

**UNIVERSIDAD DE ALCALÁ  
Escuela Politécnica Superior**

**INGENIERÍA TÉCNICA INFORMÁTICA DE GESTIÓN**

**Trabajo Fin de Carrera**

Representación en 3D de movimientos corporales  
capturados a través de acelerómetros y giroscopios

Autor: Jorge Regidor Martín  
Director: Julio Pastor Mendoza

**TRIBUNAL:**

Presidente: PEDRO ALFONSO REVENGA DE TORO

Vocal 1º: JOSÉ MANUEL VILLADANGOS CARRIZO

Vocal 2º: JULIO PASTOR MENDOZA

CALIFICACIÓN: .....

FECHA: .....



# Índice

## 1.1 Índice de Contenidos

Índice.....	3
<b>1.1</b> Índice de Contenidos .....	3
<b>1.2</b> Índice de Figuras.....	5
<b>1.3</b> Índice de Tablas.....	7
<b>1.4</b> Índice de Diagramas .....	8
<b>1.5</b> Índice de Código.....	9
<b>2.</b> Resumen.....	10
<b>2.1</b> Resumen en Castellano .....	10
<b>2.2</b> Resumen en Ingles (Summary in English) .....	10
<b>2.3</b> Palabras clave.....	10
<b>3.</b> Resumen extendido .....	11
<b>4.</b> Memoria.....	14
<b>4.1</b> Introducción .....	14
<b>4.2</b> Base Teórica .....	19
4.2.1 Conceptos Mecánicos .....	19
4.2.2 Conceptos Matemáticos .....	22
4.2.3 Tipos de comunicación.....	27
4.2.4 Materiales .....	31
4.2.5 Software .....	38
<b>4.3</b> Descripción Experimental .....	45
4.3.1 Conexión entre Arduino DUE y sensor MPU6050.....	45
4.3.2 Conexión con el DMP .....	46
4.3.3 Inicialización de los sensores.....	47
4.3.4 Calibrar los Sensores .....	48
4.3.5 Uso de Quaterniones.....	50
4.3.6 Comunicación con varios sensores .....	52

4.3.7 Comunicación entre Arduino y Unity 3D.....	54
4.3.8 Modelo 3D.....	57
4.3.9 Scripts de Movimiento .....	58
4.3.10 Interfaz de usuario .....	60
4.3.11 Hardware.....	64
4.3.12 Comunicación inalámbrica.....	66
4.3.13 Alimentación .....	68
4.3.14 Traje.....	68
<b>4.4 Conclusiones .....</b>	<b>69</b>
<b>5. Diagramas.....</b>	<b>70</b>
<b>6. Pliego de Condiciones .....</b>	<b>76</b>
<b>6.1 Condiciones generales .....</b>	<b>76</b>
<b>6.2 Condiciones Técnicas .....</b>	<b>76</b>
<b>6.3. Condiciones Económicas.....</b>	<b>76</b>
<b>7. Presupuesto .....</b>	<b>77</b>
<b>7.1 Materiales .....</b>	<b>77</b>
<b>7.2 Mano de obra.....</b>	<b>77</b>
<b>7.3 Licencias .....</b>	<b>78</b>
<b>7.4 Coste Total .....</b>	<b>78</b>
<b>8. Manual de Usuario .....</b>	<b>79</b>
<b>9. Contenido del Soporte Informático .....</b>	<b>96</b>
<b>10. Bibliografía .....</b>	<b>99</b>

## 1.2 Índice de Figuras

<b>Figura 1.</b> Sensor MPU6050: Sensor de la compañía Invensense [1] .....	11
<b>Figura 2.</b> Cuerpo Virtual 3D: Imagen del software que reproduce los datos de los movimientos recogidos.....	12
<b>Figura 3.</b> Captura de movimiento electromecánica: Ejemplo de la captura de movimientos electromecánicos de una mano[2] .....	14
<b>Figura 4.</b> Captura de movimientos electromagnética: Sistema de captura de movimientos electromagnética comercial de la marca Patriot.[2].....	15
<b>Figura 5.</b> Captura Óptica de Movimiento: Cara con sensores para la detección de movimiento mediante dispositivos ópticos. [2].....	16
<b>Figura 6.</b> Kinect: Descripción gráfica de las partes de un dispositivo Kinect.[3].....	17
<b>Figura 7.</b> Guante de Fibra óptica: Guante de captura los movimientos de una mano mediante fibra óptica [2]. .....	18
<b>Figura 8.</b> Google Glass: Imagen del uso de las gafas Google Glass. [7].....	18
<b>Figura 9.</b> Acelerómetro: Funcionamiento interno de un acelerómetro piezo-eléctrico. [4] .....	19
<b>Figura 10.</b> Giróscopo: sistema de funcionamiento de un giróscopo.[6] .....	20
<b>Figura 11.</b> Giróscopo Invensense: muestra del tamaño de un giróscopo de la compañía Invensense.[1] .....	21
<b>Figura 12.</b> Giróscopo MEMS: Vista de un giróscopo MEMS desde un microscopio. [31].....	21
<b>Figura 13.</b> Sistema coordenada: dibujo de planos en un sistema de coordenadas en 3D. [9].....	22
<b>Figura 14.</b> Vector: representación gráfica de un vector. [13] .....	23
<b>Figura 15.</b> Ángulos de Euler: representación gráfica de los movimientos de un cohete de la Nasa mediante los ángulos de Euler.[14] .....	24
<b>Figura 16.</b> Gimbal Lock: Ejemplo de bloqueo de ejes cuando gira Y se juntan X y Z. [15].....	25
<b>Figura 17.</b> Comunicación I2C: en la figura 1 vemos el esquema de conexiones de la comunicación y en la figura 2 vemos la comunicación de Start y Stop entre maestro y esclavo.....	28
<b>Figura 18.</b> Bluetooth: Sistema de saltos en la comunicación bluetooth. [20] .....	30
<b>Figura 19.</b> Arduino DUE : Imagen de la tarjeta Arduino Due. [22].....	32
<b>Figura 20.</b> Sensor MPU6050: Sensor de la compañía Invensense. [1] .....	33

<b>Figura 21.</b> Bluetooth Depeca: Modulo HC-05 junto a la tarjeta de adaptación del Departamento de Electrónica de la UAH. [23].....	35
<b>Figura 22.</b> Batería Lipo: Batería lipo de 7,4v 1000mah 2C. [25].....	37
<b>Figura 23.</b> Arduino vs Processing: Ambos IDE's de programación tienen cierto parecido.....	38
<b>Figura 24.</b> Unity 3D: Entorno de trabajo del software de videojuegos Unity 3D.....	39
<b>Figura 25.</b> Plataformas de Unity 3D: Menu de selección de plataforma del software de videojuegos Unity 3D. ....	40
<b>Figura 26.</b> MakeHuman: Software de creación de modelos humanos en 3D de licencia libre. ..	42
<b>Figura 27.</b> Librería MPU6050: Captura de datos para realizar la ingeniería inversa de Jeff Rowberg. [30] .....	44
<b>Figura 28.</b> Modelo 3D: Diseño del cuerpo humano 3D que más tarde reproducirá los movimientos.....	57
<b>Figura 29.</b> Modelo 3D en Unity: Vista del esqueleto del modelo 3D en Unity. ....	58
<b>Figura 30.</b> Interfaz de Usuario: En el ejemplo A vemos simplemente el modelo y en el B el humano es más pequeño y se incorporan los datos.....	61
<b>Figura 31.</b> Giro de la cámara: En esta imagen se muestra el modelo matemático que demuestra cómo se pasa de un movimiento lineal a circular mediante senos y cosenos.....	63
<b>Figura 32.</b> Hardware: Shield para Arduino Due que redirige la señal I2C a 12 salidas diferentes. ....	64
<b>Figura 33.</b> Bluetooth Depeca: Bluetooth diseñado por el Departamento de Electrónica de la Universidad de Alcalá. ....	66
<b>Figura 34.</b> Traje: estructura del traje que porta el proyecto y se adapta al usuario para crear los movimientos.....	68

## 1.3 Índice de Tablas

<b>Tabla 1.</b> SensoresMPU6000/ MPU6050: Características. [1] .....	34
<b>Tabla 2.</b> Relación Sensores-Modelo3D: Relación de cada uno de los sensores con las articulaciones del modelo.....	60

## 1.4 Índice de Diagramas

<b>Diagrama 1.</b> Camino de los datos: Representación gráfica del camino de los datos, desde la captura hasta la representación.....	13
<b>Diagrama 2.</b> Diagrama del Sensor MPU6050/MPU6000: Funcionamiento de la comunicación del sensor. [1] .....	33
<b>Diagrama 3.</b> Bluetooth: Funcionamiento del bluetooth. [23].....	35
<b>Diagrama 4.</b> Módulo 74HC4052: Funcionamiento de la conmutación del dispositivo.[24].....	36
<b>Diagrama 5.</b> Comutación I2C: Ejemplo de cómo conmutar la señal I2C con un módulo 74HC4052.[4] .....	52
<b>Diagrama 6.</b> Shield.sch: esquema de la Trajeta Shield.....	65
<b>Diagrama 7.</b> MPU6050: Esquema de funcionamiento del sensor MPU6050 [1]. .....	70
<b>Diagrama 8.</b> Arduino Due: Esquema de funcionamiento de la Arduino Due , se puede ver con mayor calidad en la pagina web de Arduino[23]. .....	71
<b>Diagrama 9.</b> Shield.sch: Esquema de funcionamiento de la shield.....	72
<b>Diagrama 10.</b> Shield.brd: Esquema la tarjeta shield.....	73
<b>Diagrama 11.</b> 74HC4052: Esquema de funcionamiento del módulo 74HC4052.[24] .....	74
<b>Diagrama 12.</b> Placa Depeca + HC05: Esquema de funcionamiento del bluetooth. [23] .....	75

## 1.5 Índice de Código

Arduino(MPU6050.h) Adaptación a Arduino Due 1.....	45
Arduino(MPU6050.h) Adaptación a Arduino Due 2.....	46
Arduino(MPU6050.h) FifoBuffer .....	46
Arduino(Final.ino) inicializarTodo() 1/2. ....	47
Arduino (Final.ino) Inicializartodo() 2/2 .....	48
Arduino(Calibrador.ino) Calibrador.ino 1/2 .....	48
Arduino(Calibrador.ino) Calibrador.ino 2/2. ....	49
Arduino(Final.ino) cogerDatos() .....	51
Arduino(MPU6050.cpp) setDeviceID().....	52
Arduino(Final.ino) posicion() 1/2. ....	53
Arduino (Final.ino) posición() 2/2 .....	54
Arduino(Final.ino) imprimir(). ....	55
Unity(Gui.cs) Comunicación Serie en Unity 1/2.....	55
Unity(Gui.cs) Comunicación Serie en Unity 2/2.....	56
Unity(*.cs) Script de Movimiento. ....	59
Unity(Gui.cs) Interfaz de usuario. ....	61
Unity(Gui.cs) Ver Datos .....	62
Unity(Gui.cs) Slider.....	63
Arduino(Receptor.ino) Receptor.ino.....	67

## 2. Resumen

### 2.1 Resumen en Castellano

El proyecto trata de un reconocimiento de movimiento corporal a través de sensores MPU6050 compuestos por un acelerómetro y un giróscopo, que se colocan en brazos y piernas. Estos sensores tienen un pequeño procesador que hace el tratamiento de los datos y los devuelve en forma de quaterniones.

Los datos se recogen con una tarjeta de Arduino por medio de una comunicación I2C y esta solo se encarga de redirigirlos a un ordenador a través de una comunicación en serie para que un software, a través de un cuerpo virtual, pueda reproducir los movimientos corporales.

### 2.2 Resumen en Ingles (Summary in English)

The project is based on recognition of body movement through sensors MPU6050 compounds by an accelerometer and a gyroscope, which are placed in the arms and legs. These sensors have a small processor that makes the processing of data and returns it in the form of quaternions.

The data are collected with an Arduino board through I2C communication and this board only deals redirect the data to a computer through a serial communication, and the movements are represented through a virtual body.

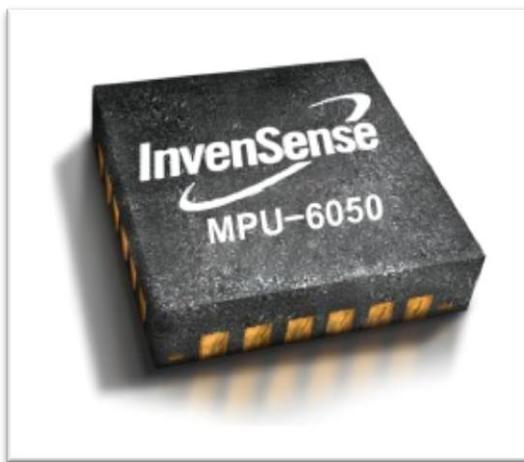
### 2.3 Palabras clave

MPU6050, giróscopo, Arduino, representación 3D, movimiento corporal.

### 3. Resumen extendido

El sistema empieza recogiendo información con una serie de sensores MPU6050 de la compañía Invensense (Figura 1), que están formados por:

- un acelerómetro
- un giróscopo
- un procesador (DMP)



**Figura 1.** Sensor MPU6050: Sensor de la compañía Invensense [1]

Estos sensores, por medio de unos velcros, se añaden a las diferentes extremidades del cuerpo. Se colocan en el brazo, antebrazo, la mano de cada brazo, el muslo, espinilla, un pie y en la espalda.

El procesador del MPU6050 está preprogramado para que calcule el quaternion del sensor y con una comunicación i2c y las calibraciones oportunas, le entregue ese valor a través de un buffer a una tarjeta de Arduino. El Fabricante apenas proporciona información sobre el funcionamiento de éste procesador.

El quaternion es un sistema de vectores que basándose en la información proporcionada por los acelerómetros y giróscopos es capaz de almacenar la información sobre la posición que se encuentra en sensor.

Los sensores están conectados a una tarjeta, que se acopla a la tarjeta Arduino DUE, diseñada para que a través de unos conmutadores con dos direcciones de I2C sea posible comunicarse con los 12 sensores utilizados.

La tarjeta Arduino Due es un microcontrolador de 32 bits programable, que dispone de varios sistemas de comunicaciones. Esta tarjeta es programada desde un ordenador para más tarde trabajar con autonomía propia.



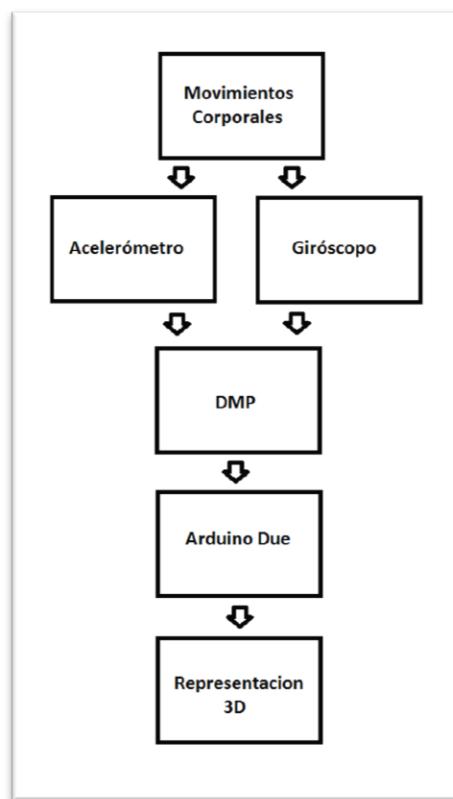
**Figura 2.** Cuerpo Virtual 3D: Imagen del software que reproduce los datos de los movimientos recogidos.

La tarjeta de Arduino dispone de unas librerías [MPU6050.h] que contienen la información con la que el microcontrolador es capaz de programar el DMP de los sensores y comunicarse con ellos para conseguir los datos de los quaterniones que estos ofrecen puedan ser leídos y procesados. Estos datos se colocan y se envían mediante comunicación serie a un software en un ordenador.

Una vez en el ordenador un software creado con Unity 3D [26], que es un programa para el diseño de videojuegos e interacción en entornos en 3D, recoge los datos enviados por la tarjeta Arduino, los ordena y estructura para que más tarde un cuerpo humano en 3D (Figura 2) sea capaz de reproducir de forma realista los datos.

El software también dispone de una colección de datos que se pueden imprimir en la pantalla representan los ángulos de cada una de las dimensiones de forma que además de ver la representación física de los movimientos puedes ver sus valores numéricos. También dispones de una barra de movimiento para poder mover el cuerpo de ángulo para poder ver con mayor comodidad algunos de los movimientos

Como se ve en el Diagrama 1 la información pasa por diversos procesos y dispositivos para llegar desde el cuerpo humano a la virtualización.



**Diagrama 1.** Camino de los datos:  
Representación gráfica del camino de los  
datos, desde la captura hasta la representación.

## 4. Memoria

### 4.1 Introducción

En la actualidad, la evolución tecnológica está teniendo un crecimiento a pasos agigantados. Una de las tendencias más importantes de estas evolución, es la intención de recrear el mundo físico en una forma virtual lo más realista posible. Existen mundos virtuales como SecondLife en los que existen mundos enteros en los que se crean sociedades autónomas o incluso los videojuegos en la actualidad recrean mundos virtuales completamente realistas.

Uno de los mayores problemas de los mundos virtuales es la interactuabilidad del usuario con la máquina, que por norma general es a través de un teclado, ratón o un joystick. En los últimos años están apareciendo diferentes modos de interacción. Los diferentes modos son:

- Captura de movimientos electromecánica
- Captura de movimientos electromagnética
- Captura óptica de movimiento
- Captura mediante Kinect
- Captura mediante fibra óptica
- Captura mediante sistemas inerciales

#### Captura de movimientos electromecánica

Los sistemas de captura de movimiento electromecánicos son aquellos sistemas en los que, en general, la captura de movimiento se realiza utilizando sensores mecánicos. [2]



**Figura 3.** Captura de movimiento electromecánica: Ejemplo de la captura de movimientos electromecánicos de una mano[2].

En el proceso de captura de movimientos, el actor o intérprete viste unos trajes especiales, adaptables al cuerpo humano. Los trajes son generalmente estructuras rígidas compuestas de barras metálicas o plásticas unidas mediante potenciómetros colocados en las principales articulaciones. El actor coloca la estructura en su cuerpo y mientras se mueve el traje se adapta a sus movimientos, y los potenciómetros recogen datos sobre el grado de apertura de las articulaciones. [2]

Uno de los mayores problemas de este método es que los trajes solo capturan el grado de apertura de las articulaciones y son incapaces de medir translaciones globales a no ser se le añada algún sensor electromagnético.

#### Capturas de movimientos electromagnéticos

En los sistemas de captura de movimiento electromagnético se dispone de una colección de sensores electromagnéticos que miden la relación espacial con un transmisor cercano. Los sensores se colocan en el cuerpo y se conectan a una unidad electrónica central, casi siempre mediante cables. Están constituidos por tres espiras ortogonales que miden el flujo magnético, determinando tanto la posición como la orientación del sensor.[2]



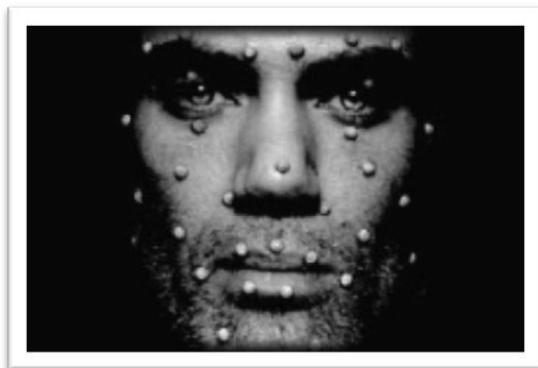
**Figura 4.** Captura de movimientos electromagnética:  
Sistema de captura de movimientos electromagnética  
comercial de la marca Patriot.[2]

Un transmisor genera un campo electromagnético de baja frecuencia que los receptores detectan y transmiten a la unidad electrónica de control, donde se filtra y amplifica. Después se envía a un ordenador central, donde se infiere la posición de todos los sensores en el espacio así como su orientación.[2]

El problema de este sistema es la renderización de los datos, en algunos casos el precio de la computadora de renderización supera el precio del sistema de captura.

### Captura Óptica de Movimiento

Los sistemas ópticos utilizan los datos recogidos por sensores de imagen para inferir la posición de un elemento en el espacio, utilizando una o más cámaras sincronizadas para proporcionar proyecciones simultáneas. Lo habitual es que los datos se recojan utilizando indicadores (markers) pegados al actor, pero los sistemas más recientes permiten recoger datos fiables rastreando superficies del sujeto identificadas dinámicamente. Estos sistemas producen datos con 3 grados de libertad para cada indicador; la orientación de una superficie se infiere utilizando la posición relativa de al menos 3 indicadores. [2]



**Figura 5.** Captura Óptica de Movimiento: Cara con sensores para la detección de movimiento mediante dispositivos ópticos. [2]

Los sistemas ópticos de captura de movimiento son, en general, métodos muy fiables para capturar determinados movimientos cuando se utilizan sistemas de última generación. Además, permiten la grabación en tiempo real, con ciertas limitaciones como el número de indicadores, el número de actores y de cámaras. [2]

Este tipo de dispositivos se utiliza sobretodo en cine y televisión ya que requiere de grandes estudios diáfanos donde poder trabajar.

### Captura mediante Kinect

Este método debería incluirse dentro de los sensores ópticos, pero en los últimos años este sistema de Microsoft ha revolucionado el mercado con este dispositivo.

El sensor de Kinect es una revolucionaria cámara nueva de profundidad que es usada en particular en la industria de los videojuegos para capturar los movimientos de las personas y los jugadores de manera eficiente, utilizando la tecnología de una cámara RGB y la cámara de

Kinect, normalmente, es usado para percepción en 3D de los movimientos humanos, sin embargo, sus usos pueden ir más allá, pues también se puede utilizar para aplicaciones robóticas. De entrada se ha liberado código abierto en LibFreenect para Linux, Microsoft mismo ha lanzado el Kinect SDK que básicamente usa Visual Studio 2012, y por su parte Google lanzó Robotic Operating System (ROS).[3]



**Figura 6.** Kinect: Descripción gráfica de las partes de un dispositivo Kinect.[3]

El problema es que solo funciona dentro del campo de visión de la cámara, aunque evita que el usuario tenga que cargar con ningún tipo de traje o dispositivo móvil.

#### Captura mediante fibra óptica

Los primeros sistemas de este estilo son los guantes de fibra óptica, pero en la actualmente se intenta aplicar esta técnica a la captura de movimientos del cuerpo entero. Los guantes de fibra óptica están constituidos por un conjunto de fibras ópticas que, al doblarse, atenúan la luz transmitida, permitiendo calcular la posición de los dedos de la mano. El primer ejemplo de sistema de este tipo es el Dataglove. [2]

Con estés sistema se puede medir la posición y la rotación de las articulaciones pero igual que el sistema electromecánico no es capaz de captar los movimientos globales de posicionamiento del cuerpo.

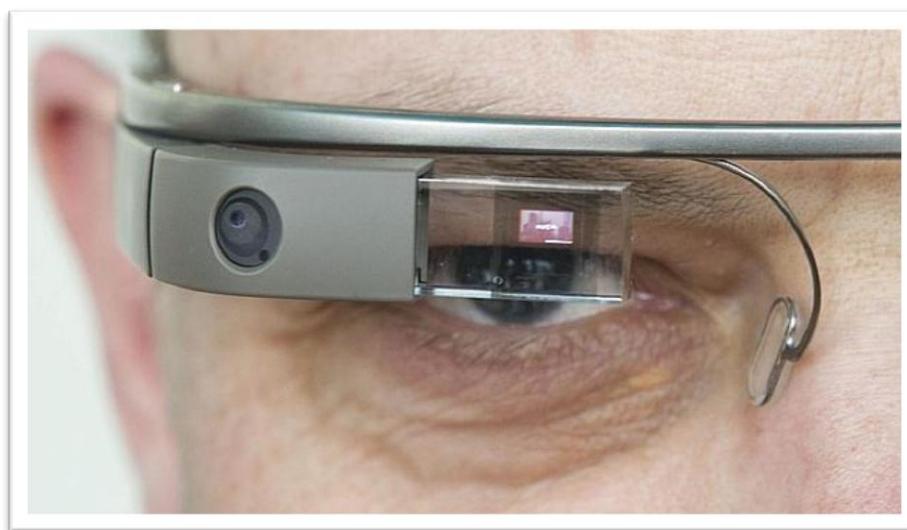


**Figura 7.** Guante de Fibra óptica: Guante de captura los movimientos de una mano mediante fibra óptica [2].

#### Captura mediante sistemas inerciales

Este sistema es sobre el que va a tratar el proyecto se basa en capturar el movimiento a través de acelerómetros y giróscopos. En la actualidad se está usando mucho también este método debido a su bajo precio.

En el mercado hay una gran cantidad de productos que utilizan este método como puede ser Nintendo Wii, cualquier Smartphone o incluso el último proyecto de Google, Google Glass, que por cierto usa un sensor MPU9150 que es similar al que se usa en este proyecto pero con brújula digital incluida.



**Figura 8.** Google Glass: Imagen del uso de las gafas Google Glass. [7]

Más adelante se detallaran los sistemas de funcionamiento de este método basado en acelerómetros y giróscopos y cómo éstos son capaces de traducir los datos físicos en datos digitales.

## 4.2 Base Teórica

El proyecto se basa en una serie de componentes y conocimientos teóricos que se van a explicar a continuación para hacer más entendible el funcionamiento del sistema.

### 4.2.1 Conceptos Mecánicos

#### Acelerómetro

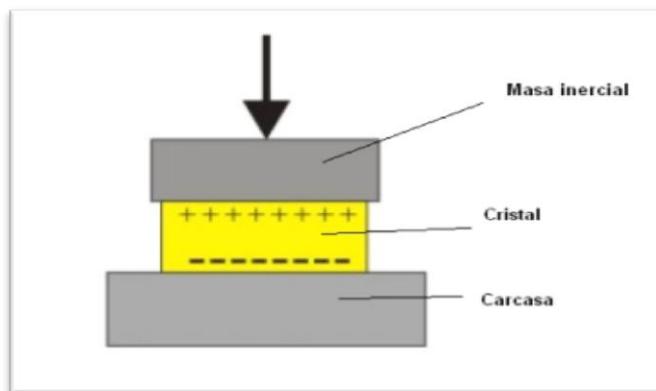
Como su propio nombre indica es un sensor que mide la aceleración que sufre un objeto, basándose en una masa inercial interna. Hace unos años solamente se usaban en entornos industriales pero la evolución tecnología ha hecho que su uso se expanda a muchos objetos de la vida cotidiana como pueden ser los Smartphone o videoconsolas.

El principio fundamental que aplica este dispositivo electrónico es la tan conocida segunda ley de Newton, la cual relaciona:

$$F = m * a$$

El tipo de acelerómetro más utilizado en sensores electrónicos se basa en las propiedades de los cristales piezoeléctricos. Estos cristales producen diferentes corrientes eléctricas según sean sometidos a algún tipo de fuerza como: compresión, flexión y extensión.

Poniendo este material entre el objeto o carcasa y una masa inercial se producirá una corriente cuando ocurra una aceleración. Midiendo esta corriente podremos calcular la aceleración, bien directamente si se trata de un acelerómetro de salida de corriente (coulombios/g) o bien convirtiéndola a un voltaje de baja impedancia si se trata de un acelerómetro de salida de voltaje. [4]



**Figura 9.** Acelerómetro: Funcionamiento interno de un acelerómetro piezo-eléctrico. [4]

En la actualidad debido al gran uso de este tipo de sensores, existen diferentes tipos de acelerómetros.

#### Principales tipos de Acelerómetros (clasificación según [5] )

- **Piezo-eléctricos:** Se basan en unos cristales Piezo-electricos que según su compresión o torsión producen una corriente eléctrica. (Figura 9)
- **Piezo-resistivo:** Es parecido al Piezo-eléctrico, pero en vez cristal utiliza un sustrato que varía su resistencia según la fuerza que se le aplique.
- **Galgas extensométricas:** Las galgas son unos puentes entre la carcasa y la masa inercial que sufren una deformación por la aceleración y estas producen variaciones en la corriente eléctrica
- **Térmico:** Se crea un núcleo de aire caliente, que con la aceleración es desplazado por el frío y con la diferencia de temperaturas se consigue sacar los valores.
- **Condensador:** el elemento que conecta la masa inercial con la carcasa es un condensador.

#### Giróscopo

El giróscopo es un sensor que permite conocer como varia un ángulo en el tiempo, mientras este este se encuentra rotando. Al girar genera una fuerza de rotación que hace que se estabilice.

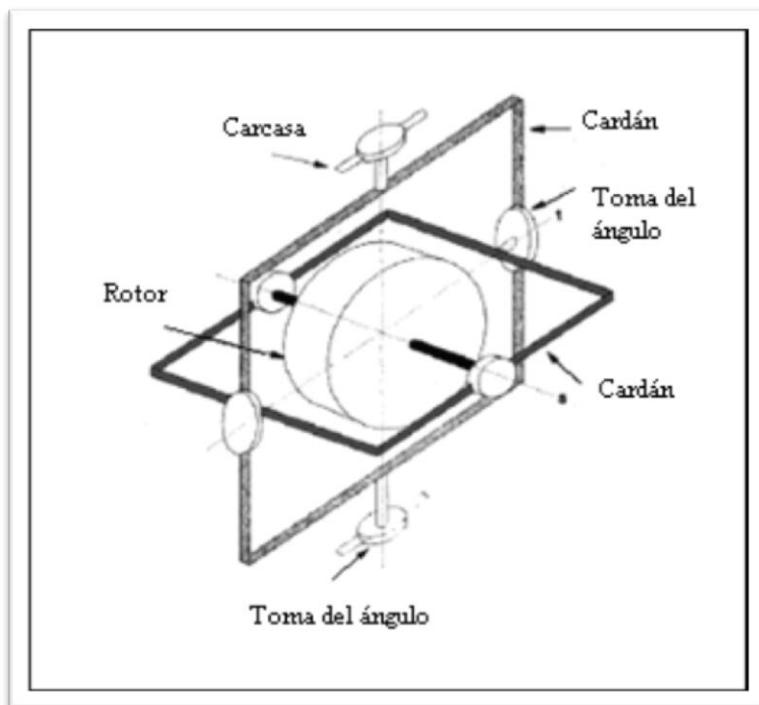


Figura 10. Giróscopo: sistema de funcionamiento de un giróscopo.[6]

Este principio utiliza una masa rotando sobre un eje sostenido por uno o varios cardanes (gimbals) dependiendo de los grados de libertad que se deseen, de tal manera que la precesión sea mínima, manteniendo así el eje estable y por tanto el giróscopo apuntara siempre en la misma dirección. Cuando se produce un movimiento en el sistema externo es posible observar el cambio en el ángulo. [6]



**Figura 11.** Giróscopo Invensense:  
muestra del tamaño de un giróscopo de  
la compañía Invensense.[1]

Aunque antiguamente los giróscopos eran de gran tamaño, y usarlos era un gran desafío, en la actualidad son de un tamaño muy pequeño (Figura 11) y la mayoría de las veces vienen acompañados de un acelerómetro. Por lo que cualquier dispositivo electrónico actual, por pequeño que sea, puede disponer de un giróscopo y controlar la posición en la que se encuentra.

Los sensores digitales como el del sensor MPU6050 funcionan con la tecnología MEMS (MicroEletroMecánica). Con esta tecnología lo que se permite es integrar en un chip de silicio pequeñas partes tridimensionales e incluso móviles. Lo que miden son variaciones en las vibraciones (que deberían ser estables) y también utilizan Coriolis para calcular sus variables.

El efecto coriolis es la aparente deflexión de la trayectoria de un objeto sobre la superficie de un cuerpo en rotación, a consecuencia de la rotación del cuerpo. El efecto es visible, por ejemplo, en las formas espirales de las tormentas terrestres y en las atmósferas de otros planetas.[8]

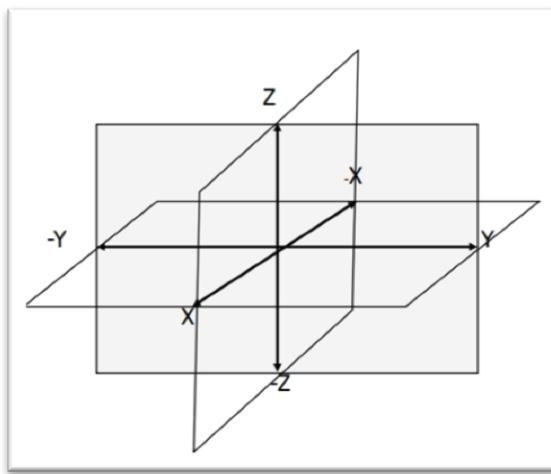


**Figura 12.** Giróscopo MEMS: Vista de un  
giróscopo MEMS desde un microscopio.  
[31]

#### 4.2.2 Conceptos Matemáticos

##### Sistema de coordenadas

Un sistema de coordenadas cartesianas se define por dos ejes ortogonales en un sistema bidimensional y tres ejes ortogonales en un sistema tridimensional, que se cortan en el origen O. Las coordenadas de un punto cualquiera vendrán dadas por las proyecciones del vector de posición del punto sobre cada uno de los ejes. [9]



**Figura 13.** Sistema coordenada: dibujo de planos en un sistema de coordenadas en 3D. [9]

##### Navegación inercial

Un sistema de navegación inercial es una forma de calcular la posición de un objeto a través de sensores en un sistema de coordenadas en 3D. Además de la posición se puede saber la inclinación y la velocidad del objeto.

Los sistemas de navegación iniciales (INS – Inercial Navigation System) se suelen utilizar en navegación marítima, aeronaves, misiles y naves espaciales, ya que un INS es capaz de detectar un cambio en la posición geográfica (un pequeño desplazamiento al norte o al este), un cambio en su velocidad (módulo y dirección) y un cambio en su orientación (rotación alrededor de un eje). Como este sistema no necesita una referencia externa (sólo inicialmente), es inmune a las interferencias que podría sufrir otro sistema, como el GPS.[10]

Los sistemas de navegación inercial, son el resultado del trabajo conjunto de los sensores iniciales ligados a una plataforma con un sistema de referencia común, mediante el cual se generan los datos a ser usados en la navegación inercial. [6]

Los principales sistemas de navegación inercial se pueden clasificar en:

- Sistemas Gimbaled
- Sistemas Strap-down

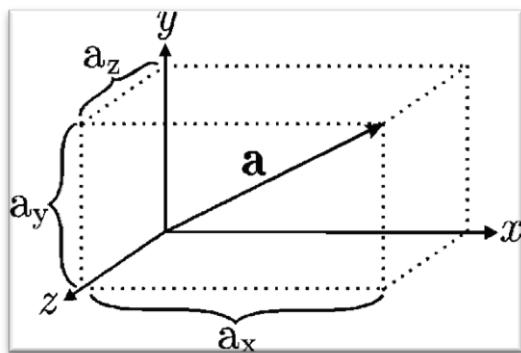
El sistema Gimbaled se basa en una plataforma rígida con un sistema mecánico inercial en el interior que aísla los movimientos exteriores del que se encuentra en el interior. Al tratarse de un sistema mecánico conlleva problemas como el peso, tamaño, etc.

El sistema Strap-down se basa en utilizar sensores digitales como acelerómetros y giróscopos que requieren de un procesamiento de datos para conseguir depurar unos datos con los que trabajar. Según [10].

### Vectores

En matemáticas, un vector describe una línea desde un punto inicial dado a otro punto en el espacio 3D. Esto significa que tiene dirección y que es finito.[12]

Un vector en un sistema de coordenadas en 3d está compuesto por la unión de dos puntos del espacio. Para ello son necesarios 6 valores, tres por cada punto, pero asumiendo, que uno de los puntos siempre es el (0,0,0), no se tendrá que representar, y con el valor del segundo punto será suficiente para representar el vector.



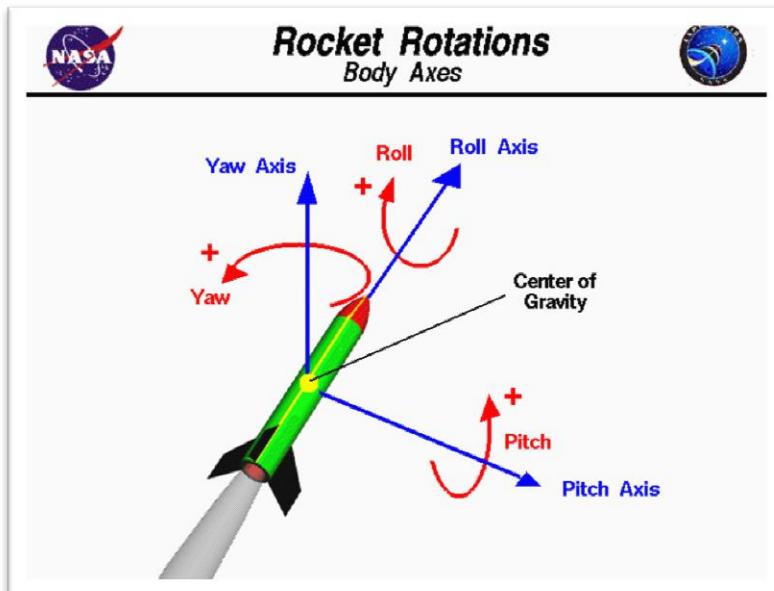
**Figura 14.** Vector: representación gráfica de un vector. [13]

Cuando un vector es normalizado, el vector es escalado a una longitud de 1. Se usan vectores normalizados o vectores unidad cuando se desea describir solo una dirección. Esto es necesario para obtener los resultados deseados con algunos algoritmos matemáticos. [12]

## Ángulos de Euler

Los ángulos de Euler es el sistema de rotación más utilizado en los métodos de rotación de objetos en sistemas en 3D. Se utilizan los valores: *yaw* (guiñada), *pitch* (cabeceo) y *roll* (balanceo).

- *Yaw* es la rotación en el plano XY anclado sobre el eje y, se podría decir que es girar de izquierda a derecha o viceversa.
- *Pitch* se le llama a los movimientos en el plano YZ en el que el objeto está anclado en el eje X, y los se podría decir que son de arriba a abajo y viceversa.
- *Roll* es el movimiento de rotación del objeto sobre el plano XY anclado en el eje Z, y como ya hemos dicho seria la rotación sobre sí mismo.



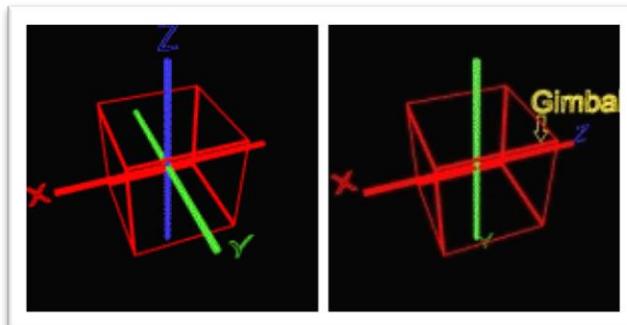
**Figura 15.** Ángulos de Euler: representación gráfica de los movimientos de un cohete de la Nasa mediante los ángulos de Euler.[14]

Uno de los mayores problemas que tiene la rotación de objetos en un sistema de coordenadas 3D por el método de los ángulos de Euler es el error de bloqueo de los ejes, más conocido por gimbal lock o suspensión de Cardan.

## Gimbal Lock

El Gimbal Lock o suspensión de Cardan es cuando dos ejes coinciden apuntando a la misma dirección y por lo tanto el tercer eje coincide girando en la misma dirección con los dos anteriores bloqueados.

Cualquier sistema que utiliza ángulos Euler tendrá problemas con el bloqueo de ejes. La razón de esto es que los ángulos Euler evalúan cada eje de forma independiente en un orden establecido. El orden es por lo general X, Y, Z (no siempre es este orden). Primero el objeto se desplaza hacia abajo en el eje X. Cuando el termina, entonces se desplaza hacia abajo en el eje Y, y finalmente en el eje Z. [15]



**Figura 16.** Gimbal Lock: Ejemplo de bloqueo de ejes cuando gira Y se juntan X y Z. [15]

El problema con el bloqueo de ejes se produce cuando se gira el objeto por el eje Y 90 grados. Dado que el componente X ya se ha evaluado y no se llevó junto con los otros dos ejes. Lo que termina pasando es el X y el eje Z consigue señalar el mismo eje. [15]

Es decir, cuando dos ejes alinean, ejercen el mismo movimiento que el tercer eje, por lo que los datos pueden ser confusos y erróneos.

### Quaternion

Los Quaterniones fueron inventados en 1893 por el matemático William Rowan Hamilton con la idea de representar vectores con números imaginarios, dedicó más de 20 años a estudiarlos, y aunque hasta hace no mucho tiempo no tuvieron demasiada importancia, en la actualidad son de gran utilidad.

Los quaterniones son una extensión de los números reales, similar a la de los números complejos. Mientras que los números complejos son una extensión de los reales por la adición de la unidad imaginaria  $i$ , tal que  $i^2 = -1$ , los quaterniones son una extensión generada de manera análoga añadiendo las unidades imaginarias:  $i, j$  y  $k$  a los números reales y tal que [10] [11]

$$i^2 = j^2 = k^2 = ijk = -1$$

Un quaternion es un número de la forma:

$$q = a + a_1 i + a_2 j + a_3 k$$

$a, a_1, a_2, a_3$  se denomina parte real o parte escalar y  $i, j$  y  $k$  son la parte imaginaria.

La complejidad que presentan las rotaciones en el espacio tridimensional se debe a su no conmutatividad. Esta propiedad implica que no pueden ser tratadas como vectores. Por tanto, el conjunto de todas las rotaciones tridimensionales no están organizadas como un espacio vectorial tridimensional, sino como un conjunto de superficies curvas y cerradas que envuelven una esfera, que es la superficie directriz. Este grupo es denominado SO3, 'Special Orthogonal tridimensional group'. Los quaterniones unidad,  $S^3$ , tienen la capacidad de capturar toda la geometría, topología, y estructura de las rotaciones tridimensionales, en la forma más sencilla posible. [16]

Podemos recoger en tres teoremas la relación entre los quaternion y las rotaciones tridimensionales: Sea  $p$  un punto en el espacio proyectivo tridimensional representado como un quaternion, de forma que  $p=[v,w]=[x,y,z,w]$ . [16]

Sea  $q$  cualquier quaternion distinto de cero.

- El producto  $(qpq^{-1})$  transforma a  $p=[v,w]$  en  $p'=[v',w]$ , siendo los módulos de  $v$  y  $v'$  iguales.
- Cualquier múltiplo de  $q$  distinto de cero actúa de la misma manera sobre  $p$ .
- Si  $q$  es un quaternion unitario, tal que  $N(q)=1$ , entonces  $q=[v^*\sin\theta, \cos\theta]$  actúa sobre  $p$  rotándolo un ángulo de  $2\theta$  alrededor del eje con vector director unidad  $v^*$ .

Ahora bien, para representar una rotación con angulo  $\Theta$  alrededor del vector unitario  $v$ , el quaternion resultante es [32] :

$$\hat{v} = [\hat{v}_1, \hat{v}_2, \hat{v}_3]$$

$$h = (w, v) = \left( \cos \frac{\theta}{2}, \hat{v} \sin \frac{\theta}{2} \right)$$

$$h = \cos \frac{\theta}{2} + \left( \hat{v}_1 \sin \frac{\theta}{2} \right) i + \left( \hat{v}_2 \sin \frac{\theta}{2} \right) j + \left( \hat{v}_3 \sin \frac{\theta}{2} \right) k$$

En donde:

$$w = \cos \frac{\theta}{2}$$

$$x = \hat{v}_1 \sin \frac{\theta}{2}$$

$$y = \hat{v}_2 \sin \frac{\theta}{2}$$

$$z = \hat{v}_3 \sin \frac{\theta}{2}$$

Los quaterniones de esta forma describen una rotación que puede representarse mediante una matriz. Esta matriz se construye a partir de los componentes de un quaternion de la siguiente forma:

$$R(h) = \begin{bmatrix} 2(w^2 + x^2) - 1 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 2(w^2 + y^2) - 1 & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 2(w^2 + z^2) - 1 \end{bmatrix}$$

Las dos ventajas más importantes que introducen los quaterniones son la simplificación de la matriz y evitar el error del Gimbal Lock. Simplificamos la matriz ya que no hace falta trabajar con los 9 valores de la matriz ya que se sacan del quaternion y este solo son 4. Y La segunda nos ayuda a evitar los errores que puede producir el Gimball Lock en los ángulos de Euler.

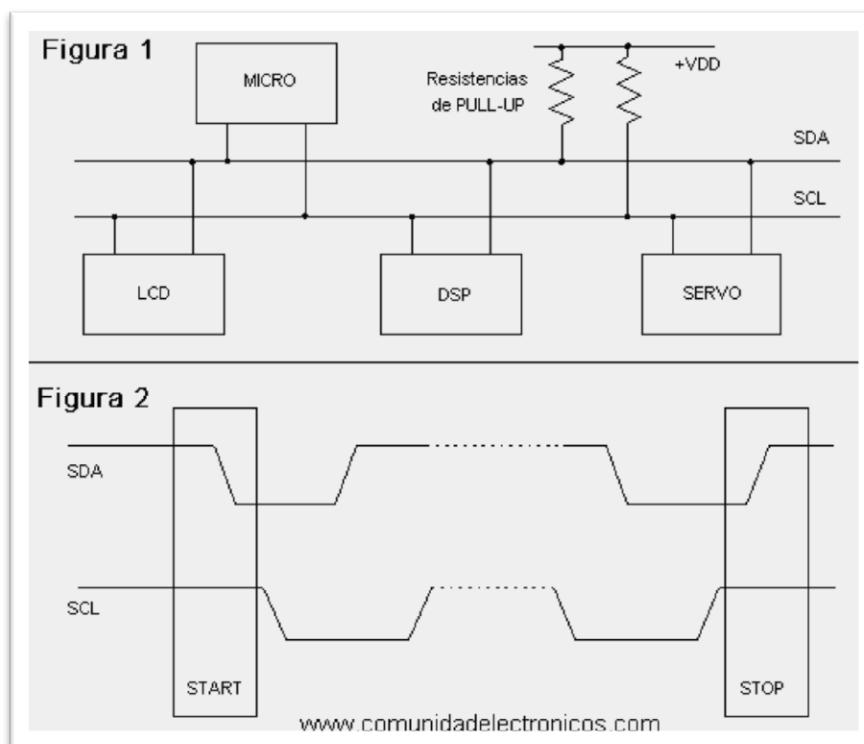
#### 4.2.3 Tipos de comunicación

##### Comunicación I2C

Uno de los protocolos de comunicación más utilizados entre un microcontrolador y un periférico es el protocolo I2C. Las características más salientes del bus I2C según [17] son:

- Utiliza la linea de datos (SDA) y la de reloj (SCL).
- Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software.
- El protocolo de comunicación entre el microcontrolador y los dispositivos es Maestro/Eslavo.
- El bus incluye un detector de colisiones que permite la conexión de varios Maestros.
- El protocolo de transferencia de datos y direcciones posibilita diseñar sistemas completamente definidos por software.

- Los datos y direcciones se transmiten con palabras de 8 bits.



**Figura 17.** Comunicación I<sub>2</sub>C: en la figura 1 vemos el esquema de conexiones de la comunicación y en la figura 2 vemos la comunicación de Start y Stop entre maestro y esclavo.

Toda conexión I<sub>2</sub>C tiene un comienzo y un fin que se define con Start y Stop. Para que el Maestro informe al Esclavo de que la comunicación puede comenzar, coloca la línea de reloj SCL en nivel alto y deja caer la línea de datos SDA a 0. A continuación se realiza toda la comunicación hasta que se decide cerrar la comunicación y la línea de datos SDA pasa a nivel alto mientras que la línea de reloj SCL sigue estando en estado alto como se ve en la Figura 17.

Entre el Start y el Stop se produce la comunicación. Esta comunicación se realiza con palabras de 8 bits que se depositan en la línea de datos SDA. El Maestro comienza colocando la dirección del esclavo con el que se quiere comunicar, éste es leído por el Esclavo y escribe un 0 en la línea en forma de ACK (acknowledge), confirmando su disposición para recibir los datos, si el master en vez de un 0 lee un 1 significa que la comunicación no ha sido posible con el Esclavo por lo que haría un Stop y liberaría el bus de datos.

### Resistencias PULL UP

Los resistores pull-up se encargan de que las entradas lógicas del sistema se mantengan en los niveles correctos en caso de que otros dispositivos se desconecten del sistema. Las resistencias pull-up "tiran" la tensión del circuito hasta una tensión precalculada. Este tipo de resistencias suelen ser débiles, sin embargo, y si otro dispositivo tira la tensión del circuito a otro voltaje, el resistor no resistirá. [18] La función principal de una resistencia pull-up en I2C es fijar un nivel alto en el bus cuando ninguno de los dispositivos fija un nivel bajo en su salida en colector abierto.

### Comunicación Serie

Junto con el I2C la comunicación en serie es uno de los protocolos de comunicación más comúnmente utilizados en la actualidad entre dispositivos. A la hora de trabajar con dispositivos electrónicos como Arduino y querer comunicarlos con un ordenador casi siempre se utiliza este tipo de comunicación ya que se puede usar a través de cables USB o dispositivos Bluetooth.

El concepto de comunicación serie es sencillo, el puerto serie envía y recibe bytes de información bit a bit. Esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo a la vez. Este método de comunicación es más sencillo y puede alcanzar mayores distancias. Por ejemplo, la especificación *IEEE 488* para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre cualesquier dos dispositivos; por el otro lado, utilizando comunicación serie el largo del cable puede llegar a los 1200 metros. [19]

Normalmente los datos que se transmiten en este protocolo suelen ser en formato ASCII y se utilizan 3 líneas para que sea posible la transmisión: tierra, de transmisión y para recibir. Como la comunicación es asíncrona es posible que mientras se está enviando información por una línea se pueda estar recibiendo por la otra al mismo tiempo. Este tipo de comunicación dispone de una serie de características a tener en cuenta:

- Velocidad de transmisión: Número de bits que se transfieren por segundo, la unidad de medida son los baudios (baud rate). Cuanto mayor sea la frecuencia de muestreo más corta debe ser la distancia entre los dispositivos.
- Bits de datos: Número de bits que mandan en la transmisión, las cantidades más comunes son de 5, 7 y 8 bits. Dependiendo de si se utiliza el protocolo ASCII estándar que tiene un rango de 0 a 127 se utilizaría solo 7 bits y si por el contrario se utilizaría el protocolo ASCII extendido que cubre un rango de 0 a 255 necesitaríamos 8 bits. Por ejemplo en Arduino si envías un objeto de tipo de Byte utilizas 8 bits por paquete. Un paquete es toda la información necesaria que se manda cuando envías un dato.
- Bits de parada: Se utilizan para señalizar al receptor el final del paquete enviado. Cada dispositivo en este sistema de comunicación utiliza un reloj propio, por lo que normalmente no están sincronizados así que estos bits también sirven para dar un

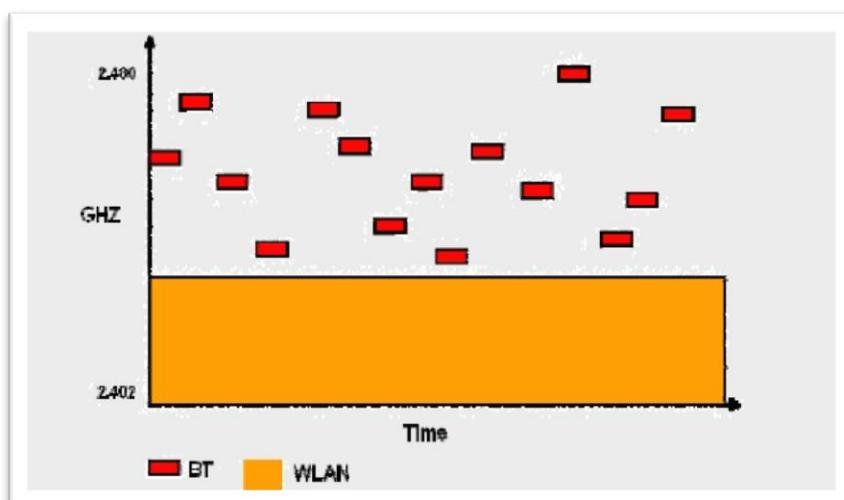
margen de tolerancia para esa diferencia entre los relojes. Cuantos más bits de parada mayor tolerancia pero también se reduce la velocidad.

- **Paridad:** Es una forma sencilla de verificar si hay errores en la transmisión serie. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serie fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados. [19]

### Bluetooth

La tecnología bluetooth tiene tres características principales que la definen:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre estos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.



**Figura 18.** Bluetooth: Sistema de saltos en la comunicación bluetooth. [20]

El bluetooth permite conexiones punto a punto y punto a multipunto. En la comunicación se pueden diferenciar dos perfiles: Maestro que controla el tráfico de datos y emite en las ranuras de tiempo pares y Esclavo que recibe las señales del maestro y emite en las ranuras de tiempo impares.

Los Dispositivos Bluetooth utilizan toda la banda de 2,4 GHz. Si una transmisión se interfiere sobre un canal, una retransmisión siempre ocurrirá sobre un canal diferente con la esperanza de que este canal esté libre. Cada ranura de tiempo tiene una duración de 625 microsegundos y generalmente los dispositivos saltan una vez por paquete. [20]

Utiliza el Salto de frecuencia adaptativo (AFH). Identifica fuentes fijas de interferencia, se va actualizando una lista con los canales disponibles. El bluetooth necesita un mínimo de 20 canales para funcionar.

Las principales medidas de seguridad que utiliza la comunicación bluetooth se podría diferenciar en dos partes: previos a la comunicación (autenticación y Pairing) y durante la comunicación (Encriptación y Generación de una clave de sesión).[20]

- Autenticación: Comprobar que el dispositivo es quien dice ser.
- Pairing: Comprobar y liberar la conexión entre dos aparatos.
- Encriptación.
- Generación de una clave de sesión.

#### 4.2.4 Materiales

##### Arduino DUE

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.[21] En la actualidad Arduino ha tenido un gran auge dentro del mundo de la electrónica debido a su sencillez y precio reducido. Puede ayudar a crear grandes proyectos sin tener un gran conocimiento en electrónica.

En 2013 ha salido el modelo Arduino Due, que es una placa basada en el procesador Atmel SAM3X8E que dispone de un núcleo ARM Cortex-M3 de 32 bits. Las principales características de este procesador son [22] :

- Un núcleo de 32 bits, que permite operaciones en 4 bytes de datos de ancho dentro de un único reloj de la CPU.
- Reloj de la CPU en 84Mhz.
- 96 Kbyte de SRAM.
- 512 Kbyte de memoria flash para el código.
- Un controlador de DMA, que puede aliviar a la CPU de realizar tareas que requieren mucha memoria.

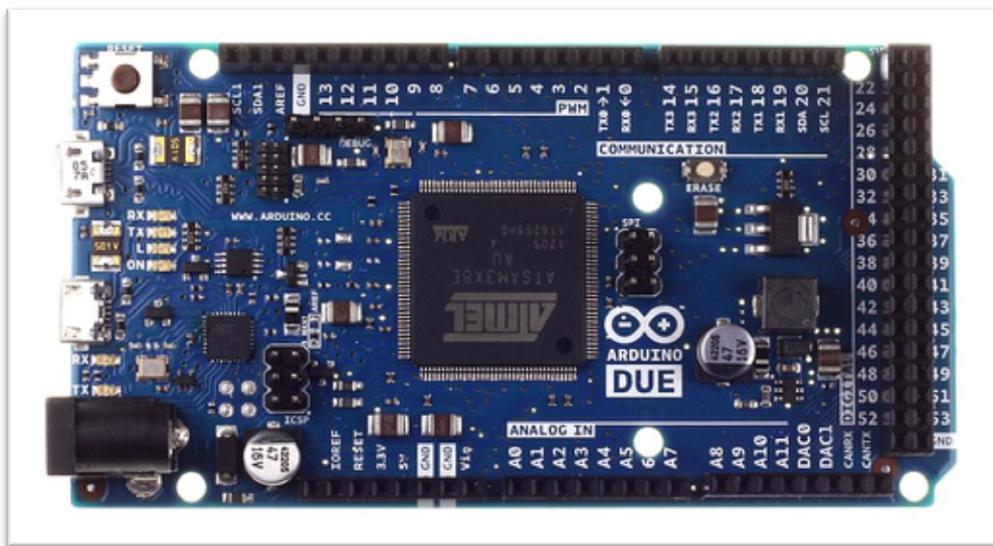


Figura 19. Arduino DUE : Imagen de la tarjeta Arduino Due. [22]

Dispone de 54 pines digitales que pueden ser usados tanto de entrada como de salida funcionando a 3,3 voltios. Estos pines pueden incluir funciones como:

- Comunicación serie (RX y TX).
- PWM.
- SPI.
- Comunicación CAN.
- Comunicación USB.
- Comunicación I2C

También dispone de otros pines para entradas analógicas (8 pines) y otros para algunas funciones de control más específicas.

Arduino se conecta al ordenador mediante USB y el puerto “programing port” para ser programado desde un software. Y también dispone de un puerto nativo para conectar diferentes dispositivos.

### Sensor MPU6050

La familia MPU-6000/MPU-6050™ son parte de los primeros dispositivos en el mundo de 6 ejes de bajo consumo, bajo coste y un alto rendimiento utilizados en Smartphone, tablets y sensores utilizados en prendas de vestir. [1]



**Figura 20.** Sensor MPU6050: Sensor de la compañía Invensense. [1]

Los sensores MPU6050 combinan un giróscopo de 3 ejes y un acelerómetro de 3 ejes en una matriz de silicio junto con un pequeño procesador digital de movimiento (DMP).

Las señales del sensor se recogen mediante una comunicación I2C a través del cual manda la información e incluso dispone de otra salida i2c mediante la que puede trabajar como Maestro para recoger datos de un segundo dispositivo, procesar los datos de este y enviar de nuevo la información la comunicación I2C principal. El sensor dispone de una entrada AD0 que según el valor puede hacer cambiar la dirección de 0x68 a 0x69 del sensor en la comunicación I2C.



**Diagrama 2.** Diagrama del Sensor MPU6050/MPU6000: Funcionamiento de la comunicación del sensor. [1]

Para el seguimiento preciso de movimientos rápidos y lentos, las partes cuentan con un rango de escala del giróscopo programable por el usuario de  $\pm 250$ ,  $\pm 500$ ,  $\pm 1.000$  y  $\pm 2.000^\circ$  / segundo (dps) y un acelerómetro programable por el usuario de  $\pm 2\text{ g}$ ,  $\pm 4\text{ g}$ ,  $\pm 8\text{ g}$ , y  $\pm 16\text{ g}$ .[1]

Para ahorrar espacio, el tamaño del paquete de los dispositivos ha sido impulsado hacia abajo y mide 4x4x0.9mm (QFN). Las características adicionales incluyen un sensor de temperatura integrado y un oscilador on-chip con  $\pm 1\%$  de variación en el rango de temperatura de funcionamiento. [1]

En la Tabla 1 se ven las características principales de este sensor.

Parte #	Gyro completo rango de escala	Sensibilidad Gyro	Accel completo rango de escala	Accel Sensibilidad	Salida digital	Voltaje de fuente lógica	Voltaje de alimentación operativo	Tamaño del paquete
UNIDADES:	(° / seg)	(LSB / ° / seg)	(G)	(LSB / g)		(V)	(V + / - 5%)	(Mm)
 MPU-6000	± 250	131	± 2	16384	I <sup>2</sup> C y SPI	VDD	2.375V-3.46V	4x4x0.9
	± 500	65,5	± 4	8192				
	± 1.000	32,8	± 8	4096				
	± 2.000	16,4	16 ±	dos mil cuarenta y ocho				
 MPU-6050	± 250	131	± 2	16384	I <sup>2</sup> C o VDD	1.8V ± 5% o VDD	2.375V-3.46V	4x4x0.9
	± 500	65,5	± 4	8192				
	± 1.000	32,8	± 8	4096				
	± 2.000	16,4	16 ±	dos mil cuarenta y ocho				

Tabla 1. Sensores MPU6000/ MPU6050: Características. [1]

El mayor problema de estos sensores es su programación es el no tener acceso al firmware interno del DMP. Según la compañía, si se utiliza su tarjeta para controlar el dispositivo se pueden realizar cálculos con el DMP del sensor para conseguir algunos datos como:

- Calculo del Quaternion de movimiento
- Calculo de caída libre
- Calculo de contador de pasos
- Etc.

Pero si se utiliza una tarjeta como Arduino, como ya se ha comentado antes, la empresa no da más opción que los valores crudos pero gracias a las librerías de Ingeniería Inversa de Jeff Rowberg [30] se pueden realizar estas acciones incluso algunas diferentes que el software libre nos ofrece.

Toda esta información del procesador DMP del sensor MPU6050 es depositada en un bus FIFO de la que el procesador principal coge la información.

En resumen, este sensor tiene un gran abanico de posibilidades gracias a su tamaño y a su gran fiabilidad de datos y de procesamiento pero el hecho de que no se pueda programar con total libertad del programador el procesador DMP de este sensor hace que sus posibilidades se debiliten de gran forma.

### Bluetooth HC-05

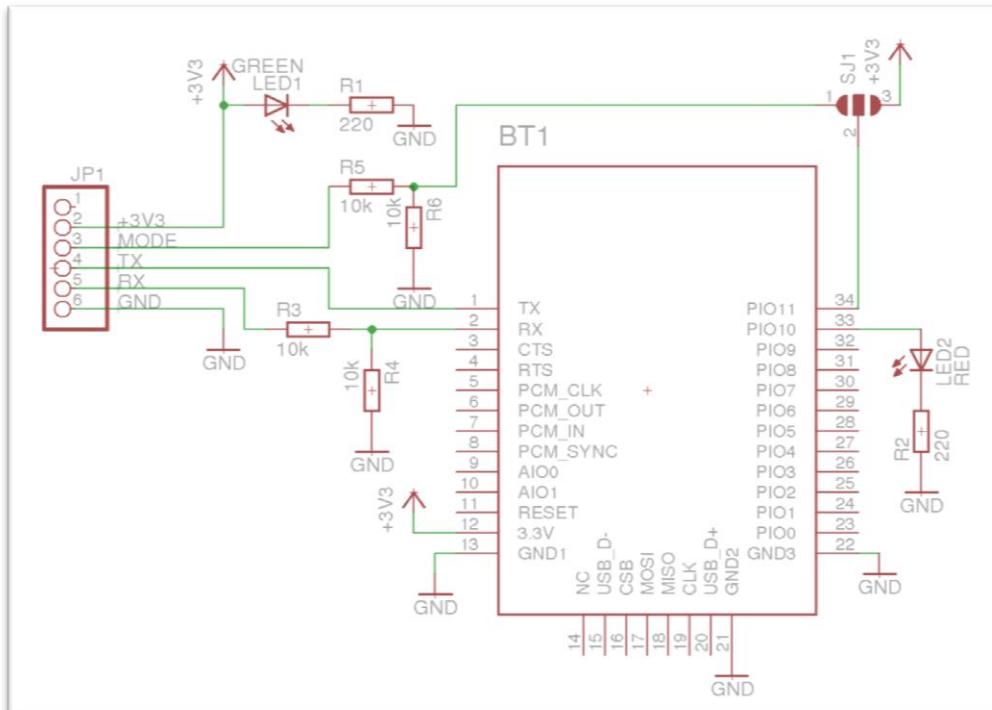
El dispositivo de bluetooth HC-05 es un módulo que trabaja con el protocolo de comunicación serie y puede trabajar tanto de Maestro como de Esclavo.



**Figura 21.** Bluetooth Depeca: Modulo HC-05 junto a la tarjeta de adaptación del Departamento de Electrónica de la UAH. [23]

Dispone de una configuración programable en la que se puede modificar desde el nombre del dispositivo hasta la velocidad de transmisión.

Como soporte para el dispositivo Bluetooth HC-05 se ha utilizado una placa del Departamento de Electrónica de la Universidad de Alcalá de Henares [23] que facilita la comunicación con Arduino. La placa principalmente conduce la información a enviar por el dispositivo desde el procesador al módulo HC-05. El Diagrama 3 muestra las conexiones de ambas tarjetas.



**Diagrama 3.** Bluetooth: Funcionamiento del bluetooth. [23]

### Comutador 74HC4052

El dispositivo 74HC4052 dispone de dos entradas binarias de datos que por medio de otros dos bits de selección reparte cada una de estas señales entre cuatro posibles salidas. De esta forma la misma señal puede ser repartida entre cuatro dispositivos diferentes en el momento en el que se le indique. El Diagrama 4 se muestra el funcionamiento de este módulo.

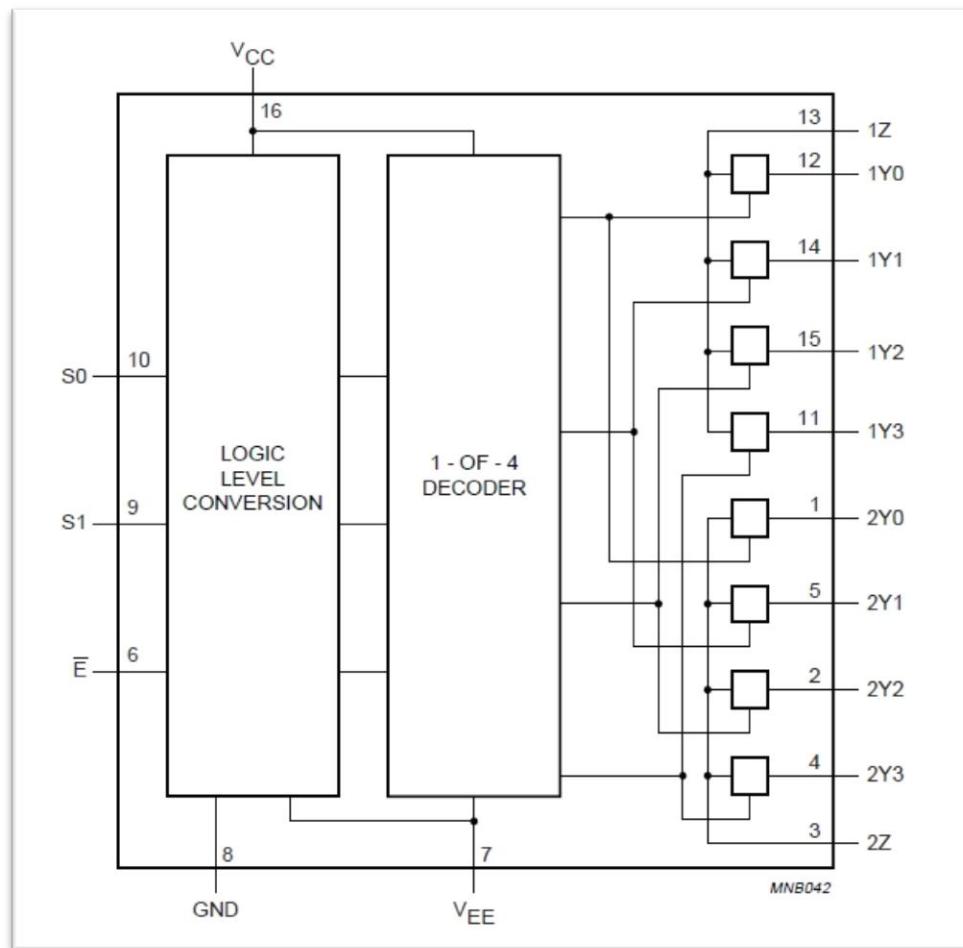


Diagrama 4. Módulo 74HC4052: Funcionamiento de la conmutación del dispositivo.[24]

### Batería Lipo 7,4v 1000mA 2S

Las características principales de las baterías lipo se podría decir que son su ligereza y la capacidad energética junto al poco efecto memoria que sufren pudiendo así alargar la vida de éstas. Su mayor problema es la poca fiabilidad en situaciones extremas como las altas

temperaturas o la sobrecarga que podrían producir una inestabilidad que llegaría a hacerla explotar de forma violenta.



**Figura 22.** Batería Lipo: Batería lipo de 7,4v 1000mah 2C. [25]

Esta batería en especial dispone de un voltaje de 7,4 con una capacidad de carga de 1000mA por hora. La batería está dividida en dos celdas para poder hacer un uso más regular de la carga pudiendo nivelar la carga de ambas celdas.

#### Shield de extensión I2C

Se ha creado un shield adaptable en Arduino para la ocasión con la intención de ampliar las direcciones de señales I2C del proyecto.

La placa recoge las señales I2C de una Arduino Due (SDA 20 y SCL 21) y las dirige a la entrada de señal de dos conmutadores analógicos 74HC4052. Mediante dos pines que seleccionan la señal (Pin 8 y Pin 9) y otros dos que seleccionan en conmutador (Pin 10 y pin 11) la señal es dirigida a uno de las doce salidas disponibles (C1..C12) las cuales dispone de un conector en el que conectar el cable que llegue a cada dispositivo MPU6050

Como los sensores MPU6050 pueden tener varias direcciones (0x68 y 0x69) la misma señal llega a dos salidas a la vez por lo que estas están conectadas:

- C1 con C2
- C3 con C4
- C5 con C6
- C7 con C8
- C9 con C11
- C10 con C12

Se debe poner especial atención a que las dos últimas conexiones no son consecutivas. Las señales SDA y SCL disponen de dos resistencias Pull-UP con dos Jumpers para la posibilidad de activarlas o desactivarlas según necesite el proyecto. Como se ve en el Diagrama 9.

Cada Salida dispone en la placa de la posibilidad de conectarle un condensador para asegurar la carga eléctrica necesaria para su correcto funcionamiento.

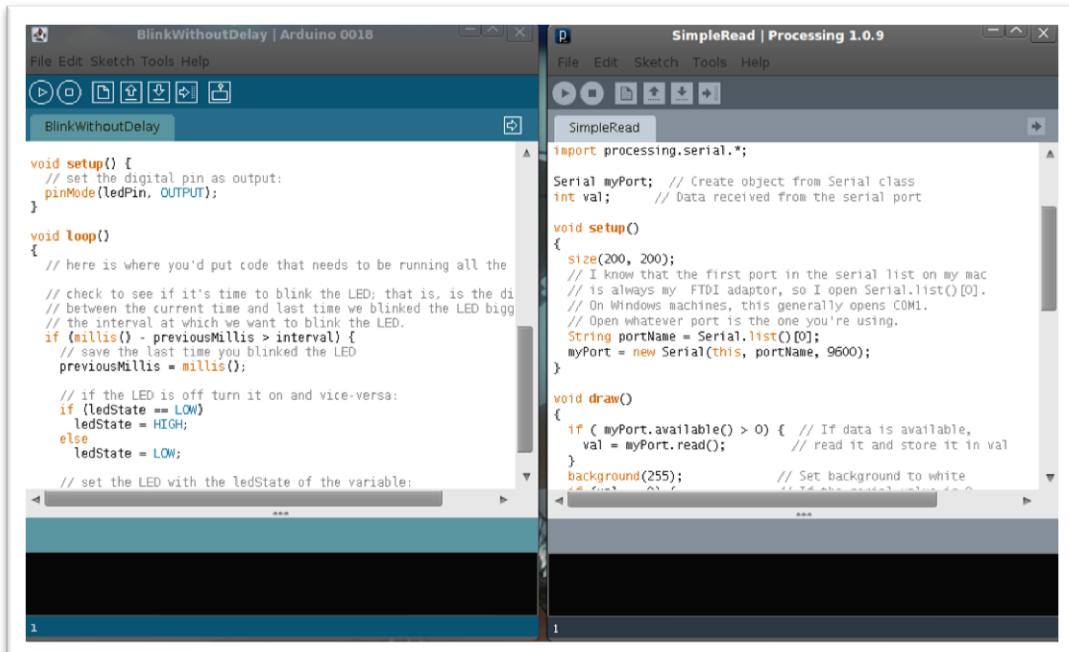
La shield dispone de 12 leds de color rojo utilizados para por confirmar, vía software, que las conexiones I2C de cada salida están realmente enlazados y en caso de fallo saber cuál es el sensor que ha fallado en la conexión.

Por ultimo se dispone de la posibilidad de utilizar la comunicación serie de Arduino Due (Rx y Tx) mediante una conexión que puede ser utilizado por un módulo bluetooth o por otros dispositivos de comunicación en serie. El orden de los pines es: Vcc, Mode, Tx, Rx y Gnd.

#### 4.2.5 Software

##### IDE Arduino

El entorno de código abierto Arduino hace fácil escribir código y cargarlo a la placa E/S. Funciona en Windows, Mac OS X y Linux. El entorno está escrito en Java y basado en Processing, avr-gcc y otros programas también de código abierto. [21]



**Figura 23.** Arduino vs Processing: Ambos IDE's de programación tienen cierto parecido.

El lenguaje de programación está orientado en C, y aunque no es exactamente igual, tiene bastante parecido. Los creadores de este IDE se orientaron en el entorno de programación de Processing ya que la interfaz gráfica es casi idéntica y otras funciones aunque no son iguales organizadas de forma muy parecida.

Este software viene con una serie de ejemplos prácticos para iniciarse en el mundo de Arduino, desde el programa necesario para encender un LED (Blink) hasta librerías para poder trabajar con la comunicación I2C desde un nivel de abstracción que hace sencilla su implementación.

Arduino dispone de dos vertientes diferentes de software para su programación en este momento

- Arduino IDE 0019
- Arduino IDE 1.5.1

Una de las diferencias más importantes y por la que este proyecto trabaja con 1.5.1 es que acepta trabajar con Arduino Due mientras que la vertiente 0019 no dispone de esa característica.

### Unity 3D

Unity es un ecosistema de desarrollo de juegos: un potente motor de renderizado totalmente integrado con juego completo de herramientas intuitivas y flujos de trabajo rápidos para crear contenido 3D interactivo, publicación multiplataforma fácil (Publicación para IOS, Android, Mac, Etc pulsando un botón) y miles de activos de calidad listos para usar en la Tienda de Activos en la que puedes comprar desde escenarios hasta condiciones meteorológicas programadas para añadir a las creaciones. [26]



**Figura 24.** Unity 3D: Entorno de trabajo del software de videojuegos Unity 3D.

Dispone de una gran variedad de posibilidades en las que desarrolladores independientes sin grandes recursos puedan crear videojuegos de un alto nivel tanto gráfico como en jugabilidad.

Los flujos de trabajo en Unity son sencillos e intuitivos, dispone de una gran cantidad de herramientas que facilitan su uso. Todo su trabajo funciona a través de Scripts tanto en C con java para dar más amplitud a la gama de desarrolladores que puedan trabajar en el mismo proyecto.

Unity busca facilitar el máximo realismo en los videojuegos poniendo a disposición de los desarrolladores un repositorio de efectos visuales que incluyen desde rayos de sol, explosiones, lluvias de estrellas junto con un motor físico que intenta fidelizar al máximo la realidad en cuestión de pesos, viento, etc.

Dispone de la herramienta Profiler, que muestra el rendimiento y el consumo de recursos que produce el videojuego pudiendo analizar donde se encuentran los cuellos de botella y poder solucionarlos de forma que el videojuego corra con mayor fluidez. Una de las posibles opciones para arreglar este tipo de problemas que producen los cuellos de botella es renderizar solo lo que interese en un momento adecuado, crear diferentes capas de detalles o incluso bajar el nivel de detalle según el dispositivo en el que se vaya a usar la aplicación.

Mecanim, singularmente potente y flexible sistema de animación de Unity, trae personajes no humanos a la vida humana y con un movimiento muy natural y fluido. Es una solución completa para la animación en los juegos. Se elimina la necesidad de gastar esfuerzos de desarrollo caros para integrar aplicaciones de terceros. Mecanim está integrado de forma nativa y optimizada para ejecutarse en el motor de Unity. Desde el Editor, obtendrá todas las herramientas y flujos de trabajo necesarios para crear y construir músculo, clips, árboles, máquinas de estado y controladores directamente en Unity. [26]



**Figura 25.** Plataformas de Unity 3D: Menú de selección de plataforma del software de videojuegos Unity 3D.

Unity ofrece la posibilidad de desarrollar proyectos para diversos dispositivos, el diseño es prácticamente igual para todos, con la diferencia de que a la hora de compilar seleccionas la plataforma que se desee. Importantes juegos para dispositivos móviles como Temple Run en IOS7 de iPhone o Bad Piggies de Rovio para Android han sido creados en esta plataforma.

### MonoDevelop

MonoDevelop es un entorno de desarrollo multiplataforma diseñado principalmente para “C #”, “.NET” y otros. MonoDevelop permite a los desarrolladores escribir rápidamente aplicaciones de escritorio y Web ASP .NET en Linux, Windows y Mac OSX. [27]

Es un entorno de programación simple y efectivo que permite conseguir efectos similares a los obtenidos con Visual Studio de Microsoft aunque de forma gratuita y para todas las plataformas.

Las principales características según [27] son:

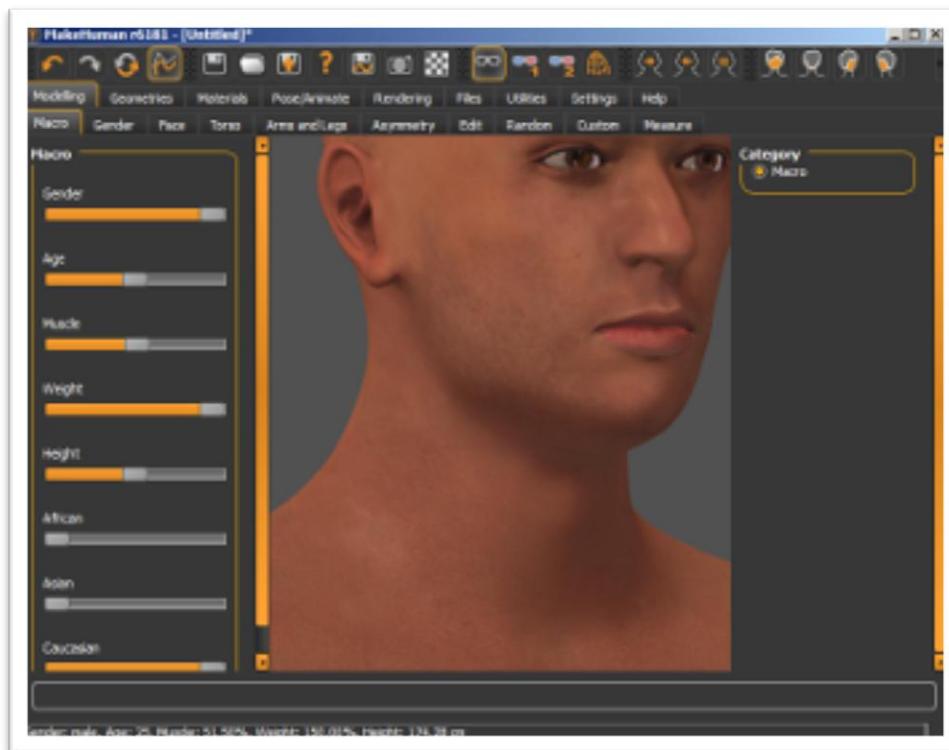
- Multi-plataforma: compatible con Linux, Windows y Mac OS X.
- Edición de texto avanzada: Apoyo de finalización Código de C # , plantillas de código, plegado de código.
- Banco de trabajo configurable: Diseños de ventana totalmente personalizables, atajos de teclado definidos por el usuario, herramientas externas.
- Múltiples idiomas apoyo: C #, Visual Basic.Net, C / C + +, Vala
- Depurador integrado: para depurar Mono y aplicaciones nativas
- GTK # Diseñador Visual: construir fácilmente aplicaciones GTK #
- ASP.NET: Crear proyectos Web con soporte de código completo y prueba en XSP, el servidor web Mono.
- Otras herramientas: Control de código fuente, makefile integración, pruebas unitarias, empaquetado y despliegue, la localización, etc.

### MakeHuman

MakeHuman es un Software libre de gran calidad que permite crear modelos humanos en 3D completamente gratis en diferentes plataformas.

Recientemente MakeHuman ha cambiado a la licencia más liberal disponible para contenido 3D. El contenido creado con MakeHuman está disponible bajo la CC0 licencia, ofreciendo a los artistas una libertad sin precedentes para el uso de sus creaciones MakeHuman en todo lo que puedan imaginar. El equipo MakeHuman espera que esta nueva licencia permitirá a más proyectos, tanto comerciales como no comerciales, para hacer uso de nuestra aplicación. Esto le dará a cualquier persona el acceso a modelos humanos de alta calidad. Como un breve resumen, con este cambio el equipo MakeHuman renuncia expresamente todos sus derechos a las obras creadas con MakeHuman en todo el mundo bajo la ley de derechos de autor, incluyendo todos los derechos conexos. [28]

Los modelos creados pueden salir en los formatos más utilizados en la actualidad como puede ser Blender, Maya, etc.



**Figura 26.** MakeHuman: Software de creación de modelos humanos en 3D de licencia libre.

El uso de este software es muy intuitivo y facilita la creación de modelos humanos en 3D a personas sin ningún conocimiento en el campo del diseño o desarrollo de mundos virtuales. Simplemente se elige en un menú las características de la creación, desde el color de la piel, el sexo o el tamaño de los ojos, hasta llegar al modelo deseado.

### Eagle

Eagle es un software de diseño de diagramas y placas PCB de fácil uso. Es uno de los programas más famosos a nivel mundial en este campo debido a que muchas de las versiones de este software disponen de una licencia Freeware y existe una gran red de repositorios con diseños de Eagle. Debido a esta fama mundial dispone de grandes bibliotecas de tutoriales sobre su uso, lo que hace que a la vez se convierta en un programa de mayor uso.

Unos de los puntos fuertes de este software es su enrutador, que es una función integrada que es capaz de crear los caminos necesarios en las capas indicadas. Sus principales características según [29] son:

- Totalmente integrado en el programa básico
- Enrutamiento de la red hasta 0,02 mm
- Motor básico para el seguimiento me-router, una herramienta que le ayuda en el trazado manual, la traza de la señal seleccionada se calcula automáticamente
- RIPUP y algoritmo de reintento
- No hay restricciones de colocación
- Utiliza el diseño de reglas de diseño
- Cambiar entre el trazado manual y automático en cualquier momento
- Hasta 16 capas de señal (con direcciones preferidas definibles por el usuario)
- Soporte completo de vías ciegas y Buried
- Toma en cuenta las distintas clases netos

El software es capaz de crear los archivos Gerber necesarios para la futura impresión de las placas PCB.

#### Librería MPU6050

El sensor MPU6050 dispone de un procesador DMP del que Invensense no da la completa libertad de programación, por lo que, luchadores de software libre, como Jeff Rowberg, creador de la mayor parte de esta librería, luchan por conseguirlo.

Debido a la falta de una buena documentación disponible públicamente sobre el funcionamiento interno de este dispositivo, toda la información relacionada con el DMP ha sido ingeniería inversa del análisis de las comunicaciones I2C entre el sensor y el software MotionApps proporcionado por Invensense [1]. Este esfuerzo de ingeniería inversa está todavía incompleta en desarrollo, y se espera terminar en una biblioteca de dispositivos que permita acceder a todos los recursos y que admita la configuración completa del DMP. Actualmente, el código fuente disponible sólo proporcionará la configuración de dispositivos básicos y lecturas como la aceleración y el giro (que es ciertamente útil, pero falta el principal punto de venta de la serie MPU-6000, que es el dispositivo de potencia la capacidad de procesamiento de movimiento). [30]

La librería lo que hace es introducir en cadenas de datos hexadecimales las líneas de código necesarias que necesita el procesador DMP del sensor MPU6050 y después pedirle los datos.

Los puntos más importantes de la librería se podría decir que son:

- Programación del DMP
- Inicialización del sensor y del DMP
- Calculo de ángulos de Euler
- Calculo de quaternion
- Calibración del Acelerómetro mediante los valores Offset
- Calibración del Acelerómetro mediante los valores Offset
- Control de características como velocidad del buffer.

En la página oficial de I2cDev.com [30] que es donde se encuentra las librerías y su información, también podemos encontrar muchísima información complementaria como por ejemplo el registro de señales I2C capturado a la hora de hacer la ingeniería inversa.

La librería incluye funciones para manejar los parámetros básicos que ofrece el sensor sin trabajar con el DMP, así como algunas funciones de programación del procesador del sensor.

Tiempo	Dispositivo	Modo	Registro	Longitud	Datos
0.0001035	0x68 (MPU6050)	Leer	0x68 (PHR_MGMT_1)	1	0x00 (DEVICE_RESET = 0, SLEEP = 0, CICLO = 0, TEMP_DIS = 0, CLK_SEL = 0)
0.001806	0x68 (MPU6050)	Escribir	0x68 (PHR_MGMT_1)	1	0x00 (DEVICE_RESET = 1, dormir = 0, CICLO = 0, TEMP_DIS = 0, CLK_SEL = 0)
0.016977	0x68 (MPU6050)	Escribir	0x68 (PHR_MGMT_1)	1	0x70 (DEVICE_RESET = 0, SLEEP = 1 ciclo = 1, TEMP_DIS = 0, CLK_SEL = 0)
0.0171415	0x68 (MPU6050)	Escribir	0x6D (bank_sel)	1	0x70 (PRFTCH_EN = 1, CFG_USER_BANK = 1, MEM_SEL = 16)
0.0173015	0x68 (MPU6050)	Escribir	0x6E (MEM_START_ADDR)	1	0x06 (START_ADDR = 6)
0.0175315	0x68 (MPU6050)	Leer	0x6F (MEM_R_N)	1	0x19 (MEM_R_N = 25)

**Figura 27.** Librería MPU6050: Captura de datos para realizar la ingeniería inversa de Jeff Rowberg. [30]

En resumen, se podría decir que estas librerías son completamente únicas y necesarias a la hora de trabajar en la investigación usando los sensores MPU6050, y aunque no están completas y contengan algunos errores, poco a poco se van ampliando y corrigiendo hasta tener el completo control del DMP complementario a la información de la pagina web de la compañía Invensense.

## 4.3 Descripción Experimental

Esta sección explica todos los puntos clave por los que ha pasado el proyecto en su desarrollo, los problemas que han surgido y la explicación de las decisiones que se han tomado a lo largo del proyecto.

### 4.3.1 Conexión entre Arduino DUE y sensor MPU6050

En primer momento, al comienzo del proyecto, tras la recopilación de información sobre sistemas parecidos en el mercado y la decisión de utilizar el sensor MPU6050 y la tarjeta Arduino Due, ya que este tiene más potencia que las Arduino normales, la intención fue una primera comunicación entre ambos dispositivos.

La comunicación utilizada es I2C ya que es la única comunicación que admite el sensor de Invensense. Tras buscar entre la información incompleta en la página de la compañía [1] ,se buscó una librería complementaria OpenSource que fuera más completa que la oficial por lo que se usó la librería i2CDev de Jeff Rowberg.

Es una librería bastante completa con algunos sketch de ejemplo de conexión, Se utilizó el primero llamado “MPU6050\_raw.ino” que su función se basa en conectar con el sensor, inicializarlo, e imprimir por el terminal los valores raw, de los sensores.

Pero aquí se encuentra el primer problema, tras hacer las conexiones correspondientes entre la placa y el sensor, el sketch de ejemplo no compila en Arduino, el compilador no encuentra:

```
# include <avr/pgmspace.h>
```

#### Arduino(MPU6050.h) Adaptación a Arduino Due 1

Y el problema se debe a que el procesador de Arduino Due es un modelo diferente a los Arduinos anteriores, y estas librerías están preparadas para estos modelos anteriores que utilizan unas librerías ya obsoletas para Arduino Due, tras buscar en los foros de la página oficial de Arduino [21] se encontró la solución, se basa en las siguientes líneas de código:

```
# Ifndef __arm__  
# include <avr/pgmspace.h>  
# else  
# define PROGMEM const  
# define F (x) x  
# endif
```

#### Arduino(MPU6050.h) Adaptación a Arduino Due 2

Estas líneas lo que hacen es que si el procesador es el de las antiguas versiones de Arduino incluya la librería que se estaba usando y sino que defina PROGMEM, que crea la misma función en los nuevos procesadores. La principal función de esta librería sería utilizar la memoria física del procesador ya que la memoria de tiempo de ejecución podría llegar a ser insuficiente.

Tras corregir estos problemas, ya se pueden mostrar los primeros datos raw del acelerómetro y el giróscopo en la pantalla de la terminal.

#### 4.3.2 Conexión con el DMP

Tras la salida de los datos crudos (raw) de los sensores, la primera intención es procesarlos en Arduino, pero pronto aparece la posibilidad de poder procesarlo directamente en el procesador DMP que incorpora el mismo sensor para directamente sacar la información ya procesada.

En este momento se utiliza el segundo sketch de ejemplo que ofrecen las librerías llamado “MPU6050\_DMP6.ino” que ofrece la posibilidad de programar el DMP para sacar los cuaterniones o ángulos de Euler de los sensores.

Este sketch, con ayuda de las librerías, lo que hace es introducir unas cadenas de datos en hexadecimal:

```
const unsigned char dmpMemory[MPU6050_DMP_CODE_SIZE] PROGMEM
```

#### Arduino(MPU6050.h) FifoBuffer

Este código ha sido conseguido por el autor, Jeff Rowberg, a través de la ingeniería inversa que provoca la programación del DMP para que escriba en un buffer de datos FIFO los datos con la siguiente estructura:

[QUAT W]	[QUAT X]	[QUAT Y]	[QUAT Z]	[GYRO X]	[GYRO Y]
1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23
[GYRO Z]	[ACC X ]	[ACC Y ]	[ACC Z ]	[ ]	
24 25 26 27	28 29 30 31	32 33 34 35	36 37 38 39	40 41	

Del cual a través de la librería incluida “helper\_3dMath.h” se puede sacar de estos valores, principalmente del quaternion, los siguientes datos:

- OUTPUT\_READABLE\_QUATERNION
- OUTPUT\_READABLE\_EULER
- OUTPUT\_READABLE\_YAWPITCHROLL
- OUTPUT\_READABLE\_REALACCEL
- OUTPUT\_READABLE\_WORLDACCEL

Se utilizar la segunda función que devuelve los ángulos de Euler ya que son fáciles de entender y por lo tanto fáciles de usar. Que indican la posición de cada ángulo (Yaw, Pitch y Roll) del sensor según la gravedad.

#### 4.3.3 Inicialización los sensores

Para que estas comunicaciones funcionen primero se deben inicializar los sensores asegurando que la comunicación es correcta y que el DMP está dispuesto a trabajar y a ofrecer la información que se le pueda pedir, la función encargada de esto se encuentra en “Final.ino” y se llama “Incializartodo()”:

```
void inicializartodo (MPU6050 sensor, int* offset,int pin)
{
    delay(2); // Espera dos Milisengundos
    sensor.initialize(); //< Inicializa el sensor
    devStatus = sensor.dmpInitialize(); //< Inicializa el DMP

    // Si la conexión es correcta enciende el pin de test
    if (sensor.testConnection())
    {
        digitalWrite(pin, HIGH)
    }
}
```

Arduino(Final.ino) inicializarTodo() 1/2.

```
    /// Si no hay errores al inicializar activa el sensor
    if (devStatus == 0)
    {
        sensor.setDMPEnabled(true);
        // Recoge si hay interrupciones en el sensor
        mpuIntStatus = sensor.getIntStatus();
        // Recoge el tamaño del paquete
        packetSize = sensor.dmpGetFIFOPacketSize();
    }
    delay(2); //Espera dos milisegundos
}
```

Arduino (Final.ino) Inicializartodo() 2/2

#### 4.3.4 Calibrar los Sensores

Tras mostrar los datos de los ángulos de Euler en la terminal, se puede comprobar que tras un cierto tiempo o ciertos movimientos que terminan volviendo a la posición inicial del sensor, no devuelve los mismos valores. Se debe a que el sensor no esta calibrado, pero dispone de unos valores offset, que son unas variables que corrigen los valores iniciales para dejarlos con un valor correcto. Por lo que se crea un programa de Arduino para calcular los valores Offset llamado “Calibrador.ino”:

```
/// Incluye la librería Wire.h que es la que controla la
comunicación I2C
#include "Wire.h"
/// Incluye la librería I2Cdev.h que controla la comunicación con
el sensor
#include "I2Cdev.h"
/// Incluye las funciones de control del sensor
#include "MPU6050.h"

/// Construimos el objeto del sensor
MPU6050 accelgyro;

/// Creamos las variables
int16_t ax, ay, az;
int16_t gx, gy, gz;
```

Arduino(Calibrador.ino) Calibrador.ino 1/2

```
// =====
// ===                                     SETUP                                     ===
// =====
void setup()
{
    // Iniciamos la comunicación I2C
    Wire.begin();

    // Iniciamos la comunicación Serie
    Serial.begin(57600);

    // Inicializamos la comunicación con el sensor
    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();

    //Comprobamos si la conexión se ha realizado correctamente
    Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6050
connection successful" : "MPU6050 connection failed");

    // Introducimos los valores para corregir la calibración
    accelgyro.setXGyroOffset(22);
    accelgyro.setYGyroOffset(-25);
    accelgyro.setZGyroOffset(6);
    accelgyro.setXAccelOffset(-300);
    accelgyro.setYAccelOffset(-1396);
    accelgyro.setZAccelOffset(2677);
}

// =====
// ===                                     LOOP                                     ===
// =====
void loop()
{
    // Pedimos los valores crudos y los imprimimos
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    Serial.print(gx); Serial.print("\t");
    Serial.print(gy); Serial.print("\t");
    Serial.print(gz); Serial.print("\t");
    Serial.print(ax); Serial.print("\t");
    Serial.print/ay); Serial.print("\t");
    Serial.println(az);
}
```

Arduino(Calibrador.ino) Calibrador.ino 2/2.

Cada sensor necesita unos valores diferentes por lo que hay que calibrar cada MPU6050 por separado y los valores que debe devolver en la terminal para que este perfectamente calibrado son:

0 , 0 , 0 , 0 , 0 , 16000

Los tres primero son los valores del giróscopo, que para corregir el error se debe dividir entre cuatro la diferencia para calcular el valor del offset, por ejemplo, si el valor es -100, el valor que se debe colocar es  $100/4 = 25$ . Así  $(25*4)-100 = 0$

Los tres segundos pertenecen al acelerómetro, por ello que el ultimo valor sea diferente de 0, ya que es el valor de la gravedad que se refiere a Z. Esto quiere decir que siempre que se calibre el sensor, este debe estar en posición horizontal para que el acelerómetro marque la gravedad completamente en Z. Para calcular el valor que se debe colocar en el valor Offset del acelerómetro debes dividir entre 8,7 aproximadamente, no es tan exacto por lo que hay que calcular después si hay que sumarle un poco más o un poco menos.

Estos valores se introducen en el sensor en la función de InicializarTodo para que al inicializar el sensor ya disponga de su correcta calibración.

#### 4.3.5 Uso de Quaterniones

En este punto de puede comprobar que los valores de los ángulos de Euler siempre que vuelve a la misma posición devuelve el mismo valor, pero en algunas posiciones el sensor no devuelve datos coherentes, y se debe a que al utilizar directamente los ángulos de Euler en el control de movimiento puede causar un gran problema llamado Gimbal's Lock.

Lo que viene a decir el Gimbal's Lock es que según algunas posibles posiciones en algún momento, dos ángulos puedes coincidir y esto puede causar confusión y error en los datos, por eso es recomendable usar el mejor sistema matemático para esta situación, los quaternios. Los datos de este sistema no son tan claros a simple vista debido al uso de números imaginarios pero a efectos prácticos es la mejor solución.

La función que se encarga de recoger los quaternios del Buffer se encuentra el "Final.ino" se llama "cogerDatos ()":

```
void CogerDatos(MPU6050 sensor, int pin, int num)

{
    // Si la conexión no se correcta en tiempo de ejecución
    // apaga el led de test
    if (!sensor.testConnection())
    {
        digitalWrite(pin, LOW);
    }
    // Recoge si el sensor tiene interrupciones
    mpuIntStatus = sensor.getIntStatus();
    // Recoge el contador del bus FIFO
    fifoCount = sensor.getFIFOCount();

    // si el bus FIFO está lleno o hay alguna interrupción se
    // resetea el bus
    if ((mpuIntStatus & 0x10) || (fifoCount == 1024))
    {
        sensor.resetFIFO();
    }

    // sino espera a que el tamaño sea el adecuado y recoge los
    // datos
    else
        if (mpuIntStatus & 0x02)
        {
            Quaternion qu; // Crea el quaternion
            while (fifoCount < packetSize) fifoCount =
            sensor.getFIFOCount();
            // Espera hasta que el tamaño sea el adecuado
            sensor.getFIFOBytes(fifoBuffer, packetSize);
            // lee el paquete del bus FIFO
            fifoCount == packetSize; // actualiza el fifoCount

            // Saca el quaternion del paquete recibido
            sensor.dmpGetQuaternion(&qu, fifoBuffer);

            // Guarda el quaternion en el vector principal
            datos[num]=qu.w;
            datos[num+1]=qu.x;
            datos[num+2]=qu.y;
            datos[num+3]=qu.z;
        }
    }
}
```

Arduino(Final.ino) cogerDatos().

Una vez conseguido los valores de los quaterniones de cada uno de los sensores, los datos son guardados en un vector principal que mantiene la correcta información de los sensores.

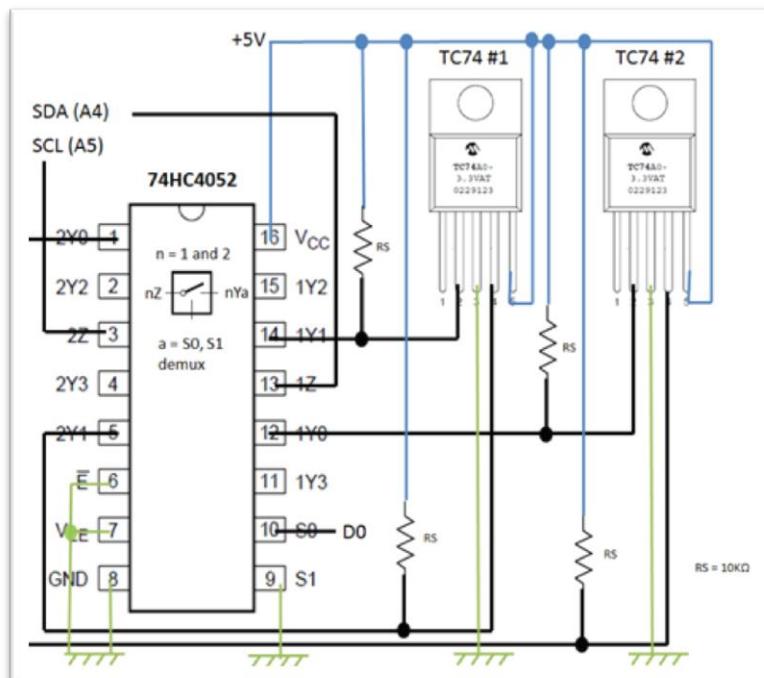
#### 4.3.6 Comunicación con varios sensores

Para la comunicación I2C con un solo sensor, con conectar una línea de SDA y otra línea de SCL junto a la alimentación ya se puede realizar una comunicación con cualquiera de ellos, pero el problema es la comunicación con todos. Una posibilidad era utilizar la función:

```
void MPU6050::setDeviceID(uint8_t id)
{
    I2Cdev::writeBits(devAddr, MPU6050_RA_WHO_AM_I,
    MPU6050_WHO_AM_I_BIT, MPU6050_WHO_AM_I_LENGTH, id);
}
```

Arduino(MPU6050.cpp) setDeviceID().

Que se encuentra en la librería “MPU6050.cpp” y que según Jeff Rowberg sirve para cambiar la dirección I2C del sensor, pero según la información de la compañía es imposible cambiar la dirección de un MPU6050, la única opción que te ofrece el sensor es cambiar el pin AD0 poniéndolo en “High” (activo) y la dirección cambiar de la predefinida que es 0x68 a 0x69, y estas son las dos únicas direcciones que podría adoptar el sensor.



**Diagrama 5.** Comutación I2C: Ejemplo de cómo conmutar la señal I2C con un módulo 74HC4052.[4]

Partiendo del dato de que solo se puede realizar la comunicación con los sensores MPU6050 con dos únicas direcciones. Se ha solucionado el problema mediante hardware. Utilizando multiplexores analógicos 74HC4052 que hagan el efecto de conmutadores y según la opción que asigne dirige la señal introducida a la dirección que se necesite, en este caso SCL y SDA a la dirección que quieras.

El número de sensores MPU6050 a conectar son 12, pero al activar el pin AD0 de la mitad, quedan 6 líneas que alimentar. Cada módulo 74HC4052 puede dividir las dos señales necesarias (SDA y SCL) en 4 caminos diferentes, por lo que es necesario dos conmutadores diferentes para poder repartir la señal hasta por 8 caminos diferentes, lo que hace que queden cubiertos los 6 que son necesarios.

Entonces de la placa de Arduino, los pines necesarios serían:

- SDA y SCL que a través de los conmutadores llegan a todos los sensores.
- Vcc y GND que llegan directamente a cada sensor y a los conmutadores.
- 2 pines para señalar el camino deseado del módulo 74HC4052.
- 2 pines para activar o desactivar cada módulo 74HC4052.

Se ha creado una función en “Final.ino” que se encarga de preparar estos pines para elegir la posición que necesitas en el conmutador, que se llama “posición()”

```
void posicion(int pos)
{
    delay(5);
    // Según la posición coloca los pines de selección
    switch (pos)
    {

        case 1: // Posicion 1
            digitalWrite(Cam_pin, LOW);
            digitalWrite(Cam_pin2, LOW);
            digitalWrite(Cam_inh_1, LOW); // en low activa U1
            digitalWrite(Cam_inh_2, HIGH); // en High desactiva U2
            break;

        case 2: // Posicion 2
            digitalWrite(Cam_pin, HIGH);
            digitalWrite(Cam_pin2, LOW);
            digitalWrite(Cam_inh_1, LOW); // en low activa U1
            digitalWrite(Cam_inh_2, HIGH); // en High desactiva U2
            break;
    }
}
```

Arduino(Final.ino) posicion() 1/2.

```
case 3: // Posicion 3
    digitalWrite(Cam_pin, LOW);
    digitalWrite(Cam_pin2, HIGH);
    digitalWrite(Cam_inh_1,LOW); // en low activa U1
    digitalWrite(Cam_inh_2,HIGH); // en High desactiva U2
break;

case 4: // Posicion 4
    digitalWrite(Cam_pin, LOW);
    digitalWrite(Cam_pin2, LOW);
    digitalWrite(Cam_inh_1,HIGH); // en low activa U1
    digitalWrite(Cam_inh_2,LOW); // en High desactiva U2
break;

case 5: // Posicion 5
    digitalWrite(Cam_pin, HIGH);
    digitalWrite(Cam_pin2, LOW);
    digitalWrite(Cam_inh_1,HIGH); // en low activa U1
    digitalWrite(Cam_inh_2,LOW); // en High desactiva U2
break;

case 6: // Posicion 6
    digitalWrite(Cam_pin, LOW);
    digitalWrite(Cam_pin2, HIGH);
    digitalWrite(Cam_inh_1,HIGH); // en low activa U1
    digitalWrite(Cam_inh_2,LOW); // en High desactiva U2
break;

} // final switch
delay(5);
} // final posición
```

#### Arduino (Final.ino) posición() 2/2

Como se ve en el código *Cam\_pin* y *Cam\_pin2* son los selectores del camino de los conmutadores y *Cam\_inh\_1* y *Cam\_inh\_2* son los pines que activan o desactivan que módulo 74HC4052 es el que funciona, partiendo de la idea de que para activar estos módulos la señal que tiene que llegar tiene que ser “low”(0);

#### 4.3.7 Comunicación entre Arduino y Unity 3D

La comunicación entre la tarjeta Arduino Due y el Software creado con Unity 3D se realiza por comunicación Serie. En el programa “Final.ino” existe un vector con todos los datos recogidos de los vectores y se imprimen en el puerto serie a la velocidad deseada mediante la función “imprimir()”:

```
void imprimir(float* vector, int tam)
{
    int cont=0;///< Variable contador

    // Recorre el vector imprimiendo todo con una coma de
    // separación
    while (cont<tam)
    {
        Serial.print(vector[cont]);
        Serial.print(" , ");
        cont++;
    }

    //Imprime el último dato con un salto de línea
    Serial.println(vector[tam]);
}
```

**Arduino(Final.ino) imprimir().**

La función principal, como ya se ha dicho, es la de imprimir todos los datos en una sola línea separados por una coma, y salta a la siguiente línea para que cada línea desde una posición diferente.

Por parte de Unity toda la programación se realiza a través de Scripts en C#, y hay un Script encargado de recoger la información del puerto serie llamado “Gui.cs”:

```
...

SerialPort sp = new SerialPort("COM2", 115200);

// =====
// ===                      START                      ===
// =====
void Start ()
{
    sp.Open();
    sp.ReadTimeout = 50;
}
```

**Unity(Gui.cs) Comunicación Serie en Unity 1/2.**

```
// ======  
// === UPDATE ===  
// ======
```

```
void Update ()  
{  
    try  
    {  
        valu = sp.ReadLine(); //Read the information  
        string[] vec3 = valu.Split(',');  
  
        q0.w= float.Parse (vec3[0]);  
        q0.z= -(float.Parse (vec3[1]));  
        q0.x= (float.Parse (vec3[2]));  
        q0.y= -(float.Parse (vec3[3]));  
  
        q1.w= float.Parse (vec3[4]);  
        q1.z= -(float.Parse (vec3[5]));  
        q1.x= (float.Parse (vec3[6]));  
        q1.y= -(float.Parse (vec3[7]));  
  
        q2.w= float.Parse (vec3[8]);  
        q2.z= -(float.Parse (vec3[9]));  
        q2.x= (float.Parse (vec3[10]));  
        q2.y= -(float.Parse (vec3[11]));  
  
    ...  
}
```

#### Unity(Gui.cs) Comunicación Serie en Unity 2/2.

Y seguiría hasta “qB” que es el nombre que recibe el quaternion del último sensor MPU6050. Estas líneas de código lo que hacen principalmente es leer la línea entera recogida por el puerto serie y separa los datos en grupos de 4, dando lugar cada uno estos conjuntos a el quaternion de nuevo con el orden necesario en Unity para que sean exactamente iguales que antes de mandarlos desde Arduino.

Un error muy común en la comunicación con el puerto serie desde Unity es la configuración de serie del programa, buscando en la ayuda de la página de Unity [26] se puede encontrar que hay que cambiar la configuración del proyecto de Unity en:

- Edit -> Project Settings-> Player -> Api Compatibility level

Cambiar la Opción de “.NET 2.0” a “.NET 2.0 SubSet”, pero se trata de una errata de los desarrolladores de Unity ya que para que funcione correctamente hay que a cambiarlo a la opción “.NET 2.0” .

De esta forma, el software creado en Unity ya recoge perfectamente los valores de todos los quaterniones de los que Arduino ha recogido de los sensores y enviado por el puerto serie.

#### 4.3.8 Modelo 3D

Aun existiendo muchos repositorios de modelos 3D en internet, se decide crear un modelo personalizado para la ocasión con un software libre llamado MakeHuman [28] que de forma sencilla permite crear un modelo humano 3D con la articulación necesaria.

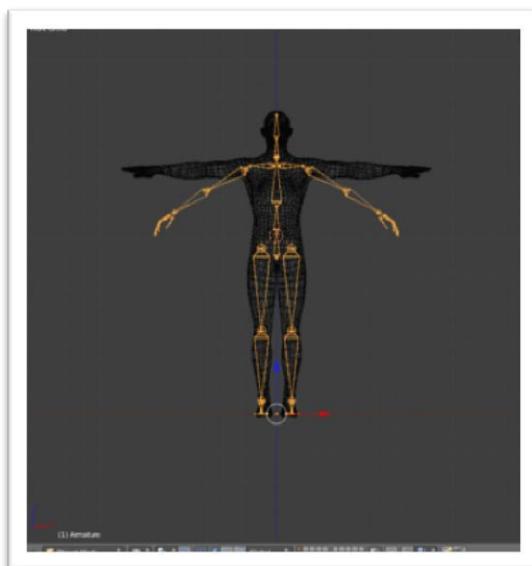
El programa permite crear un modelo que diseño personalizado y se decide crearlo con los rasgos más básicos y neutrales posibles.

En las opciones de exportación, Se selecciona la opción de ponerle un esqueleto para que más tarde disponga de articulaciones que pueda mover y lo es exportado con el formato Collada (\*.dae).



**Figura 28.** Modelo 3D: Diseño del cuerpo humano 3D que más tarde reproducirá los movimientos.

Después es importado al proyecto de Unity 3D y se introduce en la escena comprobando que correctamente dispone de un esqueleto con el que podemos partir el cuerpo para darle diferentes movimientos.



**Figura 29.** Modelo 3D en Unity: Vista del esqueleto del modelo 3D en Unity.

#### 4.3.9 Scripts de Movimiento

Para darle movimientos a las articulaciones del modelo humano en 3D, Unity trabaja con Scripts programados en C#Script o en javaScript. Se ha escogido C#Script ya que es más parecido a C# normal.

El Script lo único que hace es modificar los ángulos de rotación de la articulación, según el valor que sea introducido, que en este caso es la variable global que se ha declarado en el Script “gui.cs” llamada q0 donde se ha guardado los valores del quaternion creado por el primer sensor. El ultimo valor es una variable que indica la velocidad deseada para mover la articulación hasta llegar a los nuevos valores que marque el quaternion.

La velocidad también es un valor que está declarado en el archivo “gui.cs” como una variable global ya al ser incluida en todos los scripts de movimiento y que todos ellos deben ir a la misma velocidad, se considera más cómodo a la modificar una sola variable y que influya el cambio en todos a la vez.

Se han creado 12 scripts diferentes, uno por cada sensor y/o articulación. Se explica un ejemplo con “Brazo\_der.cs”:

```
/// Importamos librerias
using UnityEngine;
using System.Collections;

/// Creamos la clase
public class brazo_der : MonoBehaviour
{

    // =====
    // == START ==
    // =====
    void Start ()
    {

    }

    // =====
    // == UPDATE ==
    // =====
    void Update ()
    {
        // Imprimimos los movimientos a la velocidad determinada
        transform.rotation = Quaternion.Slerp (transform.rotation,
                                                gui.q0 ,
                                                gui.speed * Time.deltaTime);
    }
}
```

#### Unity(\*.cs) Script de Movimiento.

Los 12 Scripts creados para el movimiento son iguales, con la única diferencia de que el quaternion que escogen para su articulación es diferente en cada uno de ellos y que cada uno luego está asociado a una articulación diferente.

Están organizados por carpetas, una por extremidad, para que sea más fácil la organización, corrección e incluso el mantenimiento de los scripts en caso de que hubiera problemas con ello.

Una vez programados todos los scripts se asocia cada uno de ellos con la misma relación que las articulaciones físicas. Como se ve en la siguiente tabla:

<b>Articulación Real</b>	<b>Posición en la lista de Arduino</b>	<b>Quaternion Asociado en Unity</b>	<b>Articulación en Unity</b>
Brazo Derecho	0-3	$Q0$	<i>Brazo_der</i>
Codo Derecho	4-7	$Q1$	<i>Codo_der</i>
Mano Derecha	8-11	$Q2$	<i>Mano_der</i>
Brazo Izquierdo	12-15	$Q3$	<i>Brazo_izq</i>
Codo Izquierdo	16-19	$Q4$	<i>Codo_izq</i>
Mano Izquierda	20-23	$Q5$	<i>Mano_izq</i>
Pierna Izquierda	24-27	$Q6$	<i>Pierna_izq</i>
Rodilla Izquierda	28-31	$Q7$	<i>Rodilla_izq</i>
Pie Izquierdo	32-35	$Q8$	<i>Pie_izq</i>
Pierna Derecha	36-39	$Q9$	<i>Pierna_der</i>
Rodilla Derecha	40-43	$QA$	<i>Rodilla_der</i>
Cuerpo	44-47	$QB$	<i>Cuerpo</i>

**Tabla 2.** Relación Sensores-Modelo3D: Relación de cada uno de los sensores con las articulaciones del modelo.

En un primer momento se planteó solo las extremidades, y así se diseñó pero por último se ha decidido cambiar lo que debería ser el pie izquierdo por el Cuerpo, dando así movilidad ya no solo a las extremidades sino también al tronco.

#### 4.3.10 Interfaz de usuario

Cuando el usuario arranca la simulación programada en Unity con el modelo y los scripts, puede encontrarse con dos situaciones:

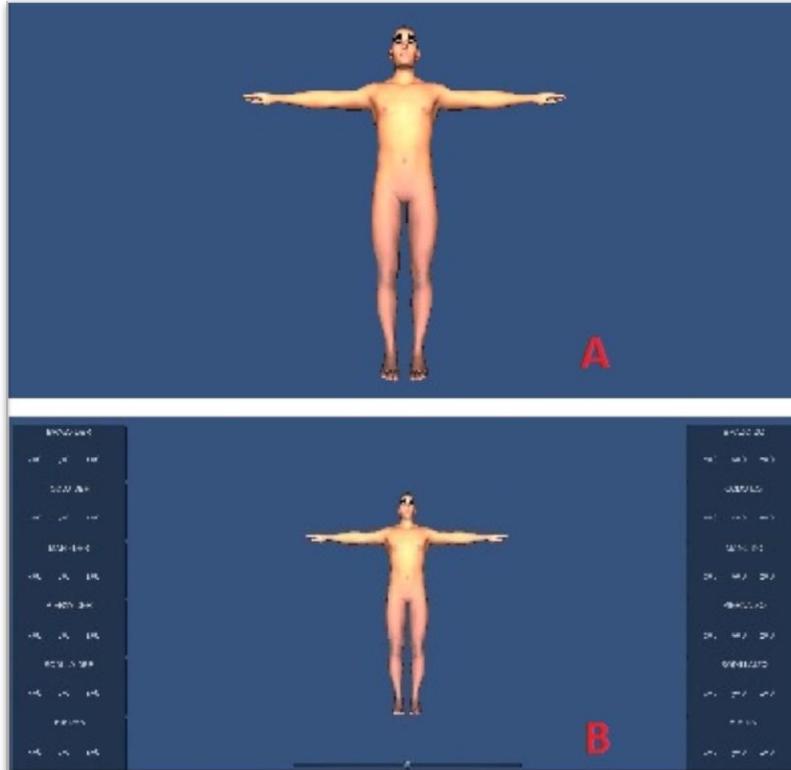
- Fondo Azul y el modelo Humano en 3D.
- Fondo Azul, Modelo Humano 3D, los datos numéricos de los ángulos y un slider.

Si el usuario pulsa la barra de espacio puede cambiar entre las dos versiones diferentes, en una el muñeco se ve más grande y en la otra más pequeño junto a los datos. El código que permite cambiar entre interfaces se encuentra en el script principal “gui.cs”:

```
if(Input.GetKeyDown(KeyCode.Space))  
{  
    if (VerDatos)  
    {  
        VerDatos=false;  
        acer=20;  
    }  
    else  
    {  
        VerDatos=true;  
        acer = 30;  
    }  
}
```

**Unity(Gui.cs) Interfaz de usuario.**

Al presionar la tecla espacio cambia de una condición a otra, activa o desactiva la variable “VerDatos” que más adelante dejará o no mostrar los datos con un condicional y la variable “acer” guarda la distancia a la que se encuentra el “Main Camera” del modelo Humano, es decir, la distancia que separa el modelo de la pantalla, por lo que su tamaño varía.



**Figura 30.** Interfaz de Usuario: En el ejemplo A vemos simplemente el modelo y en el B el humano es más pequeño y se incorporan los datos.

Los datos laterales son los de los ángulos de Euler sacados de los quaterniones recibidos de Arduino, por si hace falta la comprobación de algún ángulo de forma exacta, todo esto está dentro de un condicional que da paso la variable “VerDatos” antes descrita que se encuentra en el script principal “gui.cs”:

```
...
/// Si VerDatos ha sido activado mostramos los datos
if (VerDatos)
{
    /// Imprimimos datos del Brazo derecho
    GUI.Box (new Rect(30,50,200,100),"BRAZO DER");

    GUI.Label (new Rect(60,100,200,50),"x="+Mathf.FloorToInt(q0.eulerAngles.y));
    GUI.Label (new Rect(110,100,200,50),"y="+Mathf.FloorToInt(q0.eulerAngles.x));
    GUI.Label (new Rect(160,100,200,50),"z="+Mathf.FloorToInt(q0.eulerAngles.z));

    /// Imprimimos datos del Codo Derecho
    GUI.Box (new Rect(30,150,200,100),"CODO DER");

    GUI.Label (new Rect(60,200,200,150),"x="+Mathf.FloorToInt(q1.eulerAngles.y));
    GUI.Label (new Rect(110,200,200,150),"y="+Mathf.FloorToInt(q1.eulerAngles.x));
    GUI.Label (new Rect(160,200,200,150),"z="+Mathf.FloorToInt(q1.eulerAngles.z));

...
}
```

#### Unity(Gui.cs) Ver Datos .

Como se puede observar no es más que imprimir de forma estética y ordenada los ángulos de Euler calculados de los quaterniones, ayudado de la librería “Mathf”. Seguiría así hasta completar los 12 quaterniones.

El slider que se encuentra debajo del modelo 3D sirve para crear un movimiento de la perspectiva de visión del modelo, lo que hace es mover la cámara alrededor del cuerpo. El código también se encuentra dentro del condicional en el que se encuentra la impresión de los ángulos dentro del script general “gui.cs”:

```

...
seno = acer* Mathf.Sin (pos_cam);
coseno =acer * Mathf.Cos (pos_cam);
transform.position = new Vector3 (seno, 0,coseno);

angu = pos_cam * 180 / Mathf.PI;

transform.eulerAngles = new Vector3 (0, 180+angu,0);

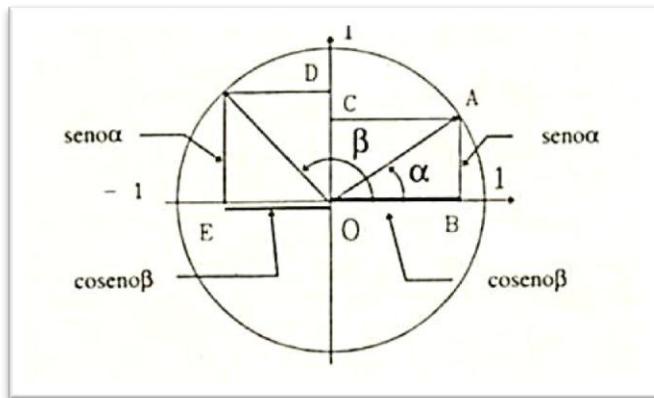
...
pos_cam = GUI.HorizontalSlider (new Rect (Screen.width/2-
200,Screen.height-100,400,30),pos_cam,-Mathf.PI/2,Mathf.PI/2);

...

```

#### Unity(Gui.cs) Slider.

Estas líneas de código lo que hacen a través del seno y el coseno es dibujar un semicírculo alrededor del modelo según la posición del slider siempre con la separación antes descrita y después lo que hacer es colocar ya no solo la posición, sino también el ángulo para que siempre este mirando al centro de su semicírculo que es donde se encuentra el modelo.



**Figura 31.** Giro de la cámara: En esta imagen se muestra el modelo matemático que demuestra cómo se pasa de un movimiento lineal a circular mediante senos y cosenos.

#### 4.3.11 Hardware

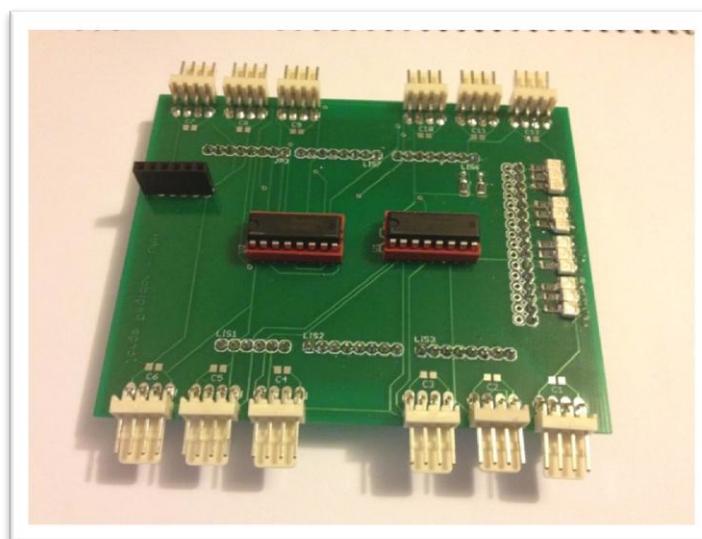
En este proyecto se ha diseñado una shield para Arduino Due que demultiplexa un canal de comunicación I2C en 6 canales diferentes. El shield ha sido diseñado completamente para la ocasión está declarado completamente Hardware Libre para su mayor difusión. El circuito impreso ha sido diseñado con el programa Eagle y fabricado en la compañía china Ithead.

Lo que hace es repartir la señal SDA y SDL (pines 20 y 21) a los dos módulos 74HC4052 para que estos repartan las señales a 6 canales compartidos por 2 salidas para cada canal en que se conectan los cables que llegan a los sensores. Los módulos están colocados sobre unos zócalos debidamente soldados a la placa.

La shield dispone de dos resistencias pull-up, por si los sensores a conectar por I2C lo necesitaran, junto a unos jumpers que pueden activar o desactivar esta opción. En este caso concreto los jumpers están desactivados ya que los sensores MPU6050 disponen de este tipo de resistencias y no necesitan que se añadan exteriormente.

Las salidas de los sensores están ordenadas de C1 a C12 y cada una de ellas dispone de un hueco para añadirle un condensador en caso de que fuera necesario. Estas salidas están organizadas de tres en tres pensado para organizar los cables de manera más eficiente debido a que cada grupo pertenece a la extremidad más cercana.

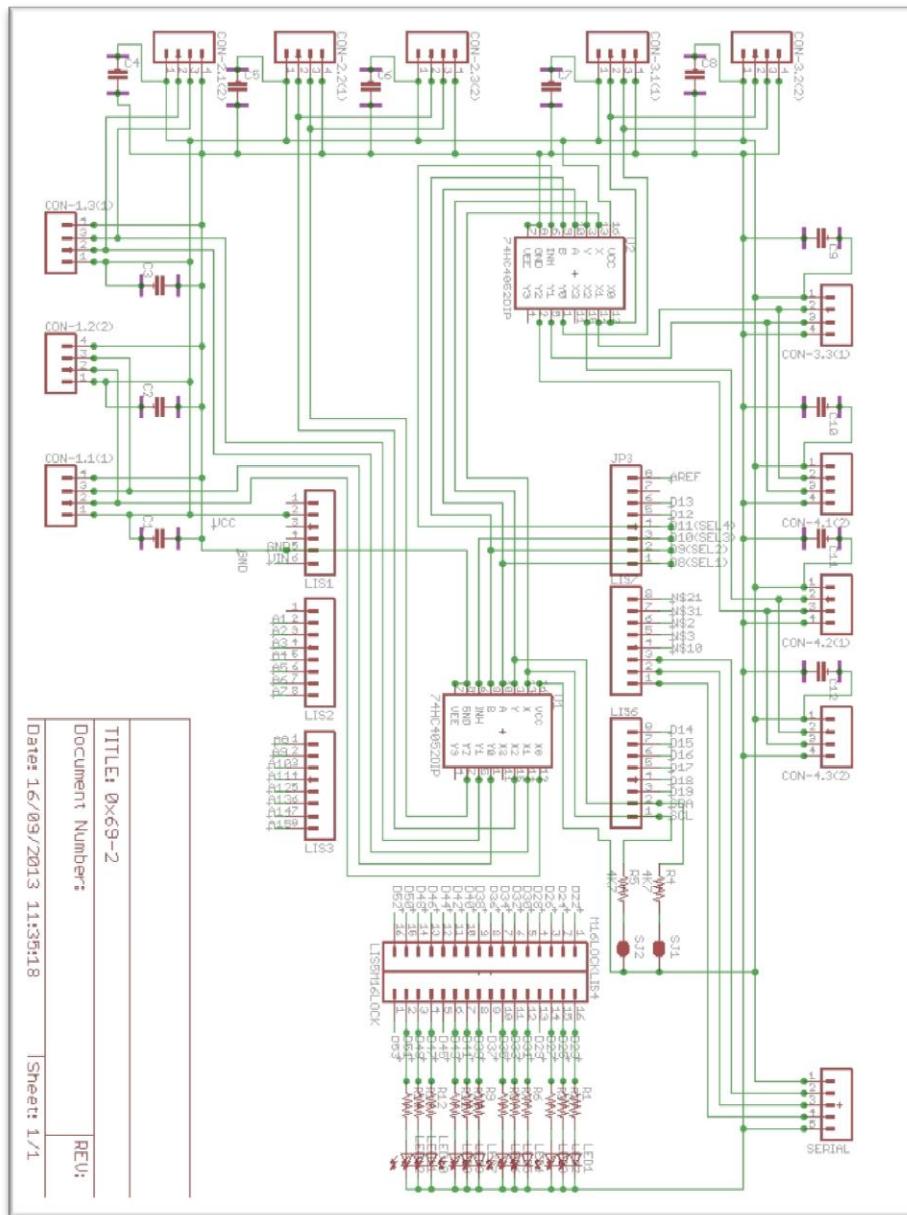
También dispone de una serie de leds rojos que indican el correcto funcionamiento de los sensores, en caso de que uno de los sensores no se conectara debidamente ese led no se encendería y alertaría de que algún problema está sucediendo.



**Figura 32.** Hardware: Shield para Arduino Due que redirige la señal I2C a 12 salidas diferentes.

Por ultimo dispone de una conexión para un módulo bluetooth conectado a los pines Rx y Tx para que se pueda realizar una comunicación inalámbrica con el ordenador.

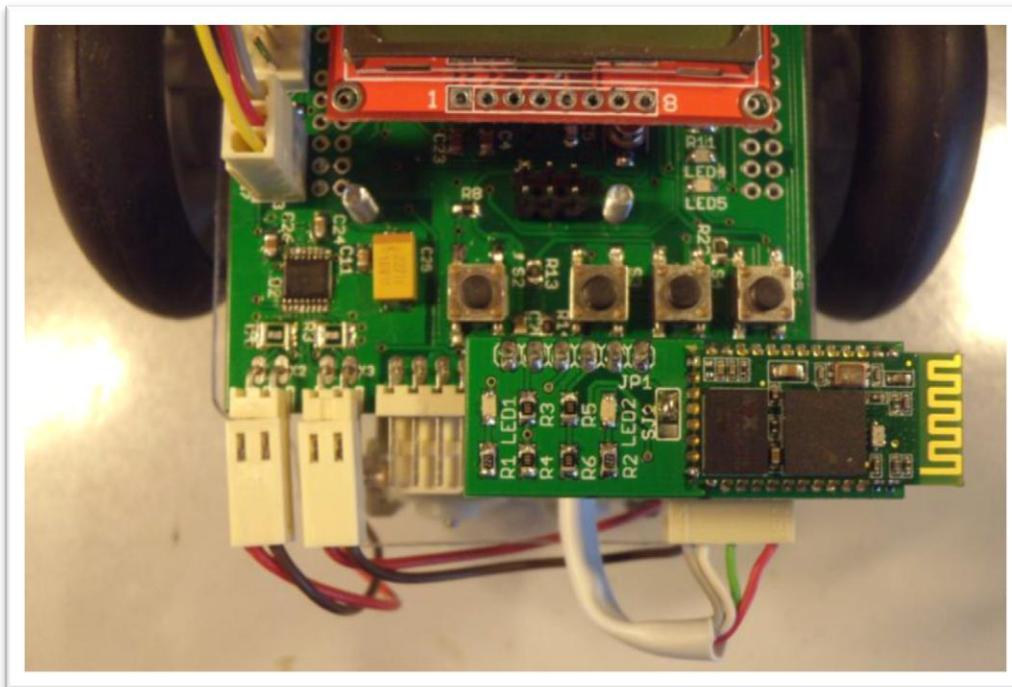
Para su fabricación se ha necesitado los archivos “\*.sch” y “\*.brd” para que con ayuda de los archivos Gerber que proporciona la fábrica Ithead se pudieran fabricar.



**Diagrama 6.** Shield.sch: esquema de la Trajeta Shield

#### 4.3.12 Comunicación inalámbrica

En un primer momento la comunicación se intentó realizar por medio de un módulo bluetooth HC-05 directamente al ordenador usando la conexión bluetooth de este. El modulo utilizado para crear la conexión bluetooth del proyecto es el HC-05 junto con una placa controladora diseñada por el Departamento de Electrónica de la Universidad de Alcalá [23] .



**Figura 33.** Bluetooth Depeca: Bluetooth diseñado por el Departamento de Electrónica de la Universidad de Alcalá.

Tras varios intentos de sincronizar el módulo HC-05 con el ordenador, se descubre que los protocolos de Windows 7 y Unity 3D ponían demasiados impedimentos para realizar directamente la conexión. Por lo que más tarde se optó por la opción de crear un cable virtual antes de llegar al ordenador es decir, recibir la señal con otro módulo HC-05 y convertir la señal de bluetooth a USB de nuevo antes de llegar al ordenador

Esta nueva arquitectura se puede realizar de dos formas diferentes: conversor USB-Serie TTL o con otra tarjeta de Arduino.

Si se realiza con un conversor lo único que hay que hacer es soldar los pines oportunos y ya podría ser conectado al ordenador y al ser sincronizado correría perfectamente.

Si se realiza con otra Arduino Due a través de un software previamente instalado, se reviven las señales por un puerto serie y son redirigidas por el USB hacia el ordenador. El programa que hace esta función en la Arduino se llama “Receptor.ino”.

```
// ======  
// === START ======  
// ======  
void setup()  
{  
  
Serial1.begin(115200);  
Serial.begin(115200);  
}  
  
// ======  
// === START ======  
// ======  
void loop()  
{  
while(!Serial);  
while(Serial1.available())  
Serial.write(Serial1.read());  
}
```

#### Arduino(Receptor.ino) Receptor.ino.

A la hora de sincronizar los dos HC-05 el proceso se realiza en modo configuración (Soldando el Jumper que pone a 1 el pin 34 del HC-05) y a través de comandos AT se configura:

AT+NAME = Depecabot (Define el nombre)

AT+UART = 115200,1,0 (Define la velocidad a 115200)

AT+PSWD = 0000 (Define el pin de emparejamiento)

AT+ROLE = 1 (1 el Master y 0 el Esclavo)

AT+CMODE = 0 (Solo el Master, y lo que hace es que solo se pueda sincronizar con un módulo)

AT+ADDR? (Solo el esclavo, y devuelve la dirección MAC)

AT+BIND = 2012,1,211036 (Solo el Master, dándole la dirección con la que tiene que sincronizar, que es la del esclavo)

Después se debe desoldar el jumper del pin 34, y ya están sincronizados los dos módulos, ya con darles alimentación quedan sincronizados.

#### 4.3.13 Alimentación

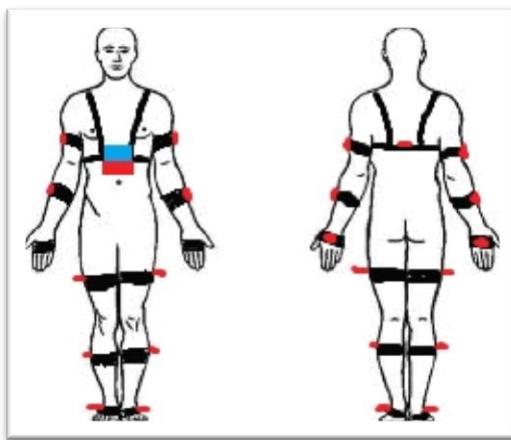
La alimentación se resume a una batería lipo utilizada en aparatos de radio control con las siguientes características:

- 7,4 V
- 3 Celdas
- 1000 mAh

También dispone de un adaptador de la salida de batería de radio control a la entrada de alimentación de Arduino, junto con un sistema de seguridad que incluye un porta fusibles aéreo por la peligrosidad que este tipo de baterías acarrea ya que la batería podría explotar y no sería lo adecuado ya que ésta va pegada al cuerpo del usuario.

#### 4.3.14 Traje

El traje en el que se transportan y usa el proyecto se basa en un sistema de gomas con velcros adaptable a todo tipo de personas, desde las más grandes hasta las más pequeñas. Se compone de 13 piezas, tres por extremidad y una principal en forma de chaleco que es la que carga con la Arduino, la shield y la batería.



**Figura 34.** Traje: estructura del traje que porta el proyecto y se adapta al usuario para crear los movimientos.

Las que van en las articulaciones disponen de una especie de bolsillo en el que introducir los sensores. En los brazos los sensores entran directamente en los bolsillos pero en las piernas y el sensor que va colocado en la parte de atrás, para la espalda, llevan unos adaptadores para que al colocarlos en los bolsillos queden en horizontal, ya que es como deben estar para su correcto funcionamiento.

## 4.4 Conclusiones

Tras el desarrollo de este proyecto se puede decir que los sensores MPU6050 son de una calidad y precisión excelente. Cabe pensar que si la programación del procesador DMP fuera libre de ser programado por cualquier usuario, las posibilidades de estos sensores podrían ser infinitas. Mientras tanto, gracias a las librerías de Jeff Rowberg, se les puede dar un gran uso ya que las posibilidades, aunque reducidas, siguen siendo amplias.

A la finalización de este proyecto el autor se ha dado cuenta de que el software libre es necesario, ya que si no existiera, la investigación se puede ver truncada o incluso se puede reinventar lo mismo muchas veces en vez de mejorarlo. En este proyecto he usado tanto librerías o software que ha hecho posible el trabajo gracias al trabajo de esos investigadores anónimos, o no tan anónimos, que han expuesto su sabiduría sin cobrar y dejando la oportunidad de continuar lo que ellos han hecho.

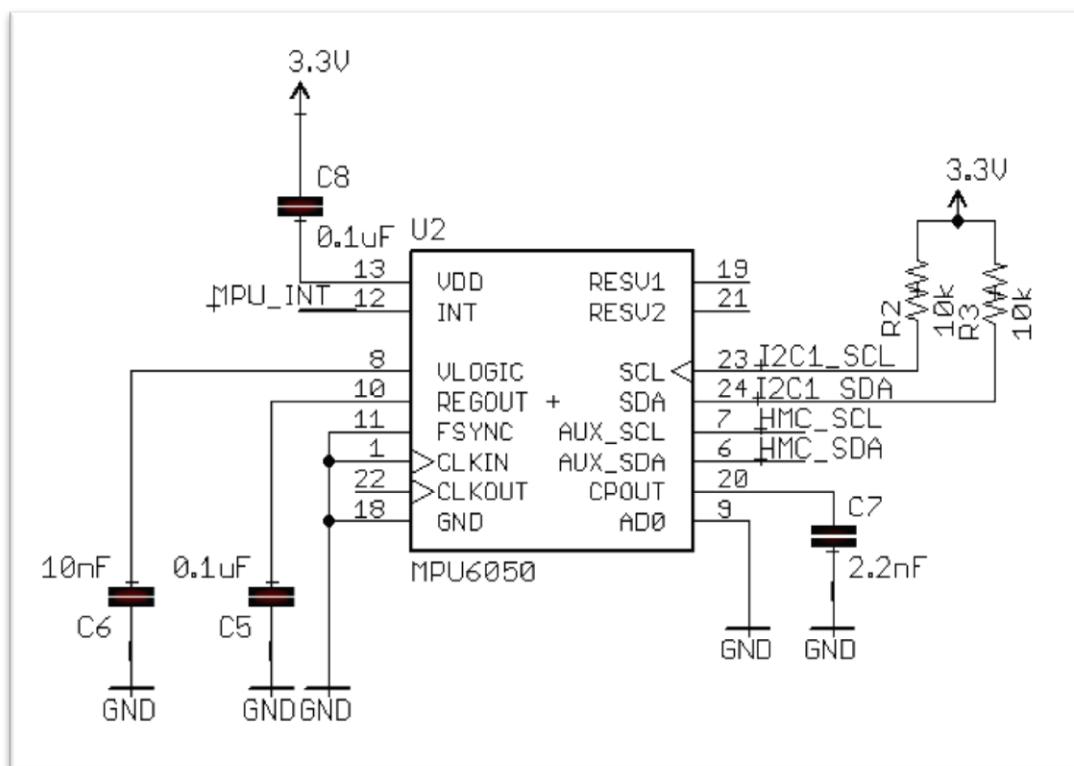
Este proyecto puede dar lugar a grandes desarrollos, ya sea con los datos que pueda ofrecer su finalidad, la representación gráfica y numérica de los movimientos, en cuestiones de deporte, fisioterapia o en el mundo de los videojuegos, como la posible ampliación de este como por ejemplo recrear movimientos en un drone humanoide en una posible guerra u otros proyectos ya que las posibilidades son inimaginables dado que parte de la evolución de la tecnología está girando entorno a la interacción hombre-máquina y este es un ejemplo de que ya es posible no trabajar con las maquinas sino vestir las maquinas.

Como conclusión final, se puede decir que después de este proyecto se ha demostrado que tanto los sensores, como Arduino o Unity tienen posibilidades de desarrollo muy grandes y que este proyecto no refleja ni una mínima parte de lo que se puede llegar a hacer con esta combinación.

## 5. Diagramas

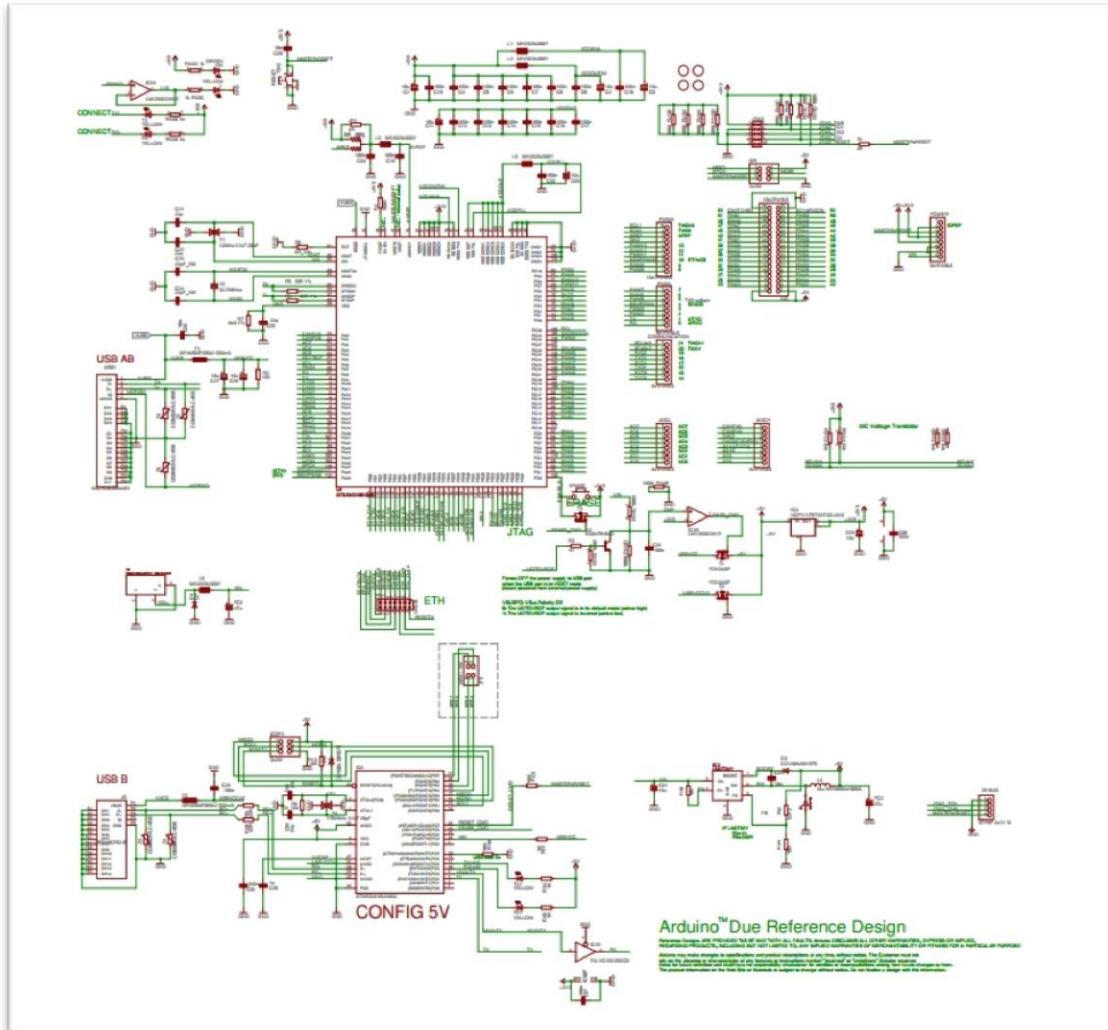
A continuación se muestran los diagramas electrónicos de los componentes electrónicos utilizados a lo largo del proyecto.

### Sensor MPU6050



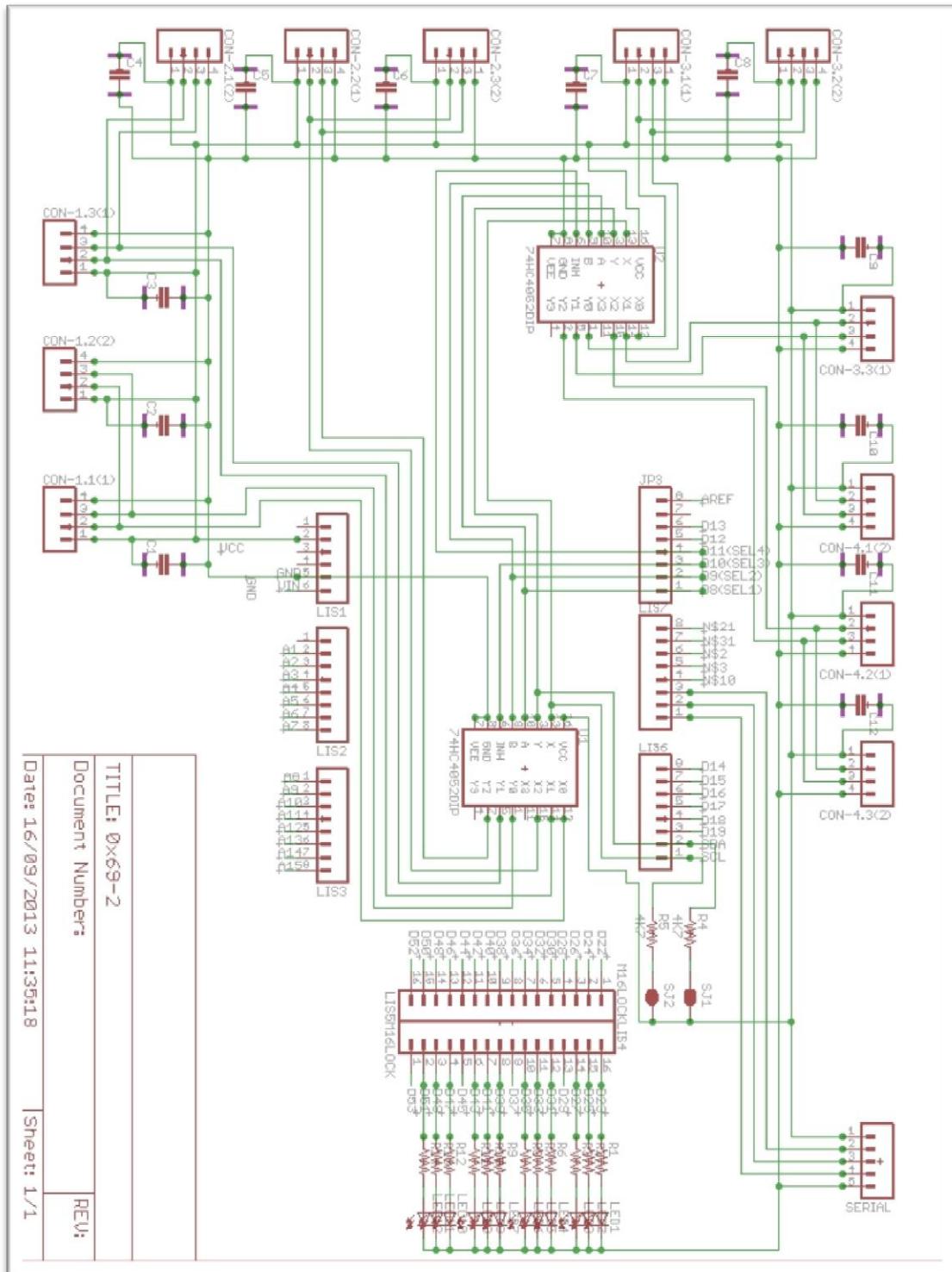
**Diagrama 7.** MPU6050: Esquema de funcionamiento del sensor MPU6050 [1].

## Arduino Due

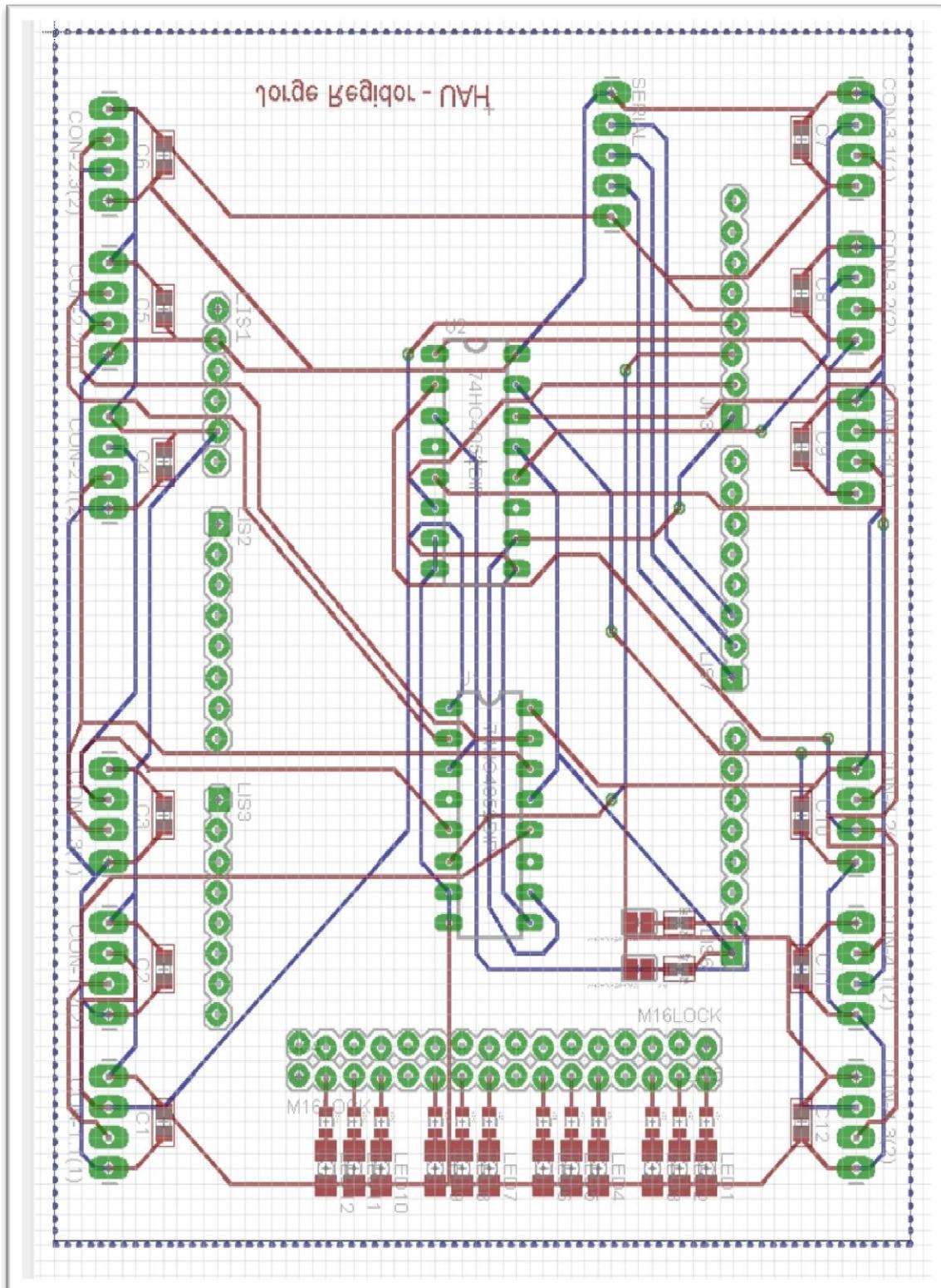


**Diagrama 8.** Arduino Due: Esquema de funcionamiento de la Arduino Due , se puede ver con mayor calidad en la pagina web de Arduino[23].

## Shield

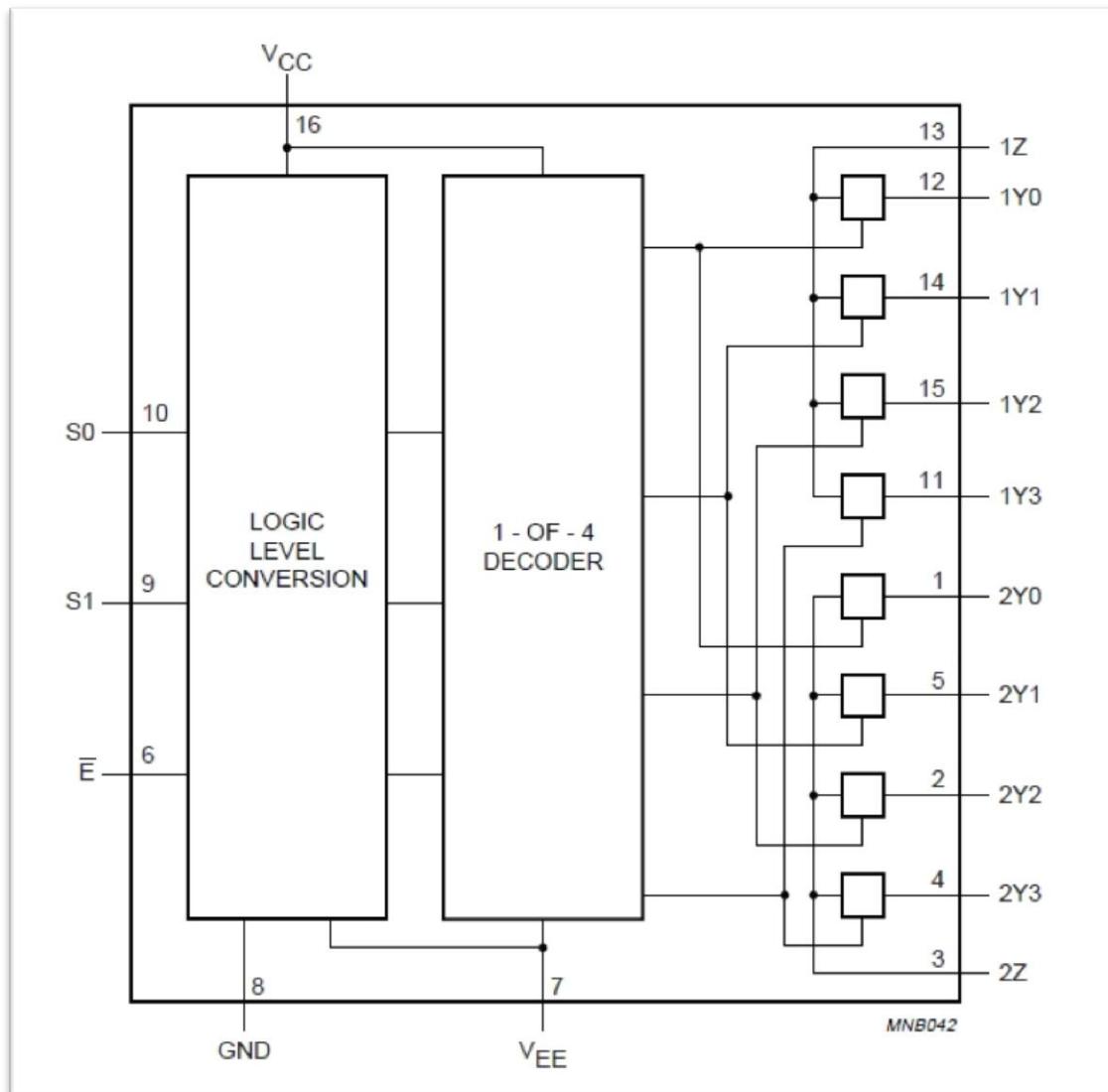


**Diagrama 9.** Shield.sch: Esquema de funcionamiento de la shield.



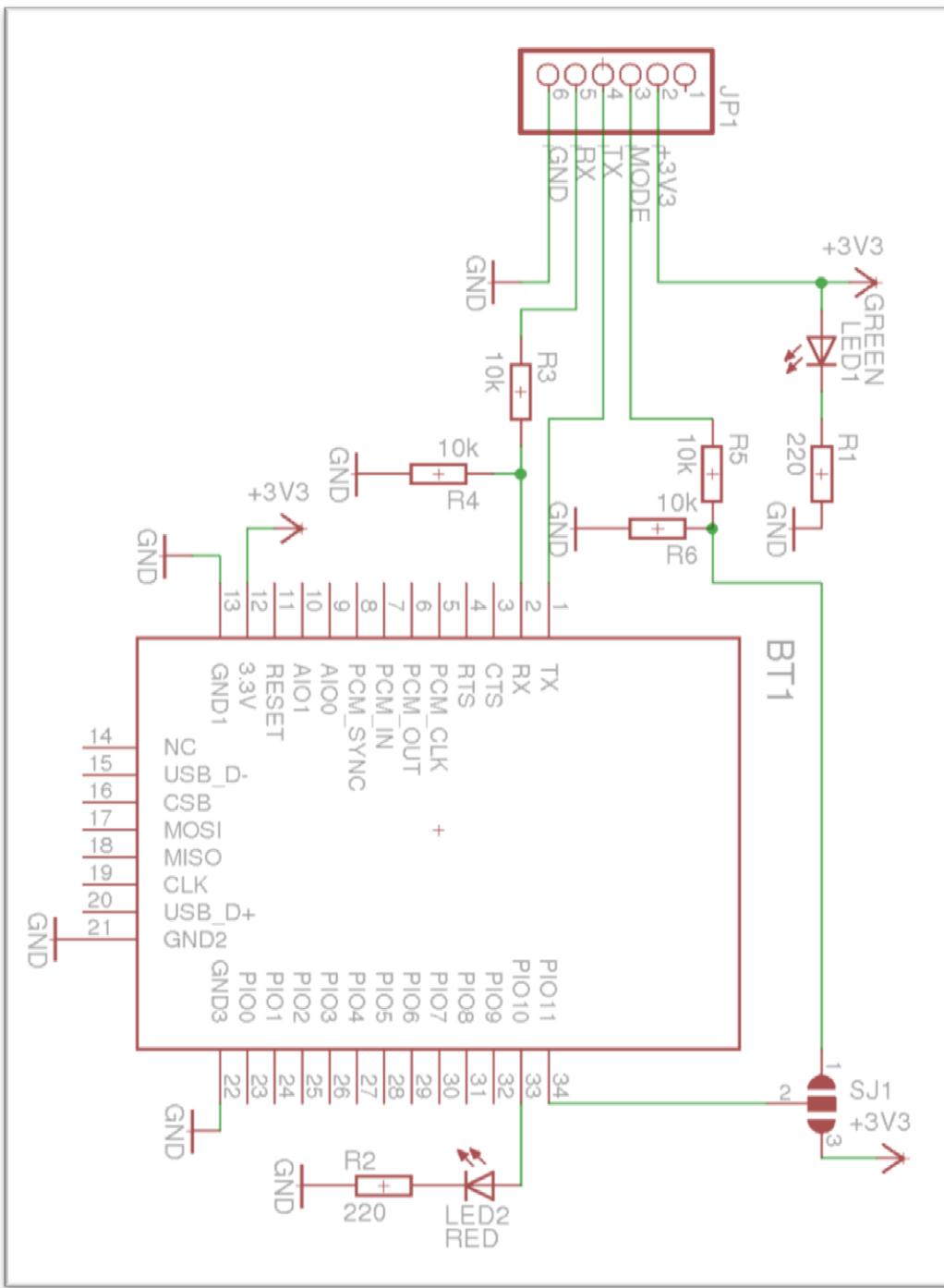
**Diagrama 10.** Shield.brd: Esquema la tarjeta shield.

74HC4052



**Diagrama 11.** 74HC4052: Esquema de funcionamiento del módulo 74HC4052.[24]

## Bluetooth



**Diagrama 12.** Placa Depeca + HC05: Esquema de funcionamiento del bluetooth. [23]

## 6. Pliego de Condiciones

### 6.1 Condiciones generales

Este proyecto está pensado para representar los movimientos corporales sobre un software de animación en 3D, por lo que el usuario del sistema deberá colocar sobre su cuerpo diversos componentes electrónicos que implica una seguridad necesaria. Se basa en una serie de reglas básicas de seguridad que aseguren la integridad física del Usuario:

- El proyecto no debe recibir grandes golpes.
- El proyecto no debe someterse a altas temperaturas.
- El proyecto no debe mojarse.
- La batería nunca debe someterse a altas temperaturas ni dejarla al sol.
- En caso de reemplazar la batería solo utilizar otra con las mismas características
- En caso de que se funda el fusible nunca conectar la batería sin él, reemplazarlo por otro.
- No utilizar cables inadecuados o en mal estado para realizar las conexiones

Si se cumplen las normas básicas descritas anteriormente se asegura el correcto funcionamiento y la seguridad del usuario

### 6.2 Condiciones Técnicas

Para el correcto funcionamiento del sistema se requiere unas especificaciones mínimas técnicas:

- Se requiere de un ordenador actual con potencia relativamente alta
- Se requiere de Window7
- La batería no debe cargarse con un cargador no adecuado.
- Los cables no pueden ser de un largo superior a 1,5 metros o posiblemente la comunicación puede tener errores.
- Si fuera necesario reemplazar uno de los sensores debería ser previamente calibrado para su correcto funcionamiento

### 6.3. Condiciones Económicas

El proyecto será financiado por parte privada librando a la Universidad de ningún tipo de pago. El coste del sistema no debe superar en ningún momento los 500 Euros, incluyendo todo el material electrónico descrito en el presupuesto, pero exceptuando la máquina que represente los movimientos en 3D y las licencias.

## 7. Presupuesto

### 7.1 Materiales

Seguidamente se muestra la tabla con los precios por unidad y totales de los materiales utilizados en el proyecto

Concepto	Precio Unidad	Cantidad	Precio total
Arduino Due	50€	1u	50€
Sensor MPU6050	20€	12u	240€
Conectores macho	0,10€	30u	3€
Conectores hembra	0,10€	30u	3€
Cable	0,30	10m	3€
Módulo 74HC4052	0,35	2u	0,70€
Tarjeta Shield	29€	1u	29€
Leds	0,3	15u	0,45
Resistencias	0,3	15u	0,45
Bluetooth	15€	2u	30€
Batería Litio 7,4V 2C	19€	1u	19€
Traje (goma + velcro)	10€	1u	10€
USB to UART	15€	1u	15€
MacBook Pro 15" (4 años)	2200€	6 meses	275€
		<b>TOTAL</b>	<b>705,6€</b>

### 7.2 Mano de obra

Seguidamente se muestra la tabla con las horas implicadas en el proyecto y su coste.

Concepto	Cantidad	Precio total
Investigación	100	3500€
Diseño	50	1750€
Construcción	50	1750€
Programación	150	5250€
Documentación	50	1750€
	<b>TOTAL</b>	<b>14000€</b>

### 7.3 Licencias

Seguidamente se muestra la tabla con las licencias necesarias para el desarrollo del proyecto.

Concepto	Cantidad	Precio total
Windows 7 (4 años)	6 meses	10,20€
Unity 3D (4 años)	6 meses	138,70€
Eagle	1u	0€
MakeHuman	1u	0€
Arduino IDE	1u	0€
MonoDevelop	1u	0€
	<b>TOTAL</b>	<b>148,90€</b>

### 7.4 Coste Total

Seguidamente se muestra la tabla con los precios totales de todo lo necesario como materiales, mano de obra y licencias utilizadas en el proyecto.

Concepto	Precio total
Materiales	705,6€
Mano de obra	14000€
Licencias	148,90€
Otros	100€
IVA (18%)	2691,81€
	<b>TOTAL</b>
	<b>17646,31€</b>

**El precio total del proyecto ascendería a 17646,31€**

## 8. Manual de Usuario

# Manual de Instrucciones

Representación en 3D de movimientos corporales capturados a través  
de acelerómetros y giroscopios

Jorge Regidor Martín - UAH

## Índice

1. Pliego de condiciones
2. Materiales
3. Conexiones
4. Ponerse el traje
5. Colocar los sensores
6. Abrir el Programa
7. Iniciar la representación
8. Interfaz de usuario
9. FAQ

## 1. Pliego de Condiciones

El sistema necesita una serie de requisitos y condiciones mínimas para asegurar que trabaje correctamente y no se estropee ni actúe de forma incorrecta:

### REQUISITOS

- Ordenador con Windows 7.
- Cargador para Baterías de 2 Celdas.

### Normas de Seguridad

- El hardware no debe recibir grandes golpes.
- El hardware no debe someterse a altas temperaturas.
- El hardware no debe mojarse.
- La batería no debe cargarse con un cargador inadecuado.
- La batería nunca debe someterse a altas temperaturas ni dejarla al sol.
- En caso de reemplazar la batería solo utilizar otra con las mismas características
- En caso de que se funda el fusible que conecta la batería nunca conectar la batería sin él, debe reemplazarlo por otro.

El uso indebido del sistema o el no cumplimiento de las normas de seguridad, puede suponer un gran peligro ya que está compuesto por piezas inflamables. Se ruega no incumplirlas y seguir las instrucciones al pie de la letra para su propia seguridad.

## 2. Materiales

El sistema se compone de una serie de materiales y/o componentes en los que todos son necesarios para su correcto funcionamiento, dentro de la caja debe haber:

- Una tarjeta Arduino Due.
- Una Shield (color verde) que encaja encima de la Arduino.
- 4 Bolsas con sensores (en cada bolsa debe haber 3 sensores con sus cables)
- 1 Bolsa solo con cables
- 1 Batería 7,4V 2C
- 2 Modulos Bluetooth
- 1 Receptor
- 12 gomas con velcros
- 1 Chaleco de goma elástica

### 3. Conexiones

#### Coneectar Arduino Due y la Shield

Coja la tarjeta de Arduino Due (azul) y la shield (verde) y comprobará que uno tiene unos orificios en la parte superior y la otra unos pines en la parte inferior, una las dos tarjetas de la única forma posible, haciendo encajar todos los pines con sus orificios. Hágalo despacio y con tranquilidad ya que no se debe doblar ningún pin, ya que sino el funcionamiento podría no ser correcto.

#### Coneectar los sensores

Los sensores ya vienen conectados a un cable, el cual puede ser cambiado, que al otro lado tiene un conector hembra de 4 pines, fijándose en el numérico que hay en la parte posterior de los sensores y los números que hay encima de los conectores de la shield, conéctelos en este orden

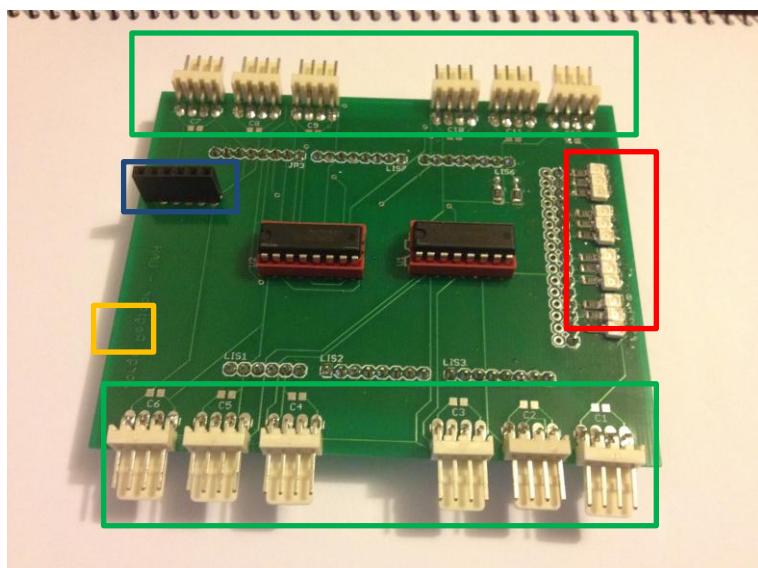
- Sensor #0 = Conexión C1
- Sensor #1 = Conexión C2
- Sensor #2 = Conexión C3
- Sensor #3 = Conexión C4
- Sensor #4 = Conexión C5
- Sensor #5 = Conexión C6
- Sensor #6 = Conexión C7
- Sensor #7 = Conexión C8
- Sensor #8 = Conexión C9
- Sensor #9 = Conexión C10
- Sensor #A = Conexión C11
- Sensor #B = Conexión C12

### Conecitar los bluetooth

El Bluetooth tiene 6 pines, lo conectamos a la tarjeta shield mediante el cable que pequeño con 6 pines que viene con uno de los modulo y el segundo se conecta al receptor y este a su vez por bluetooth al ordenador.

### Conectamos la Batería

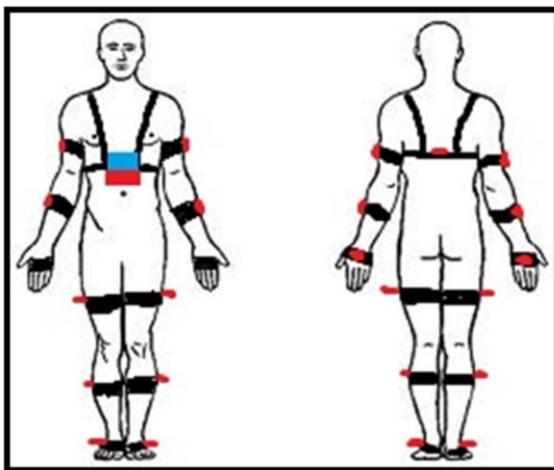
Debe conectar el conector Jack Power que sale de la batería al conector Jack Power hembra que se encuentra a la izquierda de la tarjeta Arduino Due. Desde el momento en el que realice esta conexión debe tener cuidado porque ya la electricidad empieza a correr.



Sensores - Bluetooth - Alimentación - Leds

## 4. Ponerse el traje

El traje se compone de 5 partes diferenciables: El chaleco y las extremidades.  
Las tres piezas de cada extremidad van unidas.



### El chaleco

El chaleco se diferencia por ser la pieza más grande. Una vez la hayas encontrado busca que en el centro tiene un bolsillo, ese bolsillo va a la espalda y por fuera, es decir no tocaría el cuerpo. Cuando encontremos la posición nos ponemos el chaleco y lo aseguramos con el velcro que se encuentra en la parte delantera derecha.

### Extremidades

Las extremidades podemos diferenciarlas por el tamaño de las gomas, las más grandes pertenecen a las piernas y las más pequeñas a los brazos.

En los brazos debemos colocar la primera y más grande, a la altura del bíceps con el bolsillo en la parte exterior con la apertura hacia arriba, el segundo debemos colocarlo en la mitad del antebrazo con el bolsillo en la parte exterior con la apertura hacia arriba y el ultimo debe ir en la mano, en la palma de la mano, con el bolsillo en la parte superior de la mano y la apertura hacia la muñeca. El traje no diferencia entre brazo derecho o brazo izquierdo por lo que no hay problema de uno u otro.

En las piernas hacemos prácticamente lo mismo, colocamos uno en la mitad del muslo con el bolsillo en la parte exterior de la pierna y la apertura orientada hacia arriba, el segundo entre la rodilla y el tobillo, con el bolsillo hacia la parte exterior de la pierna con la apertura orientada hacia arriba y el ultimo lo colocamos rodeando el pie, pasando por la suela y el empeine, dejando el bolsillo en la planta del pie con la apertura orientada hacia afuera.

## 5. Colocar los sensores

Los sensores están divididos en grupos de 3 para conectar cada grupo en extremidades.

### Brazo Derecho

Los tres primeros sensores, del 0 al 2, pertenecen al brazo derecho, simplemente hay que meter el sensor en el bolsillo con el cable hacia arriba y el orden de las conexiones debe de ser:

- Brazo Derecho      -> Sensor 0
- Antebrazo Derecho      -> Sensor 1
- Mano Derecha      -> Sensor 2

### Brazo Izquierdo

Los tres siguientes sensores, del 3 al 5, pertenecen al brazo izquierdo, simplemente hay que meter el sensor en el bolsillo con el cable hacia arriba y el orden de las conexiones debe de ser:

- Brazo Izquierdo      -> Sensor 3
- Antebrazo Izquierdo      -> Sensor 4
- Mano Izquierda      -> Sensor 5

### Pierna Izquierda

Si seguimos el orden, los tres siguientes sensores, del 6 al 8, pertenecen a la pierna izquierda y el orden de las conexiones debe de ser:

- Pierna Izquierda      -> Sensor 6
- Espinilla Izquierda      -> Sensor 7
- Pie Izquierdo      -> Sensor 8

Como ya habrás visto los sensores disponen de unas escuadras para que al introducirlos en los bolsillos estos queden paralelos al suelo y siempre con el cable hacia atrás.

#### Pierna Derecha

Los dos siguientes sensores, pertenecen a la pierna derecha, la forma de ponerlos en el traje es exactamente igual que en la pierna izquierda, paralelos al suelo y con el cable hacia tras, el orden de las conexiones debe ser:

- Pierna Derecha      -> Sensor 9
- Espinilla Derecha      -> Sensor A

#### Tronco

En la misma bolsa que los sensores de la pierna derecha se encuentra el ultimo sensor, este debe ir colocado en la parte de atrás del chaleco, introduciendo parte de la escuadra metálica en el bolsillo para hacer que el sensor quede unido a la espalda de forma que quede paralelo al suelo y con el cable hacia atrás.

- Tronco      -> Sensor B

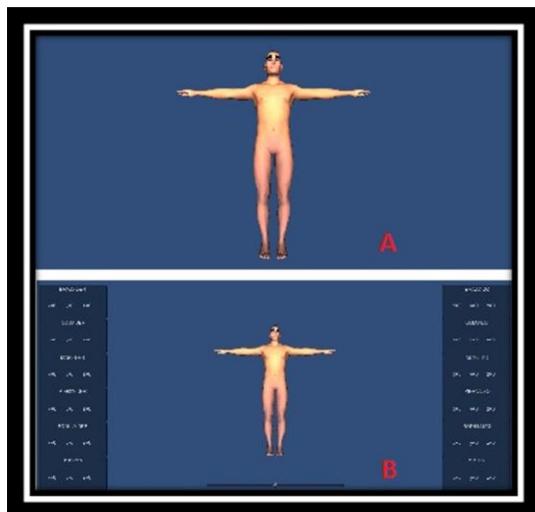
#### Tarjetas y batería

Para colocar las tarjetas debes coger la goma que se encuentra en el pecho y rodear ambas tarjetas por el centro de forma que ambas queden sujetas al pecho pero siempre que no coja ningún cable, ya que eso podría llegar a crear problemas de conexión.

La batería se conecta en el siguiente orificio de la goma que quede justo debajo de las tarjetas.

## 6. Abrir el Programa

Solo tenemos que buscar el ejecutable llamado “movimientos.exe” y ejecutarlo, nos saldrá la pantalla principal de nuestro programa.



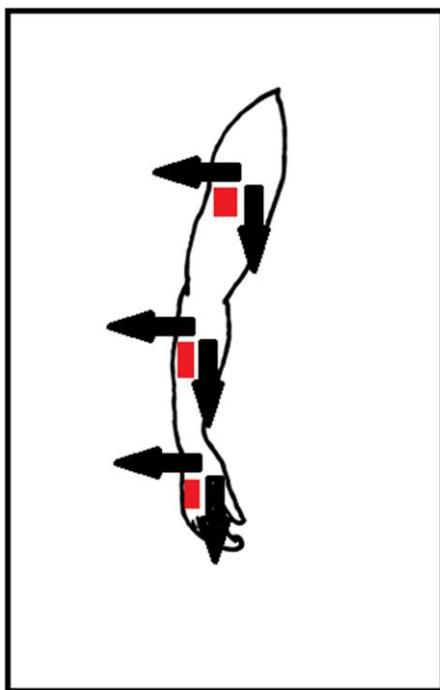
Ahora debemos comprobar que todas las luces de nuestra tarjeta están encendidas. Si algunas de las luces permanecen apagadas debemos comprobar que todos los dispositivos están bien conectados.

Una vez encendidas todas las luces lo que da señal de que todo está correcto y veremos a nuestro personaje empezar a moverse. ¿No funciona bien? Pasa al siguiente apartado.

## 7. Iniciar la Presentación

¿Has arrancado el programa los movimientos no son precisos? No te preocupes, es normal, es porque los sensores no se suelen inicializar bien, pero ahora vamos a inicializarlos correctamente.

Debes colocarte en posición vertical, completamente recto, y colocar los sensores de forma que los sensores de los brazos apunten perfectamente para abajo y se encuentren todos paralelos entre sí.



Respecto a las piernas, hacemos lo mismo, que todos los sensores estén completamente alineados mirando al frente y cuando todos los sensores este en su posición correcta, pulsamos el botón reset (se encuentra en la tarjeta de Arduino junto a la entrada de alimentación) y esperamos a que se enciendan todas las luces de la tarjeta shield.

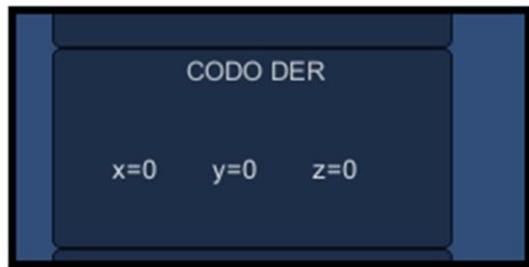
Y ya está, ya está funcionando la representación de movimientos.

## 8. Interfaz de Usuario

Ya tenemos nuestro sistema funcionando, pero ahora queremos ver que podemos hacer. Como ya se ha mostrado anteriormente, tenemos dos posibilidades: Solo ver el cuerpo de cerca o ver los datos numéricos.

### Cuerpo y datos numéricos

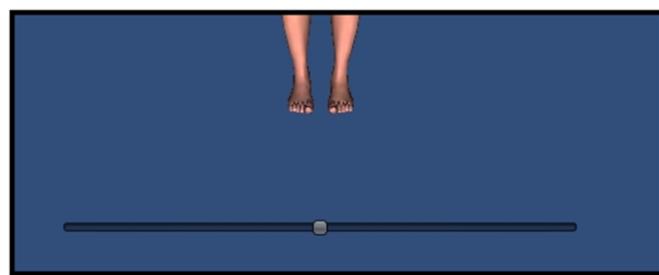
También nos encontramos un slider debajo del modelo humano (una barra que le puedes cambiar el valor) y al moverlo descubrimos que cambiamos el ángulo de visión de nuestro modelo por si necesitamos ver alguna posición en particular.



En la opción de ver cuerpo y datos numéricos, podemos ver el cuerpo rodeado de unos cuadros con datos numéricos. Esos datos numéricos, son los ángulos que dibuja nuestro cuerpo en grados. El nombre superior a los datos es el ángulo del que estamos hablando. Los números muestran la posición respecto a los tres ejes de coordenadas.

### Solo cuerpo

En esta interfaz solo veremos el cuerpo pero con un tamaño más grande y en la posición o ángulo de visión que lo hubiéramos dejado en la otra interfaz con el slider. Podemos ver los movimientos desde una perspectiva más cercana. Para volver a ver los datos y el slider no tenemos que pulsar la tecla “Espacio”.



## 9. FAQ (Preguntas Frecuentes)

### **Mi Bluetooth no se enciende**

- Si no se enciende ninguna luz de toda la placa, comprueba que la batería esté bien conectada.
- Si no se enciende ninguna luz de toda la placa y la batería está bien conectada, comprueba que el fusible no esté fundido.
- Si hay otras luces encendidas, comprueba que el bluetooth está colocado de la forma correcta.

### **Las luces de conexión no se encienden**

- Si no se enciende ninguna luz de toda la placa, comprueba que la batería esté bien conectada.
- Si no se enciende ninguna luz de toda la placa y la batería está bien conectada, comprueba que el fusible no esté fundido.
- Si solo se encienden algunas, prueba a desconectar los sensores que estén apagados y volverlos a conectar. Después pulsa Reset.
- Si alguna luz persiste apagada, prueba cambiando el cable, por uno nuevo. Después pulsa Reset.
- Si aun así persiste, prueba a quitar la alimentación y volver a conectarla.

### **¿Puedo poner un modelo humano que se parezca a mí?**

- NO, en un principio en esta versión no se puede, quizás en próximas versiones.

**Mi humano 3D hace movimientos raros**

- Comprueba que has realizado el paso de inicializar movimientos.
- Comprueba que los sensores están conectados en el número adecuado.
- Comprueba que los sensores estén colocados en la articulación adecuada.

**Tengo Windows Xp o Windows Vista.**

- Puede que funcione pero no podemos asegurar que funcione correctamente.

**Mi batería se está hinchando**

- Quítatela cuanto antes y corre, puede que explote.

## 9. Contenido del Soporte Informático

En CD está dividido en cuatro carpetas principales: Arduino, Unity, Eagle y Documentos. Están clasificadas de la siguiente forma:

- Arduino
  - Final
    - **Final.ino:** Software que se introduce en la tarjeta de Arduino Due para el funcionamiento del proyecto.
  - Calibrador
    - **Calibrador.ino:** Software que se utiliza para calibrar los sensores desde Arduino Due.
  - Receptor
    - **Receptor.ino:** Software que recibe información por un puerto serie y lo manda por otro.
  - Libraries
    - I2Cdev
      - **I2Cdev.h:** Descriptor de la librería I2Cdev.
      - **I2Cdev.cpp:** Contenedor de la librería I2Cdev.
    - MPU6050
      - Examples
        - **MPU6050\_DMP6**
          - **MPU6050\_DMP6.ino:** Ejemplo de funcionamiento del DMP del sensor MPU6050
        - **MPU6050\_raw**
          - **MPU6050\_raw.ino:** Ejemplo de extracción de los valores Raw del sensor MPU6050
      - **Helper\_3dmath.h:** Librería capaz de trabajar con quaterniones y modificarlos.
      - **MPU6050.h:** Descriptor de la librería MPU6050 que da acceso al Sensor.
      - **MPU6050.cpp:** Contenedor de la librería MPU6050 que da acceso al Sensor.
      - **MPU6050\_6Axis\_MotionApps20.h:** Librería que da acceso al DMP del sensor MPU6050.
      - **MPU6050\_9Axis\_MotionApps41.h:** Librería capaz de trabajar con el sensor MPU6050 mas una brújula.

- Unity

- **Movimientos.exe:** ejecutable que contiene el programa principal.
- Proyecto\_1
  - Asset
    - Brazo\_der
      - **Brazo\_der.cs:** Script que controla el Brazo Derecho.
      - **Codo\_der.cs:** Script que controla el Codo Derecho.
      - **Mano\_der.cs:** Script que controla la Mano Derecha.
    - Brazo\_izq
      - **Brazo\_izq.cs:** Script que controla el Brazo Izquierdo.
      - **Codo\_izq.cs:** Script que controla el Codo Izquierdo.
      - **Mano\_izq.cs:** Script que controla la Mano izquierda.
    - Pierna\_izq
      - **Pierna\_izq.cs:** Script que controla la Pierna Izquierda.
      - **Rodilla\_izq.cs:** Script que controla la Rodilla Izquierda.
      - **Pie\_izq.cs:** Script que controla el Pie Izquierdo.
    - Pierna\_der
      - **Pierna\_der.cs:** Script que controla la Pierna Derecha.
      - **Rodilla\_der.cs:** Script que controla la Rodilla Derecha.
    - Cuerpo
      - **Tronco.cs:** Script que controla el tronco.
    - Escena\_funciona.usf: archivo que controla todos los elementos anteriores.
    - Gui.cs: Script que se comunica con Arduino y muestra los datos por pantalla.
  - Library: Carpeta con Librerías predefinidas por Unity.
  - ProjectSettings: Carpeta con archivos de configuración de proyecto.
  - Temp: Carpeta contenedora de Archivos temporales del proyecto.
  - **Otros Archivos:** otros archivos del proyecto Unity.

- Eagle
  - **0x69.sch:** *Diseño del esquema de la shield*
  - **0x69.brd:** *Diseño de la placa shield.*
  - Gerber: *Carpeta que contiene los archivos para la fabricación de la placa.*
- Documentos
  - Datasheet: *Carpeta con los Datasheet de los elementos utilizados.*
  - **Memoria.pdf:** *Archivo PDF con esta memoria.*
  - **Manual.pdf:** *Archivo PDF con el manual de usuario.*

## 10. Bibliografía

- [1] Invensense [on-line] [consulta: 15 Agosto 20013]. Disponible en: <<http://www.invensense.com>>.
- [2] MoCap - Tecnologías de captura de movimiento [on-line] [consulta: 15 Agosto 2013]. Disponible en: <<http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Peliculas/Mocap/tecnol.htm>>.
- [3] iNGENET Bitácora | ¿Te has preguntado cómo funciona el sensor de Kinect? [on-line] [consulta: 16 Agosto 2013]. Disponible en: <[http://bitacora.ingenet.com.mx/2012/12/%C2%BFte-has-preguntado-como-funciona-el-sensor-de-kinect](http://bitacora.ingenet.com.mx/2012/12/%C2%BFte-has-preguntado-como-funciona-el-sensor-de-kinect/)>.
- [4] pTolomeo Mexico [on-line] [consulta: 16 Agosto 2013]. Disponible en: <<http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/213/A11.pdf?sequence=11>>.
- [5] Carlos Giménez. Teoría y Aplicaciones de la Informática 2. Trabajo Práctico (Ingeniería Electrónica). Asunción, Paraguay. Universidad Católica Nuestra Señora de Asunción , Facultad de Ciencias y Tecnología, 2009.
- [6] David Fernando Pozo Espín. Diseño y construcción de una plataforma didáctica para medir ángulos de inclinación usando sensores inerciales como acelerómetros y giróscopos. Proyecto Final de Carrera (Ingeniería en Electrónica). Quito, Ecuador. Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, 2010.
- [7] Google [on-line] [consulta: 11 Septiembre 2013]. Disponible en: <<http://www.google.com>>.
- [8] Definición de Efecto coriolis por el diccionario gratuito de Babylon [on-line] [consulta: 11 Septiembre 2013]. Disponible en: <[http://diccionario.babylon.com/efecto\\_coriolis](http://diccionario.babylon.com/efecto_coriolis/)>.
- [9] Anónimo. Sistemas de coordenadas y sistemas de referencia. Transparencias (Ingeniería Agrónoma). Madrid, España. Universidad Politécnica de Madrid, Departamento de Física y Matemáticas, 2013.
- [10] Xataka Ciencia: La ciencia de forma sencilla [on-line] [consulta: 02 Octubre 2013]. Disponible en: <<http://www.xatakaciencia.com/sabias-que/que-es-un-sistema-de-navegacion-inercial>>.
- [11] Lic Adriana Favieri. Introducción a los Cuaterniones. Tutorial (Ingeniería Aeronáutica). Buenos Aires, Argentina. Universidad Tecnológica Nacional, Facultad Regional Haedo, 2008.

- [12] Ogress - Quaterniones y Rotación [on-line] [consulta: 9 Octubre 2013]. Disponible en: <<http://ogress.wikispaces.com/Quaterniones+y+Rotaci%F3n>>.
- [13] WIKIPEDIA, la enciclopedia libre [on-line] [consulta: 9 Octubre 2013]. Disponible en: <<http://es.wikipedia.org/wiki/Vector>>.
- [14] NASA Rocket Rotacion [on-line] [consulta: 10 Octubre 2013]. Disponible en: <<http://www.grc.nasa.gov/WWW/k-12/rocket/rotations.html>>.
- [15] EularQuats [on-line] [consulta: 12 Octubre 2013]. Disponible en: <<http://www.anticz.com/eularqua.htm>>.
- [16] Cuaterniones Trabajo de Clase [on-line] [consulta: 12 Octubre 2013]. Disponible en: <<http://www.scribd.com/doc/64060691/CUATERNIONES-Trabajo-de-Clase-1>>.
- [17] El bus I2C [on-line] [consulta: 12 Octubre 2013]. Disponible en: <<http://www.comunidadelectronicos.com/articulos/i2c.htm>>.
- [18] ¿Cómo funciona una resistencia pull-up? | eHow en Español [on-line] [consulta: 14 Octubre 2013]. Disponible en: <[http://www.ehowenespanol.com/funciona-resistencia-pullup-como\\_167886](http://www.ehowenespanol.com/funciona-resistencia-pullup-como_167886)>.
- [19] Comunicación Serial: Conceptos Generales - National Instruments [on-line] [consulta: 14 Octubre 2013]. Disponible en: <<http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1>>.
- [20] Manuel Sánchez Rubio. Tecnología Bluetooth. Transparencias (Ingeniería Informática). Alcalá de Henares, España. Universidad de Alcalá, Departamento de Ciencias de la Computación, 2009.
- [21] Arduino - HomePage [on-line] [consulta: 30 Octubre 2013]. Disponible en: <<http://www.arduino.cc/es/>>.
- [22] Arduino - ArduinoBoardDue [on-line] [consulta: 30 Octubre 2013]. Disponible en: <<http://arduino.cc/en/Main/arduinoBoardDue>>.
- [23] Taller DEPECABOT 3.0 UAH [on-line] [consulta: 3 Noviembre 2013]. Disponible en: <[http://geiser.depeca.uah.es/moodle\\_robot/course/view.php?id=40](http://geiser.depeca.uah.es/moodle_robot/course/view.php?id=40)>.
- [24] Philips Semiconductors. Datasheet 74HC4052. 2004
- [25] Aviones de radio control, helicópteros, coches, barcos, FPV y Quadcopters - HobbyKing [on-line] [consulta: 3 Noviembre 2013]. Disponible en: <<http://www.hobbyking.com>>.
- [26] Unity – Motor de juego, Herramientas y Multiplataforma [on-line] [consulta: 4 Noviembre 2013]. Disponible en: <<http://spanish.unity3d.com/unity>>.

- [27] MonoDevelop [on-line] [consulta: 4 Noviembre 2013]. Disponible en: <<http://www.monodevelop.com>>.
- [28] MakeHuman Mesh Licence - MakeHuman [on-line] [consulta: 5 Noviembre 2013]. Disponible en: <[http://makehuman.org/doc/node/makehuman\\_mesh\\_license.html](http://makehuman.org/doc/node/makehuman_mesh_license.html)>.
- [29] CadSoft EAGLE PCB Design Software - EAGLE Support, Tutorials, Shop [on-line] [consulta: 5 Noviembre 2013]. Disponible en: <<http://www.cadsoftusa.com>>.
- [30] MPU-6050 6-axis accelerometer/gyroscope | I2C Device Library [on-line] [consulta: 5 Noviembre 2013]. Disponible en: <<http://www.i2cdevlib.com/devices/mpu6050#captures>>.
- [31] Giroscopio Digital visto al Microscopio | Microsiervos [on-line] [consulta: 5 Diciembre 2013]. Disponible en: <<http://www.microsiervos.com/archivo/tecnologia/giroscopio-digital-microscopio.html>>
- [32] Quaterniones [on-line] [consulta: 8 Diciembre 2013]. Disponible en : <[http://catarina.udlap.mx/u\\_dl\\_a/tales/documentos/lis/munoz\\_r\\_o/appendiceB.pdf](http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/munoz_r_o/appendiceB.pdf)>