencia de ejes ortogonales, móvil, respecto a otro sistema de referencia de ejes ortogonales fijo.

Estos ángulos representan una secuencia de tres rotaciones elementales. Siendo la primera rotación sobre el eje z con un ángulo α, la segunda rotación sobre el eje y con un ángulo β y la tercera y última rotación de nuevo sobre el eje z con un ángulo γ.

Las rotaciones realizadas parten de un sistema de coordenadas fijo.

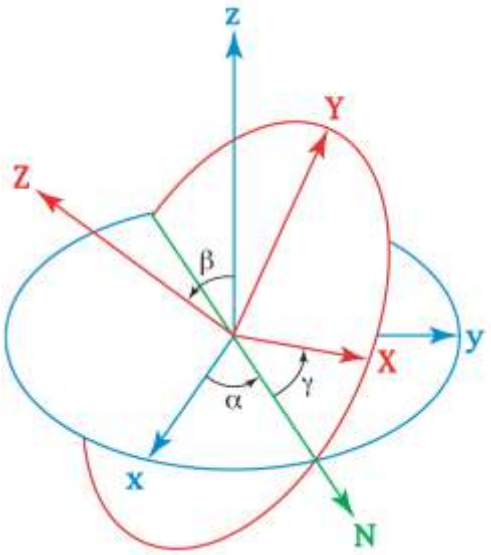


Ilustración 32 - Sistemas de coordenadas ortogonales en el que se muestran los ángulos de Euler

Muchas veces en ingeniería aeronáutica se utiliza el nombre de ángulos de Euler para hablar de los ángulos que en geometría se conocen como ángulos de Tait-Bryan (por el matemático escocés Peter Guthrie Tait). Comúnmente a los ángulos de Tait-Bryan se les suele llamar Yaw, Pitch y Roll.

La principal diferencia reside en que para obtener los ángulos de Tait-Bryan se realizan tres rotaciones sobre tres ejes distintos, mientras que para Euler dos de las rotaciones se realizan sobre el mismo eje.

Sin embargo, como ambas son formas de expresar la orientación de un cuerpo, existe una relación entre ellos; pudiéndose expresar unos en función de otros mediante una matriz de transformación.

Estos ángulos se prefieren en aeronáutica porque le asignan un ángulo de inclinación cero a un avión en horizontal, a diferencia de los ángulos de Euler, que le asignarían PI/2.

La principal ventaja de utilizar estos ángulos para realizar un control de orientación, es que definen una rotación de forma única alrededor de cada uno de los ejes intrínsecos del objeto.

Por lo tanto, cuando se habla de corregir el Yaw, Pitch o Roll, lo que se pretende es que los ángulos de Tait-Bryan se mantengan en un valor concreto.

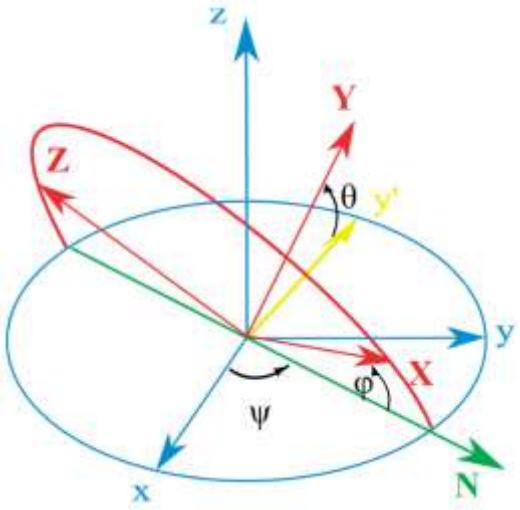


Ilustración 33 - Sistemas de coordenadas ortogonales en el que se muestran los ángulos de Tait-Bryan

Al iniciar el programa y con el robot en reposo, se tomarán los valores del Yaw, Pitch y Roll obtenidos con los datos de la I.M.U.

Con estos valores se definirá el sistema de referencia fijo al cual se referenciará el robot, de modo que cuando se modifique la orientación, aparecerán unos nuevos valores de Yaw, Pitch y Roll.

La diferencia entre los nuevos valores y los antiguos, será la variación que se ha producido en cada uno de los ejes.

El sistema de referencia del robot en cuanto a orientación, es el siguiente:

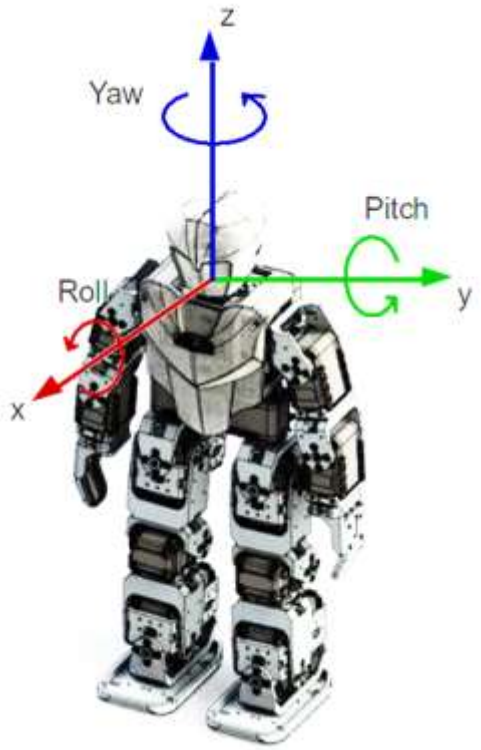


Ilustración 34 - Ejes Yaw, Pitch & Roll en el robot Bioloid

6.2 Obtención de los ángulos de Yaw, Pitch y Roll de la I.M.U

Para poder trabajar con los datos proporcionados por la I.M.U, es necesario realizar algunas tareas previas. Primero la calibración del acelerómetro y el magnetómetro, siguiendo el proceso explicado en el apartado 3.5. Es necesario realizar la calibración si se producen cambios importantes en el entorno del robot, ya que algunos de sus elementos como el magnetómetro pueden modificar notablemente su comportamiento.

Se ha configurado el D.M.P de la I.M.U para que proporcione el valor de los cuaterniones de la orientación del sensor. Para ello se dispone de la siguiente función, la cual realiza la lectura del buffer de salida del D.M.P:

```
dmp_read_fifo(gyro, accel, cuaternion, &sensor_timestamp, &sensors, &more);
```

De este modo se obtiene el valor del cuaternión, un vector de cuatro elementos $[q_1, q_2, q_3, q_4]$. Los cuaterniones permiten abstraer rotaciones y traslaciones con cierta simplicidad, permitiendo la obtención de la orientación relativa entre sistemas de coordenadas, de modo más sencillo y rápido que por ejemplo con el uso de matrices. Para poder utilizar los valores obtenidos, primero hay que normalizarlos:

Cálculo de la norma: $$q_{norm} = \sqrt{{q_1}^2 + {q_2}^2 + {q_3}^2 + {q_4}^2}$$

Normalización de los valores:

$$q_{1_{norm}} = \frac{q_1}{q_{norm}} \qquad q_{2_{norm}} = \frac{q_2}{q_{norm}} \qquad q_{3_{norm}} = \frac{q_3}{q_{norm}} \qquad q_{4_{norm}} = \frac{q_4}{q_{norm}}$$

Vector normalizado: $$q = [q_{1_{norm}}, q_{2_{norm}}, q_{3_{norm}}, q_{4_{norm}}]$$

Con los valores normalizados, se puede obtener los valores del Yaw (ψ), Pitch (θ) y Roll (ϕ), mediante las siguientes relaciones:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1 - 2(q_2^2 + q_3^2)} \end{bmatrix}$$

Para poder implementar estas ecuaciones utilizando lenguajes de programación, han de realizarse algunas modificaciones, ya que las funciones **arctan** y **arcsin** implementadas en las librerías de matemáticas, solamente ofrecen resultados para valores de ángulos entre $-\pi/2$ y $-\pi/2$, de modo que al realizar las tres rotaciones (una por cada eje) no se obtendrían resultados para todas las combinaciones posibles. Para corregir el problema y así obtener todo el rango de resultados, hay que substituir las funciones **arctan** y **arcsin** por **atan2.**

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(q_0 q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)) \\ \text{asin}(2(q_0 q_2 - q_3 q_1)) \\ \text{atan2}(2(q_0 q_3 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix}$$

De este modo se consiguen los valores de Yaw, Pitch y Roll, necesarios para el desarrollo de los bucles de control.

6.3 Adaptación del rango de valores del Yaw

Al realizar los cálculos explicados anteriormente, se devuelven valores de Yaw, Pitch y Roll comprendidos en el rango [-180º,180º]. Para poder utilizar el valor de Yaw del modo deseado es necesario disponer de un rango de valores que varíe entre [0º,360º]. Para conseguir el rango deseado, hay que sumar 360º a los valores negativos.

7 Análisis y estudio de la motricidad del robot humanoide Bioloid.

Respecto al movimiento del robot, en este proyecto se han utilizado dos variantes, la primera consiste en la técnica que proporciona el fabricante, la cual consiste en unos movimientos predefinidos, que se realizan siempre del mismo modo.

La segunda opción utiliza la **API de locomoción** desarrollada por Juan Carlos Brenes Torres, la cual permite el cálculo de la cinemática inversa del robot.

7.1 Movimientos predefinidos

La técnica que se utiliza para realizar movimientos con el robot consiste en lo siguiente:

- Se dispone de una matriz de n filas, donde cada fila supone un conjunto de posiciones para los servomotores del robot y 18 columnas donde cada una corresponde a un servomotor.

```
Word FWalk1[7][18]=
      {{233,786,279,744,462,561,358,666,503,555,353,725,267,807,641,386,533,544},
       {248,801,279,744,462,561,358,666,521,544,357,718,264,765,648,421,531,541},
       {263,816,279,744,462,561,358,666,528,537,355,713,259,744,652,437,528,537},
       {276,829,279,744,462,561,358,666,524,533,352,718,252,751,655,434,524,533},
       {287,840,279,744,462,561,358,666,519,528,353,720,248,760,660,427,519,528},
       {295,848,279,744,462,561,358,666,513,522,357,719,246,769,666,418,513,522},
       {301,854,279,744,462,561,358,666,507,516,364,716,248,775,671,409,507,516}};
```

- Para cada instante se enviará la información de una fila. Para ello se crea un paquete con la información codificada de modo que cada servomotor sepa cuál es la posición donde debe ir.
- Si la comunicación es correcta y para cada instante todos los servomotores consiguen alcanzar la posición deseada, el movimiento se realizará correctamente.
- Para poder realizar movimientos más complejos como andar o girar, será necesario definir un número mayor de matrices.
- Se pueden configurar diversos parámetros de los servomotores tales como el Par y la velocidad, para mejorar los movimientos.

Siguiendo esta premisa se han creado las diferentes funciones para realizar movimientos. A continuación se detallará el objetivo de dichas funciones cuyo código estará disponible en los anexos. Para ello se utilizará código de las siguientes fuentes:

- **10 DXL SYNC WRITE**: Ejemplo de escritura sincronizada por el bus Dynamixel proporcionado por Robotis en el paquete de ejemplos para programar la CM-530 en c.
- **Desarrollo e implementación de un sistema de control de movimientos mediante sensores para robot humanoide:** Trabajo final de grado de Carles Alcover Aguilar donde se implementan distintas funciones motrices para robot Bioloid con la CM-510.
- **Robomotion:** Software proporcionado por Robotis donde se pueden modificar y crear movimientos para el robot.

Nombre	Tarea
Robot_ready	Función simple que sitúa el robot en una posición erguida y estable.
Robot_walk	Función para realizar pasos hacia delante con el robot, permite indicar cuantos pasos se desea realizar.
Robot_giro_derecha	Función para realizar giros hacia la derecha, permite indicar cuantos pasos se quieren realizar. El desplazamiento angular de esta función si el movimiento se ha realizado correctamente es de 30 grados.
Robot_giro_izquierda	Función para realizar giros hacia la izquierda, permite indicar cuantos pasos se quieren realizar. El desplazamiento angular de esta función si el movimiento se ha realizado correctamente es de 40 grados.
Robot_giro_derecha_medio	Partiendo de la función de giro a la derecha, se modificaron los valores intermedios de la cadera del robot para que realizara un giro más pequeño, de modo que se consigue un desplazamiento angular de unos 15 grados. Esto se debe a que el desplazamiento del pie cuando está en el aire es menor, reduciendo el giro.
Robot_giro_izquierda_medio	Se siguió el mismo procedimiento que en la función anterior, consiguiendo reducir el desplazamiento angular, en este caso se redujo el desplazamiento angular hasta los 15 grados también.
Robot_giro_derecha_pequeno	Ante la necesidad de realizar un giro aún más pequeño, se optó por subir la velocidad de los motores, de modo que al realizarlo más rápido, no alcanzaban completamente la posición deseada en los pasos intermedios, reduciendo así el desplazamiento angular a menos de 10 grados.
Robot_giro_izquierda_pequeno	Aplicando el mismo procedimiento que con la función anterior, se consigue reducir el desplazamiento angular a menos de 10 grados.

Tabla 9 - Funciones de movimiento para el robot Bioloid (CM-530)

7.2 Movimientos mediante cinemática inversa

La API de locomoción permite realizar movimientos mediante el cálculo de la cinemática inversa del robot humanoide Bioloid. Gracias a ello, se consigue una mayor variedad de movimientos ya que no nos limitamos a utilizar unos movimientos predefinidos. Esta libertad de movimiento, permite que el robot pueda hacer frente a variaciones externas, como lo son los cambios de pendiente en una superficie.

El mecanismo de ejecución de los movimientos es el mismo que utiliza el fabricante. Es decir, se crea una matriz que contiene la información para realizar el movimiento y a continuación se transmite por el bus de comunicación la información a los servomotores.

El procedimiento que sigue la API de locomoción para generar la matriz de movimiento es el siguiente:

- Se define la referencia en el pie derecho del robot.
- Se definen las posiciones (x,y,z) y orientaciones (Yaw, Pitch y Roll) de ambas caderas respecto a la referencia.
- Se realiza el cálculo de la matriz de movimiento.
- Se envía la información a los servos.

No obstante, para poder utilizar dicha API, es necesario disponer de una potencia de cálculo elevada, ya que la complejidad y cantidad de cálculos a realizar va acompañada de la necesidad de que estos se efectúen rápidamente para que el robot pueda hacer frente de manera correcta a las variaciones externas. Para satisfacer la potencia de cálculo necesaria, se utilizará la BeagleBone Black.

8 Implementación del bucle de control

La obtención de los datos de la I.M.U se realiza desde la BeagleBone Black, no obstante la implementación del bucle de control se implementará en un sistema local (Beaglebone Black) o distribuido (CM-530), dependiendo de la aplicación a desarrollar.

Para la aplicación de corrección del Yaw, se ha utilizado el protocolo de comunicación entre la BeagleBone Black y la CM-530 definido en el apartado 4.3. Es necesaria dicha comunicación, ya que el controlador propio del robot no tiene acceso a la I.M.U y de este modo la BeagleBone Black podrá suministrar la información acerca del Yaw a la CM-530 para que pueda realizar el bucle de control que tendrá implementado:

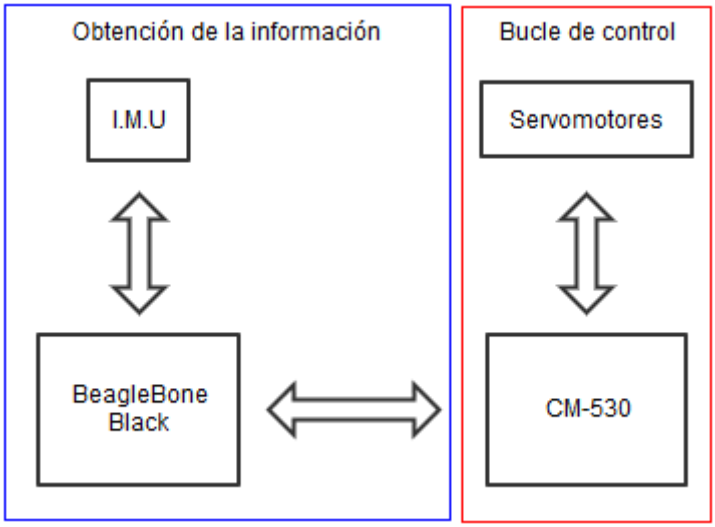


Ilustración 35 - Adquisición y control para la corrección de la orientación

En las aplicaciones de corrección de Pitch y Roll, la obtención de la información y la implementación del bucle de control se realizan en la BeagleBone Black, esta hará uso del USB2dynamyxel para comunicarse con los servomotores ya que no dispone de por si misma de un bus de conexión Dynamixel.

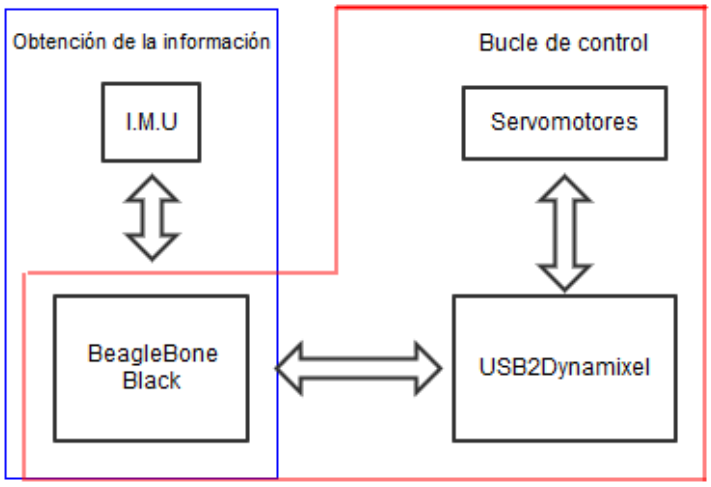


Ilustración 36 - Adquisición y control para la corrección del Pitch y Roll

9 Opciones para mejorar el comportamiento motriz del robot.

Se van a diferenciar entre dos tipos de soluciones. La primera, solución de giros iterativos, consiste en hacer uso de las funciones de movimiento definidas en el apartado 7.1, de modo que el robot sea capaz de alcanzar la orientación deseada (utilizando el modelo 1 de robot). En el segundo tipo, solución de reajuste de la posición del robot, se utilizará la API de locomoción para calcular a cada instante, la posición que debe adoptar el robot para compensar las variaciones (utilizando el modelo 2 de robot).

9.1 Solución de giros iterativos

En esta opción, el robot toma una orientación de referencia al inicio del programa, la cual servirá para orientarse respecto su eje Z, en futuros movimientos.

Dicha orientación viene proporcionada por la I.M.U en el valor del Yaw, y realizando las conversiones oportunas, se consigue que su rango de valores quede entre 0 y 360º, con lo cual se trabajará en grados.

El objetivo es alcanzar una orientación determinada referenciándose desde la orientación de referencia.

El procedimiento para conseguirlo se describe en el diagrama de flujo anexo.

Con esta técnica se consigue alcanzar la orientación deseada, pero teniendo en cuenta que existe un error de ±6º.

Durante los distintos ensayos realizados para la mejora de esta solución, se ha llegado a la conclusión que el error existente era aceptable para las tareas que iba a realizar el robot.

No existe acumulación del error al realizar varias veces la misma operación ya que para el cálculo de la orientación objetivo, siempre se parte de la orientación de referencia definida la primera vez, que es absoluta.

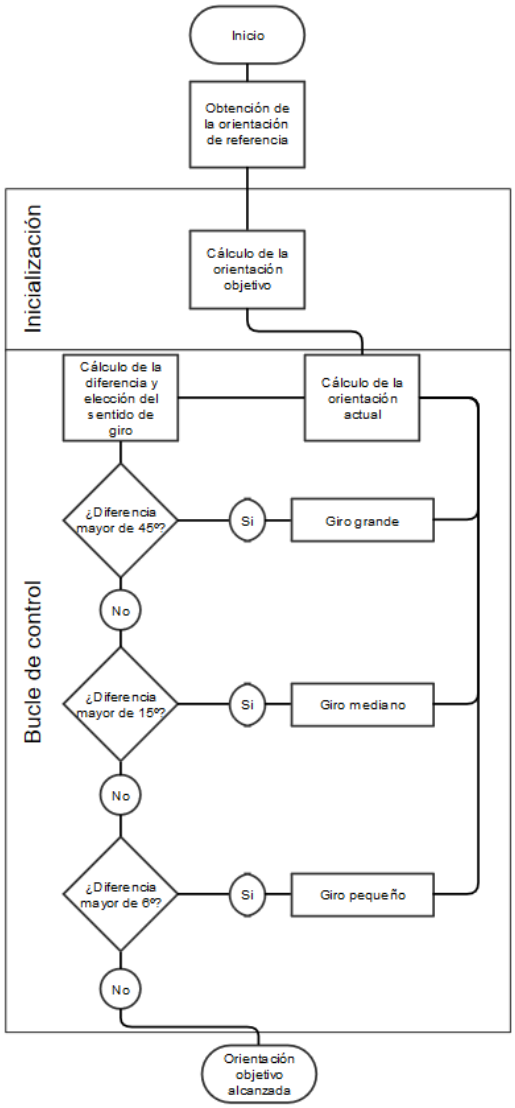


Ilustración 37 - Diagrama de flujo de la solución por giros iterativos

9.2 Solución de reajuste de la posición del robot

Existen distintas opciones para abordar el problema planteado, ya que cada pierna cuenta con seis grados de movilidad, de modo que según la articulación que definamos para corregir el Pitch y Roll, la posición adoptada por el robot, variará.

Para el desarrollo de este trabajo se ha optado por utilizar la cadera para corregir el Pitch y el Roll, ya que la API de locomoción permite definir un Pitch y Roll deseados para cada pierna además de una posición espacial deseada para la cadera.

9.2.1 Corrección del Roll

La corrección del Roll, es decir de los giros que se produzcan respecto al eje X del robot se realizará del siguiente modo. La orientación de los pies no variará ante los cambios de inclinación de la superficie. Mientras que los motores de la cadera variarán su posición acorde con estos cambios. Para completar la adaptación del robot, este debe doblar una de las piernas, dependiendo de hacia donde se produzca la inclinación, para que ambos pies queden paralelos.

A continuación se muestran las acciones explicadas anteriormente de modo separado, no obstante, el robot alcanzará directamente la posición final, sin pasos intermedios:

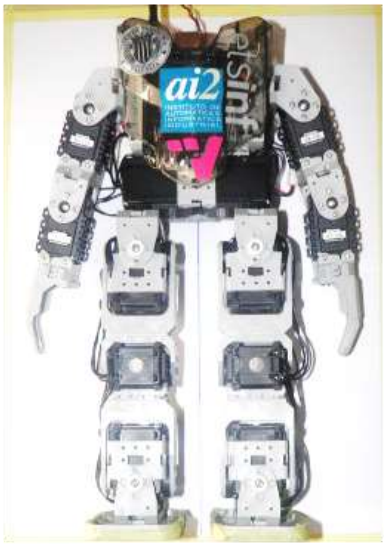


Ilustración 38 - Estado inicial del robot

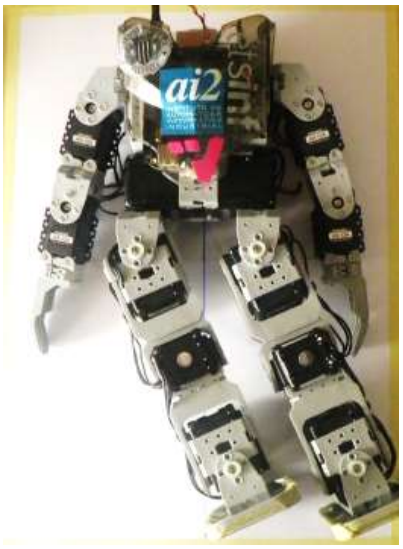


Ilustración 39 - Corrección del roll desde la cadera

Ilustración 40 - Ajuste de la posición de la pierna

El principal problema que presentaba esta aplicación, era la necesidad de definir la posición de la pierna que se encogía, para cada instante. Gracias a la API de locomoción, este problema se soluciona con un resultado muy efectivo, ya que solamente será necesario calcular la posición espacial donde se debe situar la cadera (se sitúa la cadera tomando como referencia el pie), y el ángulo de roll de esta. Una vez obtenidos estos datos, se dispone de la función IK6ServoLegtoHip () que realiza el cálculo de la posición a adoptar por el robot.

Por lo tanto se necesita conocer las posiciones de las caderas y el Roll a corregir. El valor del Roll se obtiene a partir de los datos de la I.M.U. Para conocer la posición de las caderas hay que aplicar trigonometría. Se definen 4 puntos P1 (Pie derecho), P2 (Pie izquierdo), P3 (Cadera derecha) y P4 (Cadera izquierda), el objetivo es que los cuatro puntos queden constantes o en función del ángulo de Roll (α).

Dependiendo del sentido que tome el giro se distingue entre dos casos:

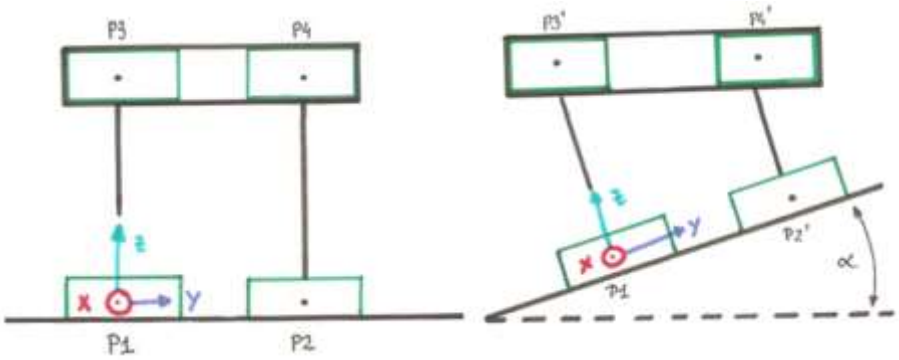


Ilustración 41 - Caso 1 (Giro positivo)

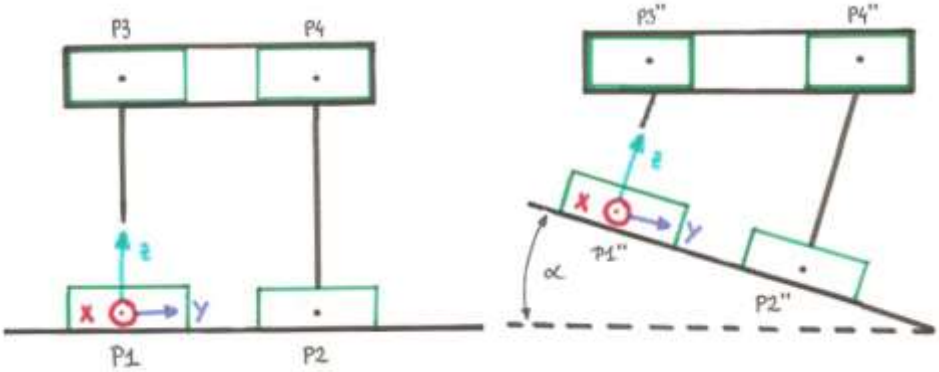


Ilustración 42 - Caso 2 (Giro negativo)

En la siguiente tabla se reflejan los valores de posición que toma cada punto para cada caso:

Punto	X (cm)	Y (cm)	Z (cm)
P1	0	0	0
P2	0	7.5	0
P3	0	0	14
P4	0	7.5	14
P1’	0	0	0
P2’	0	7.5	0
P3’	0	0	14
P4’	0	7.5	$14 - (7.5 \cdot tg(\alpha))$
P1’’	0	0	0
P2’’	0	7.5	0
P3’’	0	0	$14 + (7.5 \cdot tg(\alpha))$
P4’’	0	7.5	14

Ilustración 43 - Posición de los pies y las caderas ante distintas variaciones del Roll

Como se puede comprobar en la tabla, tan solo varían los valores de Z para las caderas, y obtener la posición correspondiente para contrarrestar la variación del Roll es sencillo. Con esto ya se dispone de toda la información necesaria para abordar el problema.

El bucle de control diseñado tiene un comportamiento similar al de un regulador proporcional de ganancia unitaria, ya que se dispone de un valor de Roll de referencia, de modo que a cada iteración se calcula la diferencia (hRoll) que existe entre el Roll actual y la referencia deseada. Este valor es muy importante, ya que se trata del ángulo que hay que aplicar en las caderas, y además se utiliza para calcular los nuevos valores de Z para las piernas.

Una vez calculados todas las variables implicadas, se procede a llamar a la función ik6servolegtohip () para calcular la posición del robot y finalmente enviar la información a los servomotores.

A continuación se muestra el diagrama de flujo que se corresponde con el comportamiento del Bucle de control.

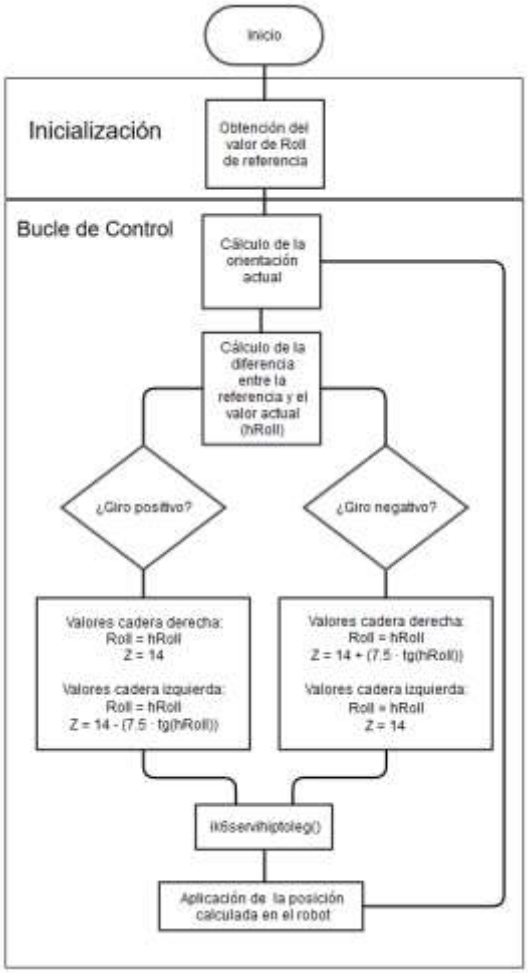


Ilustración 44 – Diagrama de flujo para la corrección del Roll

Finalmente, a continuación se muestran las posiciones teóricas que debería adoptar el robot para mantenerse estable ante las variaciones del Roll:

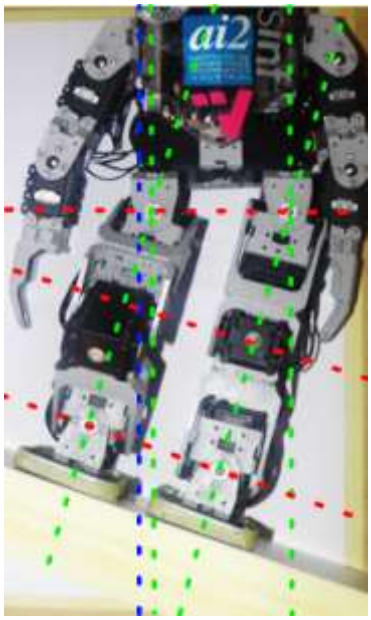


Ilustración 45 - Giro negativo (Roll)

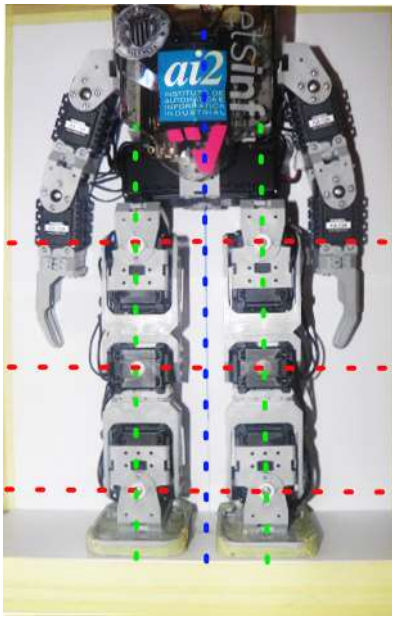


Ilustración 46 - Posición estable (Roll)

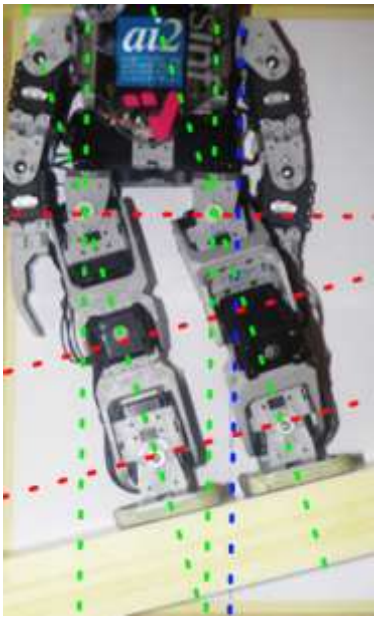


Ilustración 47 - Giro positivo (Roll)

9.2.2 Corrección del Pitch

La corrección del Pitch es más sencilla que la del Roll, ya que solamente es necesario el valor del ángulo del Pitch (proporcionado por la I.M.U). Esto se debe a que ambas piernas permanecen inalteradas durante todo el proceso, ya que no es necesario realizar ninguna modificación. Es la cadera quien realiza toda la compensación.

El bucle de control de nuevo, tiene un comportamiento similar a un regulador proporcional de ganancia unitaria.

Primero se define un valor de Pitch de referencia, a continuación se obtiene el valor de Pitch actual y se procede a calcular la diferencia entre ambos, para obtener el valor de hPitch.

Finalmente se llama a la función ik6servolegtohip () para calcular la posición del robot, y se mandan las instrucciones a los servomotores.

A continuación se muestra el resultado teórico con la posición que debería adoptar el robot frente a las variaciones del Pitch.

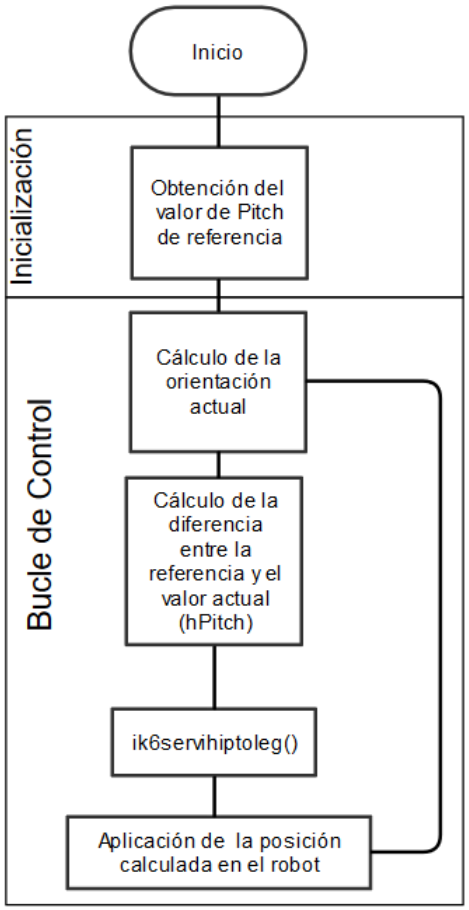


Ilustración 48 - Diagrama de flujo para la corrección del Pitch

Ilustración 49 - Giro negativo (Pitch)

Ilustración 50 - Posición estable (Pitch)

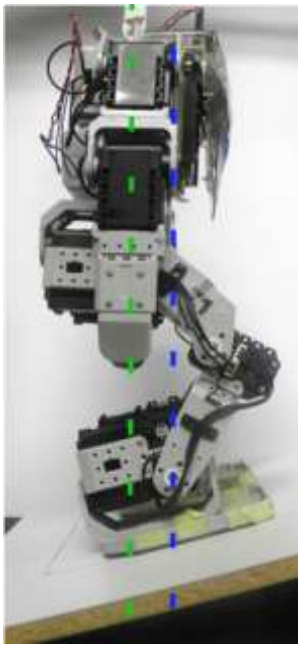


Ilustración 51 - Giro positivo (Pitch)

9.2.3 **Corrección del Roll y Pitch**

Al disponer de la opción de corregir tanto el Pitch como el Roll por separado, se decidió implementar una solución que corrigiera ambos a la vez. Tomando como base el método utilizado para la corrección del Roll, ya que es el más complejo, solamente habría que añadir el cálculo de la referencia del Pitch al inicializar el sistema, y a continuación, para cada instante calcular la diferencia entre esta referencia y el valor actual (tal como se realiza en el bucle de control de la corrección del pitch) obteniendo así el valor de hPitch que se añadiría a los parámetros a utilizar en el cálculo de la nueva posición. A continuación se muestra el diagrama de flujo del proceso.

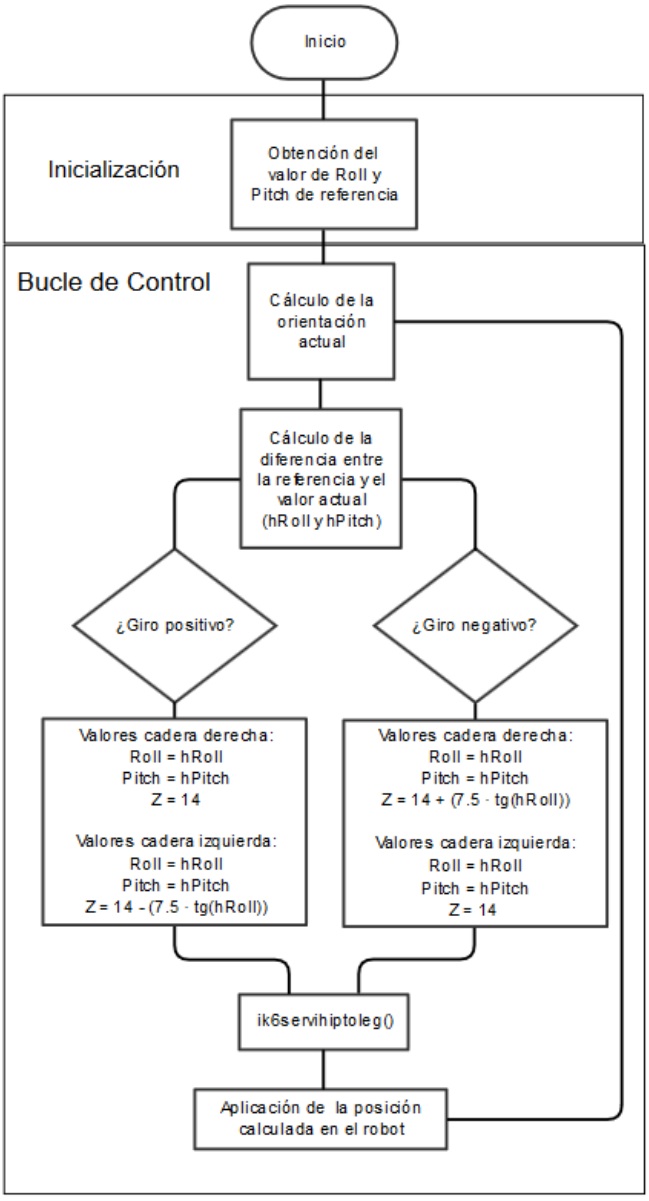


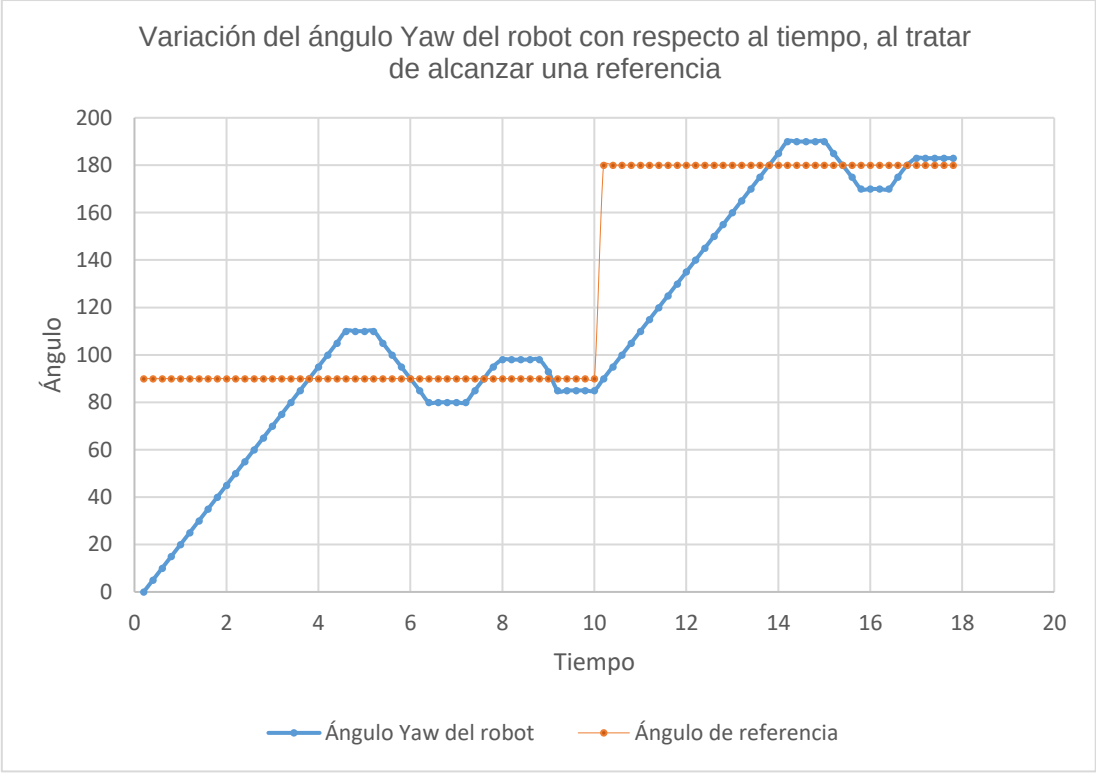
Ilustración 52 - Diagrama de flujo de la corrección del Roll y Pitch

10 Validación y test del trabajo realizado.

En este último apartado se procederá al análisis de los resultados obtenidos para ver si se han cumplido los objetivos.

10.1 Corrección de la orientación

La primera evaluación consiste en alcanzar primero la orientación de 90º y a continuación la de 180º. El resultado se ha graficado para poder ver la evolución del valor del Yaw, se muestra a continuación:



El comportamiento que muestra el sistema al tratar de alcanzar una orientación concreta, recuerda a una respuesta sub-amortiguada. El tiempo de establecimiento puede variar ligeramente en función del número de giros realizados para alcanzar la referencia, y de la rapidez a la hora de tomar los valores de la I.M.U. Al alcanzar el valor de referencia se puede contemplar el error esperado de ±6º esperado.

Para analizar la precisión y repetitividad, se han evaluado realizando repeticiones de seis veces el mismo giro, para giros de: 45º, 90º, 135º, 180º, 225º, 270º y 315º. Siempre partiendo desde el origen. Los resultados se muestran en el siguiente gráfico:

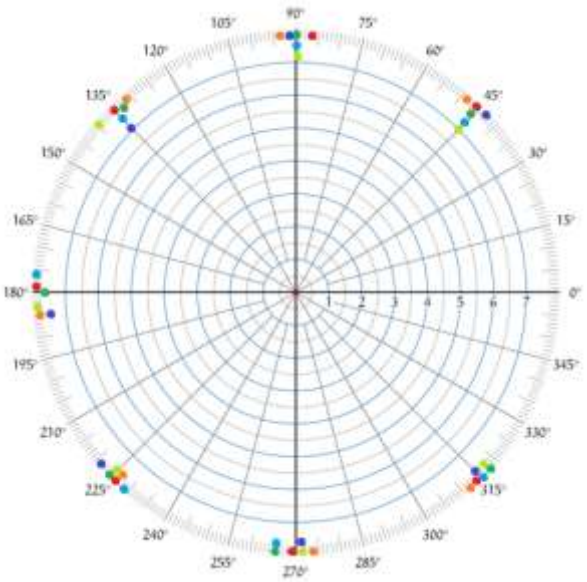


Ilustración 53 - Resultados del test de corrección de la orientación

Los resultados en cuanto a orientación, a primera vista parecen muy buenos, ya que todos los giros han alcanzado la orientación deseada dentro del error esperado de ±6º. No obstante existe un factor muy importante que no se suele ver reflejado al realizar solamente un giro para alcanzar una posición angular determinada. Este factor es el desplazamiento que sufre el robot al ir realizando los giros.

En el siguiente gráfico se analiza el efecto del desplazamiento, tomando como datos la posición (punto rojo) y orientación final (línea discontinua negra). Para ello, se sitúa el robot en la posición de origen y a continuación se realizan una serie de giros aleatorios.

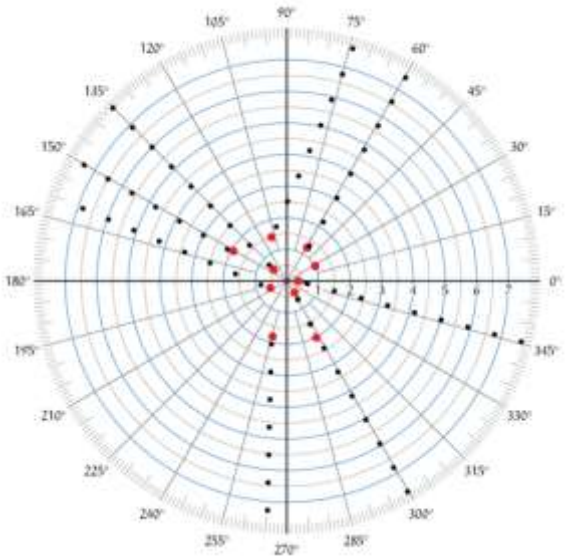


Ilustración 54 - Análisis del desplazamiento del robot

Concluyendo, la aparición del desplazamiento tiene cierta importancia en función de cual sea la aplicación, ya que si se realiza una corrección del Yaw al andar, de modo que el robot pueda recuperar una orientación determinada, este desplazamiento suele ser cuantitativamente menor que el desplazamiento que ya aparece al realizar una trayectoria larga.

En cambio si el objetivo de la corrección es realizar giros sobre si mismo de manera fiable sin desplazamiento, este efecto puede ser problemático ya que la orientación sería correcta, pero la posición espacial del robot no.

10.1.1 Implementación en aplicaciones

La corrección del Yaw se ha implementado en las aplicaciones diseñadas para la competición CEABOT 2016, concretamente en la prueba de Visión y la de obstáculos

En la primera aplicación, el robot tenía que descodificar unos códigos QR que contenían información respecto al giro que debía realizar el robot. Si este giro se realizaba correctamente, el robot debería terminar frente al siguiente código QR.

En la prueba de obstáculos se utilizó la corrección del Yaw para orientar el robot de un modo determinado dependiendo de la situación en que se encontrase. Con esta técnica se pudo implementar un algoritmo en el cual, el robot era capaz de evitar los obstáculos y a continuación recuperar una orientación concreta.

En ambas aplicaciones el resultado fue muy bueno, y nos permitió implementar técnicas que contrastaban con las utilizadas comúnmente. Como se reflejó en los resultados, esto implicaba una mejora notable.

10.2 Corrección del Roll

Para evaluar el comportamiento del diseño realizado, se sitúa al robot en una plataforma para poder variar el Roll con facilidad, los giros se realizan lentamente para que el robot pueda adaptarse. A continuación se muestran los resultados para un giro positivo y uno negativo en el eje X:

Ilustración 55 - Resultado Giro negativo

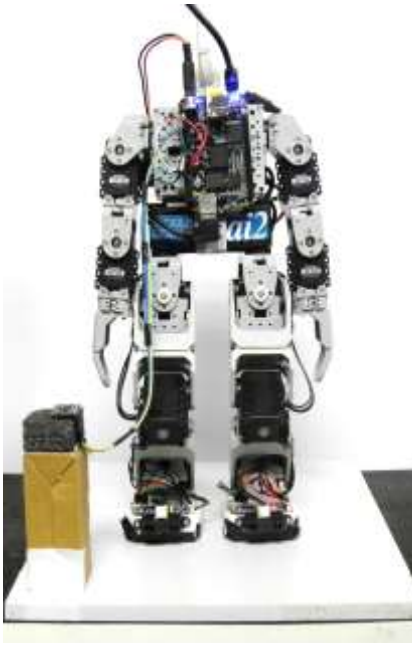


Ilustración 56 - Posición estable

Ilustración 57 - Resultado Giro positivo

Los resultados han sido satisfactorios ya que el robot se estabiliza correctamente. En las primeras pruebas el proceso de estabilización era muy lento. Tras depurar un poco el código y ajustar los tiempos de espera, se ha conseguido mejorar la respuesta, pero sigue sin ser todo lo rápida que uno desearía, ya que realizar todos los cálculos necesarios es una tarea costosa en términos de cómputo.

Respecto a los límites de inclinación que soporta el robot, para los valores teóricos, se situó al robot en la posición que le permitiría hacer frente a la máxima inclinación, sin que aparecieran colisiones entre las partes del robot. A continuación se calculó el valor de inclinación al que correspondería dicha posición.

Los valores reales, se han obtenido experimentalmente, llevando al robot hasta la inclinación que le provocaba una caída o deslizamiento. Al comparar los valores se observa una diferencia de 15º entre el máximo valor alcanzable teórico y el máximo real.

Por este motivo, se optó por repetir el experimento pero con los pies sujetos, comprobando que en este caso, el valor máximo si que es igual al teórico. En la siguiente tabla se detallan los resultados.

	Giro negativo (º)	Giro positivo (º)
Teórico	**-35**	**35**
Real (Pies sin sujeción)	**-35**	**35**
Real (Pies sujetos)	**-20**	**20**

Tabla 10 - Ángulos máximos para la corrección del Roll

La gran diferencia entre los valores reales (sin sujeción) y teóricos, se debe a que el robot es capaz de corregir su postura hasta el valor teórico máximo, no obstante, al sobrepasar los 20º de inclinación, los pies dejan de mantener la fricción necesaria para mantenerse y el robot se desliza.

10.3 Corrección del Pitch

Utilizando la misma plataforma que en el apartado anterior, se ha evaluado la respuesta del robot frente a los giros en el eje y.

Ilustración 58 - Giro positivo

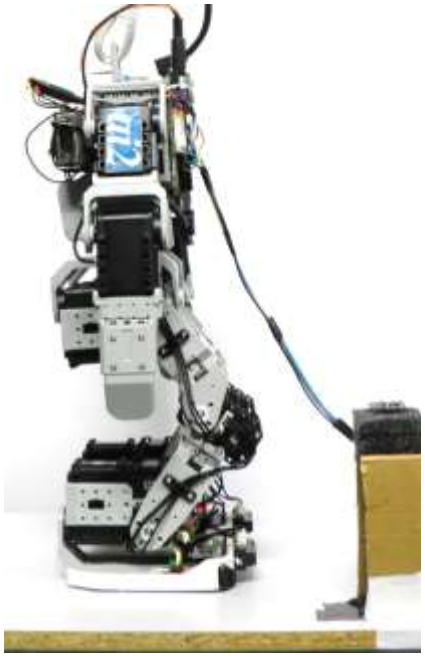


Ilustración 59 - Posición estable

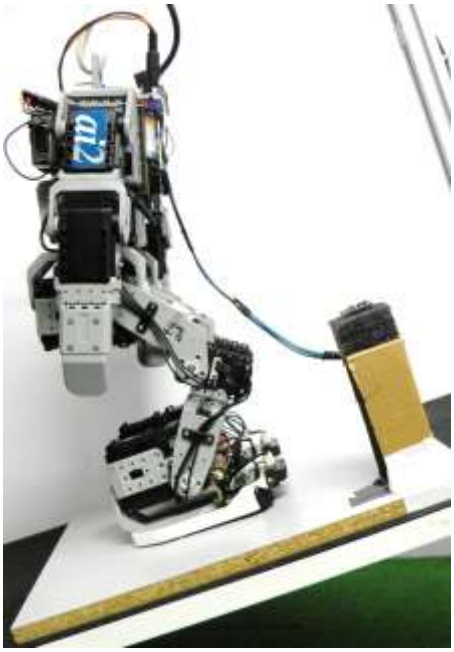


Ilustración 60 - Giro negativo

Los resultados han sido satisfactorios, ya que el robot ajusta su posición del modo esperado y más rápidamente que en la corrección del Roll.

Para el análisis de los límites de inclinación que soporta el robot, se ha repetido el mismo proceso que en el apartado anterior. Obteniendo los resultados que se muestran en la siguiente tabla.

	Giro negativo (º)	Giro positivo (º)
Teórico	**-60**	**50**
Real (Pies sin sujeción)	**35**	**35**
Real (Pies sujetos)	**-25**	**25**

Tabla 11 - Ángulos máximos para la corrección del Pitch

En este caso, los valores teóricos distan considerablemente de los reales, tanto con sujeción como sin ella. Al realizar la sujeción de los pies, se puede llegar a un ángulo de inclinación mayor, no obstante a partir de los 35º el esfuerzo que han de realizar los servomotores puede resultar perjudicial, ya que tiene que soportar gran parte del peso del robot.

Si por el contrario, no se realiza ninguna sujeción, el límite disminuye porque aparece de nuevo el deslizamiento a causa de que los pies no cuentan con suficiente agarre.

10.4 **Corrección del Roll y Pitch**

Utilizando la misma plataforma y colocando debajo una pelota, se consigue una superficie con la que variar Pitch y Roll fácilmente.

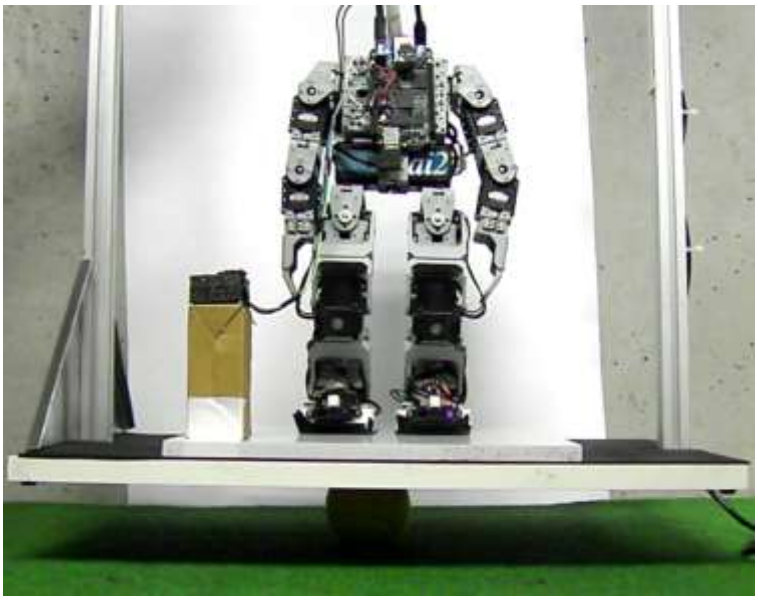


Ilustración 61 - Montaje para realizar variaciones de Pitch y Roll

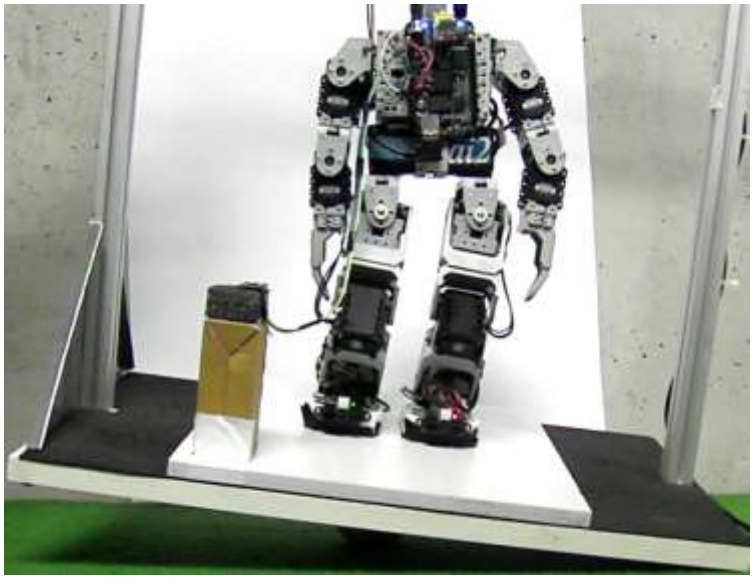


Ilustración 62 - Corrección de Pitch y Roll correcta

El resultado obtenido es satisfactorio, el robot consigue corregir su postura del modo esperado, tanto para cambios en el Pitch como en el Roll. En cuanto a límites de inclinación soportados, se ha comprobado que se mantienen los mismos valores obtenidos para los casos de corrección anteriores.

	Giro negativo (º)	Giro positivo (º)
Roll	**-20**	**20**
Pitch	**-25**	**25**

Tabla 12 - Ángulos máximos para la corrección conjunta de Pitch y Roll

11 Trabajos futuros

Una vez alcanzados los objetivos propuestos para este proyecto, se plantean las siguientes opciones para futuros trabajos:

- Adaptar la portabilidad realizada para la MPU-9150, para poder utilizar la nueva versión desarrollada por INVENSENSE, la MPU-9250. Esta nueva versión ofrece mejoras como la reducción del ruido que afectaba al giróscopo o la sustitución de los convertidores ADC de 8 bits por los nuevos convertidores ADC de 16 bits, lo cual mejora las medidas ofrecidas.

- Integración de la I.M.U en otros tipos de robot, en los cuales se pueda diseñar un sistema de navegación inercial. Es decir, que mediante el uso de la I.M.U conjuntamente con otros sensores, el sistema fuera capaz de obtener información acerca de su posición, velocidad, aceleración y orientación.

- Mejorar la velocidad de respuesta del robot al estabilizarse, de modo que sea capaz de hacer frente a cambios en la inclinación más rápidos.

- Estudio de opciones de posicionamiento diferente para permitir que el robot se estabilice ante cambio de inclinación. Existen diversas opciones para posicionar el robot y sería interesante poder compararlas, de modo que se valoren las ventajas y desventajas de cada opción.

- Incorporación de la corrección del Pitch y Roll en movimientos más complejos del robot. Sería interesante que el robot mantuviera una posición estable durante todos los pasos intermedios que componen un movimiento.

12 Conclusiones

Como conclusión principal, se puede afirmar que los objetivos definidos se han cumplido, proporcionando al robot humanoide Bioloid de la capacidad de obtener información acerca de su orientación y actuar en consecuencia. Permitiendo de este modo que el robot pueda realizar acciones con las que antes no podía contar, como son:

- Alcanzar una posición angular respecto su eje Z determinada.
- Adaptar su posición para mantenerse estable en una superficie que presenta cambios de inclinación.

La aplicación del control de la orientación respecto el eje Z del robot, se implementó en dos de las pruebas (visión y obstáculos) de la competición CEABOT 2016, en la cual participamos cuatro alumnos del Master. En estas pruebas, la posibilidad de tener controlada la orientación, fue crucial para conseguir los resultados obtenidos. En ambas pruebas se demostró la fiabilidad del control diseñado, permitiendo además la implementación de algoritmos para la resolución de las pruebas, que comportaran un comportamiento más similar al humano.

A destacar, el papel fundamental de la I.M.U, tecnología que cada vez se utiliza en más aplicaciones. En el campo de la robótica, se convierte en un sensor muy importante por la gran variedad de información que puede aportar, la cual puede permitir el diseño de aplicaciones no viables sin dicha tecnología. Además, existen gran variedad de opciones, lo que facilita la elección del modelo que más se adapte a las necesidades de la aplicación.

En cuanto a la posibilidad de trabajar con un robot humanoide, campo que no se había abordado directamente en las asignaturas de robótica cursadas en el master, ha facilitado el aprendizaje de gran cantidad de conceptos relacionados con la robótica humanoide, así como la aplicación de distintos métodos para controlar la motricidad del Robot, pudiendo comprobar las ventajas y desventajas de realizar un control en bucle cerrado sobre el posicionamiento del robot.

Para finalizar, comentar que el resultado global del trabajo, ha sido satisfactorio. Aunque ha requerido un gran esfuerzo para conseguir desarrollarlo, completar el diseño de un control funcional para después implementarlo en el Robot y comprobar que funciona correctamente, supone la confirmación de que el conocimiento adquirido durante el curso, ha sido comprendido y se dispone de el para hacer frente a nuevos retos.

13 Referencias y bibliografía

- Página de referencia de la MPU 9150:
  https://www.sparkfun.com/products/retired/11486

- Oficial Invensense webpage.
  http://www.invensense.com/products/motion-tracking/9-axis/mpu-9150/

- Arduino software implementing 9-axis data fusion using the InvenSense MPU-9150 IMU.
  https://github.com/zarthcode/MPU9150Lib

- BeagleBone ROS package that publishes the Invensense MPU-9150 data into a Topic.
  https://github.com/vmayoral/bb_mpu9150

- Software to interface the Beaglebone Black (BBB) with the MPU-6050 3 axis accelerometer and gyro IMU
  https://github.com/arembedded/imu_cam_strm

- Euler Angles
  https://en.wikipedia.org/wiki/Euler_angles

- Github repo with easy functions for cm530
  https://github.com/tician/cm530

- Conversion between quaternions and Euler angles
  https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles

- Quaternion
  https://en.wikipedia.org/wiki/Quaternion

- TFG: Desarrollo e implementación de un sistema de control de movimientos mediante sensores para robot humanoide – Carlos Alcover Aguilar

- TFM: API para interacción de sensores y actuadores de robot Bioloid, basado en BeagleBone Black – David Sisternes Roses

- TFM: API de locomoción para robot Bioloid – Juan Carlos Brenes Torres