Universidad Zaragoza
1542

Trabajo Fin de Grado

DESARROLLO DE UN SISTEMA DE CONTROL PARA JUEGOS INMERSIVOS BASADO EN EL MANDO DE LA PSMOVE

Autor/es

Adrián Lázaro Cirajas

Director/es

José Ramón Beltrán Blázquez

Escuela de Ingeniería y Arquitectura

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

2020

Contenido

1.Introducción

1.1 Resumen

Los dispositivos de interacción para los ordenadores están en constante crecimiento, se han desarrollado diferentes alternativas a la hora de interactuar con ellos, se empezó con el ratón y el teclado hasta hoy en día donde tenemos reconocimiento de voz, realidad virtual y aumentada.

Esto nos lleva a que podemos utilizar estos dispositivos en el ámbito que queramos, pero en este proyecto nos centraremos en el ámbito de los videojuegos ya que podemos aprovechar todas sus características de interacción gracias a las herramientas existentes hoy en día.

En este proyecto mencionaremos los distintos dispositivos de interacción y estudiaremos con profundidad el mando PSMove que acompañado con la cámara detecta la orientación y movimiento que se podrá replicar y mostrar por pantalla.

Esto es posible gracias a diferentes librerías existentes que permiten enlazar el mando PSMove con un ordenador para poder utilizarlo como elemento de control de juegos interactivos. En este proyecto se propone el estudio y la implementación del mando PSMove en un entorno de juegos basado en ordenador.

Al final del proyecto mostraremos algunos prototipos para saber hasta qué nivel nos puede ofrecer PSMove en el entorno de videojuegos con el programa Unity.

1.2 Objetivo

El objetivo del proyecto es el desarrollo de un sistema de control para juegos interactivos inmersivos basado en el mando de la PSMove.

Para tener un correcto funcionamiento necesitamos cumplir una serie de objetivos.

Para ello será necesario realizar tanto la conexión del mando a un PC en Windows como la interacción con la cámara para conocer la orientación y el posicionamiento del mando, para ello necesitaremos conocer y estudiar las principales herramientas que nos ofrece hoy en día PSMove en Windows.

Toda la información de nuestro mando puede ser configurada, controlada y almacenada a través de dos librerías llamadas PsmoveApi y PsmoveService, deberemos estudiar sus códigos y qué partes nos interesan para nuestro proyecto (funciones, parámetros, conectividad…) y así trasladarlas a Unity ya que nuestro mando se deberá integrar en el entorno de desarrollo de juegos Unity.

Por último, se realizarán varios prototipos de juegos en Unity para demostrar la capacidad de interacción y la respuesta del mando en entornos inmersivos 3D.

2.Evolución Tecnológica

2.1 Evolución

En este apartado, se volverá la vista atrás para ver cómo han cambiado los dispositivos de interacción en los videojuegos. Se empezará viendo cómo eran los mandos que se utilizaban para jugar a los primeros videojuegos y se finalizará explorando la tecnología más novedosa que se usa hoy en día. No se van a cubrir todos los dispositivos desarrollados ya que la lista se volvería muy extensa, sin embargo, se van a mencionar los que se han considerado más importantes en la evolución de la industria.

El primer mando que sale al mercado fue el VCS (Figura 2.1) en 1977, es un Jockstick que contaba con un único botón de acción y 4 direcciones de entrada [1]. Este mando se desarrolló para tener un control sobre los juegos de la consola Atari2600 (Figura 2.2), se conectaba mediante un conector DE.9 a la consola, era un mando digital, no analógico, y su diseño hizo que los jugadores de la consola pudieran manejarlo apoyado sobre la mesa (las ventosas aún no se habían aplicado a este tipo de periféricos) o bien cogerlo con una mano mientras se accionaba el Jockstick con la otra.

Figura 2.1 VCS Atari 2600

Figura 2.2 Consola Atari 2600

A partir de 1980 comienza la era de los D-Pad (Figura 2.3) que consiste en un botón digital, a menudo en forma de cruz que permite controlar el movimiento en 4 direcciones distintas [2].

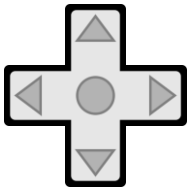


Figura 2.3 Botón D-Pad

En la década de los 90 todo cambia ya que empezaron hacer juegos más complejos y con más opciones de control. Para ello se necesitaban más botones y la solución que revolucionó el mercado fue el mando SNES (Super Nintendo Entertainment System™) (Figura 2.4). Este mando disponía de dos filas de botones que acabarían convirtiéndose en la norma del mercado con los cuatro que ahora suelen utilizarse. Lleva incorporado

un cable USB para la conexión y fue el primer mando que podía conectarse al ordenador sin el uso de un adaptador, Además, dispone de dos gatillos en la superficie de arriba que resultan imprescindibles en los mandos actuales [3].

Figura 2.4 La Snes (1990)

Aunque la Virtual Boy (Figura 2.5) acabara siendo un fracaso absoluto de ventas y de crítica, hubo innovaciones claras en aquella consola que quiso acercar a los usuarios a esa versión primitiva de las experiencias de realidad virtual [4].
Ese mando era especialmente llamativo por una circunstancia: constaba de una configuración en espejo**,** con dos D-Pads y dos pares de botoneras "A" y "B" que hacían que el jugador pudiera controlar estos juegos de forma más compleja, en la parte de atrás dispone de una batería formada por 6 pilas triple AAA y un cable de tipo DE-6 (Figura 2.6). Esa opción sería la antecesora de los dos mandos analógicos que hoy son un clásico en los mandos de las consolas.

Figura 2.5: Mando Virtual Boy

Figura 2.6: Mando parte de atrás

Nintendo en 1996 creó un mando (Figura 2.7) con conexionado por cable USB, que por primera vez hacía uso de un stick analógico para que Mario se pudiera mover en 360°, además de controlar la velocidad del personaje presionando más o menos fuerte la palanca de ese stick.

Aquel formato de control supuso un cambio general no solo para la consola, sino para todo un género de videojuegos que cobró (y nunca mejor dicho) otra dimensión. Aquellos cambios se verían reflejados en otros géneros y otros diseños posteriores**,** ya que a partir de entonces las industrias no están dejando de fabricar mandos con Jockstick analógicos [5].

Figura 2.7: Mando Nintendo 64

Cuando parecía que todo estaba inventado, en 1997 apareció la PlayStation. El dualshok2 (Figura 2.8) fue el primer mando para playStation1 incluía dos sticks analógicos, botones de dirección, ocho botones y un conector único para conectarlo tanto en la PlayStation1 y PlayStation2 [6].

Figura 2.8 Mando dualshok2

En 2006 se lanzó la PlayStation 3 y llegó una importante revolución en el aspecto de controladores para consolas de Sony. Esa consola llevaba el Dualsock3 que tenía la función de detección del movimiento y vibración. A este primer mando se le llamó Sixaxis, nombre que significa «seis ejes» haciendo referencia a los seis (six) ejes (axis) de detección de movimiento (3 para movimientos posicionales en el espacio mediante acelerómetros y 3 para la detección de rotación). Más tarde, debido a las críticas de los usuarios hacia la falta de vibración, se hizo una revisión del mando con el nombre DualShock3 (Figura 2.9), que añade la función de vibración de nuevo al mando [7].

Las principales características de este mando son:

- Tiene función inalámbrica a través de Bluetooth, con batería de litio de aproximadamente 30 horas de autonomía, cargándose a través de un cable mini-USB. Además, se puede conectar a la consola a través del mismo cable, sin necesidad de usar la función inalámbrica.
- Los botones R2 y L2 son ahora gatillos analógicos y se ha añadido un botón PS en el medio
- Tiene cuatro leds en la parte delantera, que permiten saber en qué puerto está conectado.
- Se ha mejorado su sensibilidad con respecto a su predecesor.

Figura 2.9 DualShock3 (Parte delantera)

Figura 2.9 DualShock3 (Parte Trasera)

El DualShock 4 (Figura 2.10) fue anunciado el 20 de febrero de 2013 y fue lanzado el 15 de noviembre. Es el primer mando de la línea DualShock en cambiar significativamente su diseño, presentando un diseño más redondeado que sus predecesores. Presenta un panel táctil parecido al usado en la PlayStation Vita. Posee un diseño más ergonómico, junto con una barra de luz LED en su parte frontal, la cual cambia de color de acuerdo al número de jugador (Jugador 1 azul, jugador 2 rojo, jugador 3 verde, jugador 4 morado), tomando los cuatro colores de los botones emblemáticos de PlayStation, además en juegos de un solo jugador se permite a los desarrolladores cambiar el color o la intensidad del LED para crear efectos relacionados con el juego, como por ejemplo cuando el jugador recibe daño. Cabe destacar que el Mando cambia a naranja cuando se conecta a cualquier dispositivo con Bluetooth o por medio de USB. Además, incluye un pequeño altavoz en el centro por arriba del botón PS, dando indicios al jugador en algunos juegos, eventos significativos por medio de un sonido como golpes, apertura de puertas, estado de vida, mejorando la experiencia de inmersión en el juego [8].

Los botones Start y Select han sido remplazados por un solo botón Options. El DualShock 4 también cuenta con un botón "Share" (Compartir) que permite al usuario hacer grabaciones de lo que juega o hacer streaming a través de Twitch, Ustream y YouTube. También el mando es compatible con la PlayStation 3 y accesible para cualquier juego.

Figura 2.10 DualSock4

En 2004 se lanzaron consolas con pantallas táctiles que era el caso de la Nintendo DS (Figura 2.11). Estas pantallas, normalmente son de 3 pulgadas, TFT LCD,256x192 píxeles con 260.000 colores diferentes [9].
Esto añadía una nueva dimensión a los videojuegos, haciendo que la complejidad pudiera subir (o no) según el desarrollo y que la interacción aprovechara esa nueva capacidad cuando lo necesitara.

Figura 2.11 Nintendo DS

En 2006 apareció en primer mando con detección de movimiento llamado Wii Remote (Figura 2.12). La comunicación entre el Wiimote y la consola se realiza a través de Bluetooth.
Dos son las tecnologías básicas en las que se basa el Wiimote: acelerómetro y un sensor óptico. El acelerómetro detecta la aceleración del mando en los tres ejes, es decir detectar hacia donde movemos el mando. El sensor óptico nos permite saber a qué punto de pantalla estamos apuntando. Para ello se ayuda de la barra de infrarrojos que se conecta a la consola y que se coloca encima o debajo del televisor. Esta barra dispone de diez LEDS que emiten infrarrojos, los cuales son recibidos por el sensor óptico que, a partir de ellos, triangula la posición.
Además, incluye en su interior un altavoz y un pequeño motor que ofrece la posibilidad de activar la vibración. Wiimote funciona con 2 pilas de tipo AA [10].

Figura 2.12 Wii Remote

Su éxito fue masivo, y de hecho tanto la PlayStation como la Xbox acabarían copiando descaradamente esa característica con Move y Kinect respectivamente.

En la Kinect, ya no era necesario el Wiimote**,** porque nuestro cuerpo y nuestras manos eran el mando. Las cámaras integradas en Kinect y un software especial hacían que el reconocimiento de gestos cobrase sentido en juegos muy del estilo de la Wii, pero la idea era que su adaptación a videojuegos más exigentes también fuera posible.
Actualmente se puede hablar de dos generaciones de Kinect, la primera desarrollada para la Xbox360 (Figura 2.13) y la segunda para Xbox One (Figura 2.14), además cada una dispone de un adaptador que hace posible su conexión a PC.
Este sensor cuenta con una cámara RGB (encargada de capturar y transmitir los datos de video a color), una cámara de profundidad y emisor de laser de infrarrojos ya que hacen posible la captación de imágenes de profundidad y micrófono MultiArray (encargado de captar la voz).

También incorpora un LED que indica si los controladores de Kinect están correctamente instalados [11].

Figura 2.13 Cámara Kinect Xbox 360

Posteriormente, en el año 2013 se lanzó la nueva consola Xbox One de Microsoft, junto a una segunda versión del sensor de movimientos (Kinect 2.0), que ofrece una serie de mejores respecto a la primera generación. Dispone de mayor resolución: 1920x1080 (Full HD), con lo que presenta un mayor detalle de imagen y una mayor calidad.
El concepto de detección de profundidad es diferente en esta nueva versión. Una única cámara de infrarrojos capta únicamente al individuo por medio de un láser. Este láser vuelve al lugar de origen, y analizando el tiempo de vuelo se puede conocer la posición del jugador [12].

Figura 2.14 Cámara Kinect Xbox One

En el interior de Kinect Xbox One, hallaremos dos PCB (Figura 2.15), dentro de estos hay varios elementos destacables.

- 1 cámara RGB de 1080p a 30 FPS
- 1 cámara de infrarrojos
- 3 infrarrojos de control remoto
- 1 procesador
- 1 memoria
- 4 fases
- 1 LED de encendido

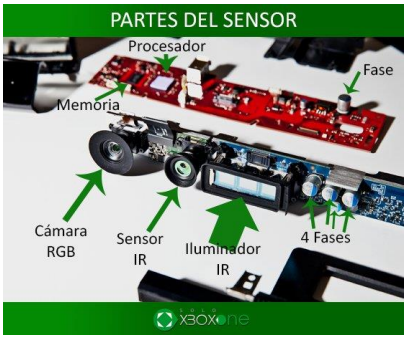


Figura 2.15: Placas de Circuito Impreso Kinect

Al mismo tiempo que Xbox Sony lanzó PSMove (Figura 2.16) donde con una cámara que registraba el movimiento de esos curiosos mandos con forma de micrófono.

Figura 2.16 Mando PSMove

La ultima genialidad fue la Nintendo Switch (Figura 2.17) que salió al mercado en 2017 permite disfrutar de la experiencia de una consola de sobremesa y de una consola portátil en un solo dispositivo. Parte de ese éxito se lo debemos a los singulares y geniales Joy-Con, los mandos que se acoplan y desacoplan a los laterales de la pantalla que cuentan con su acelerómetro y giroscopio para poder usarlos como controles por movimiento**,** además de otros elementos como el sensor infrarrojo del Joy-Con R (derecho) para ayudar al seguimiento de los gestos o para identificar objetos. En el caso de no usarlos por separado, pueden ser acoplados con accesorios como el Power A - Grip Confort para tener en las manos un mando algo más tradicional.

Figura 2.17: Nintendo Switch

Una vez presentados los controladores más importantes en la Tabla 1 se realiza una comparativa de cada uno de ellos.

Tabla 1. Comparativa de mandos de control para videojuegos

Controladores	Año	Consola	Comunicación	Sensores	Batería/Pila
Jockstick Atari 2600	1977	Atari 2600	Cable conector 9-DE	NO	-
La Snes	1990	Super Nintendo	Cable USB	NO	Pila Batería CR2032
Mando Virtual Boy	1995	Mando Virtual Boy	Cable conector 6-DE	NO	6 pilas AAA
Mando Nintendo 64	1997	Nintendo 64	Cable USB	NO	2 pilas
Dualshok2	1997	PlayStation1 /PlayStation 2	Cable de 9 pines	NO	Batería
Dualshok3	2006	PlayStation3	Bluetooth	Giroscopio Acelerómetro	Batería de litio LIP1359
Dualshok4	2013	PlayStation4	Bluetooth	Giroscopio Acelerómetro	Batería de Litio. CUH-ZCT1X
Wiimote	2006	WII	Bluetooth	Sensor óptico Acelerómetro	Batería Recargable o 2 Pilas AA
PSMove	2010	PlayStation	Bluetooth	Giroscopio Acelerómetro Magnetómetro	Batería de Litio
JoyToy	2017	Nintendo Switch	Bluetooth	Giroscopio Acelerómetro Sensor de infrarrojos	Batería original HAC-006

2.2 PSMove vs. Wiimote

En este apartado se comparan las características técnicas del principal competidor que tiene PSMove que es el mando Wii Remote [13] con el que comparte características muy semejantes.

Empezamos centrándonos en el diseño de cada uno. Wii Remote (Figura 2.18) con un peso de 180 g presenta en su cara frontal los botones "A", "1", "2", "+", "-", "HOME" y la cruz de direcciones, más un botón "POWER" para apagar la consola, algo inédito hasta entonces. En la parte anterior sólo presenta el botón "B", en un formato similar a un gatillo al igual que el trigger del mando PSMove.

Además, en su parte frontal incluye un altavoz y cuatro luces numeradas que indican el número de jugador al que corresponde tal mando y la carga de las pilas. En la parte inferior se encuentra el puerto de expansión del mando y viene unida una correa de

seguridad que se amarra a la muñeca para evitar soltar y dañar el control de forma accidental.

PSMove a diferencia de Wii Remote cuenta con un peso de 140 g, no tiene cruz de direcciones ni altavoz, pero presenta más botones en la cara frontal y lo más significativo la esfera que se encuentra en la parte superior del controlador que contiene un solo LED multicolor.

Figura 2.18 Wii Remote

Figura 2.19 PSMove

Ahora diferenciaremos los aspectos más técnicos de cada uno:

Wii Remote presenta un sensor de aceleración de 3 ejes (ADXL 330), que puede detectar el movimiento del control remoto sobre 3 ejes (Arriba-abajo, izquierda-derecha, adelante-atrás), este es el único sensor de movimiento utilizado por este controlador.

En cambio, PSMove cuenta con un acelerómetro, giroscopio y un magnetómetro conocido como sensor de campo magnético terrestre, básicamente es una brújula integrada en el mando, que utiliza el campo magnético de la Tierra para determinar la orientación del dispositivo.

Mientras que los sensores internos en el control remoto Wii y la varita Move difieren ligeramente, la tecnología de seguimiento de movimiento externo que gobierna los dispositivos varía drásticamente. La Wii se basa en el seguimiento de objetos infrarrojos utilizando sensores de luz ubicados en la parte superior del control remoto. La barra de sensores contiene un conjunto de LED infrarrojos que sirven como balizas para el sensor de la Wii. Al rastrear dónde están los LED, el sensor puede determinar hacia dónde apunta el control remoto. La "Barra de sensores" en sí misma no detecta nada, y cualquier fuente de luz infrarroja adecuada podría cumplir con sus obligaciones. Las "Barras de sensores" inalámbricas de terceros son esencialmente solo conjuntos de LED infrarrojos alimentados por baterías y perfectamente empaquetados en un rectángulo de plástico. En el software Wii, el Wii Remote envía valores de seguimiento con una resolución de 1024x768. Sin embargo, estos son datos interpolados basados en el sensor PixArt del Wii Remote, que no se pueden comparar directamente con los datos brutos del sensor visual. Aunque las especificaciones del

sensor PixArt no son públicas, los entusiastas de Wii han especulado que tiene una resolución "verdadera" aproximada de 352x288.

PlayStation Move, por otro lado, utiliza el accesorio de la cámara PlayStation Eye para seguir visualmente el movimiento de la varita, al igual que la tecnología de captura de movimiento de videojuegos y películas. La bombilla de goma suave en el extremo de la varita cubre una matriz de LED RGB que hace que la bombilla brille de diferentes colores. Cuando un juego calibra la varita, registra de qué color brilla la bombilla y usa esa información para monitorear dónde se coloca la varita y cómo está orientada. En este caso, el Eye actúa como un sensor visual y la bombilla de la varita proporciona la información de posición, lo opuesto al sistema de sensor externo del control remoto Wii. Dado que PlayStation Eye tiene una resolución máxima de 640x480 a 60Hz y 320x240 a 120Hz. Suponiendo que detecta a 60Hz, es potencialmente mucho más preciso que el control remoto de Wii.

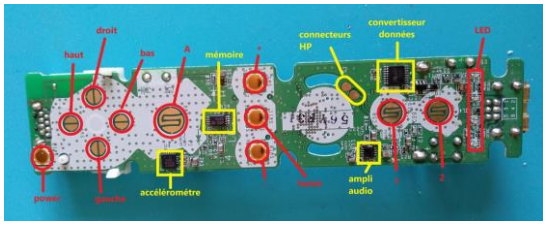


Figura 2.20 PCB Wii Remote(Delantera)

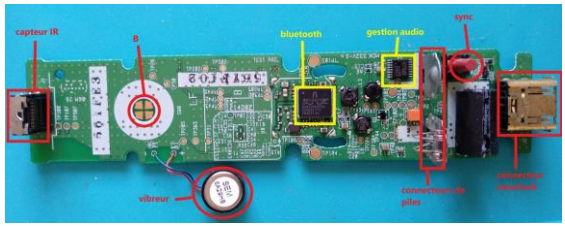


Figura 2.20 PCB Wii Remote(Trasera)

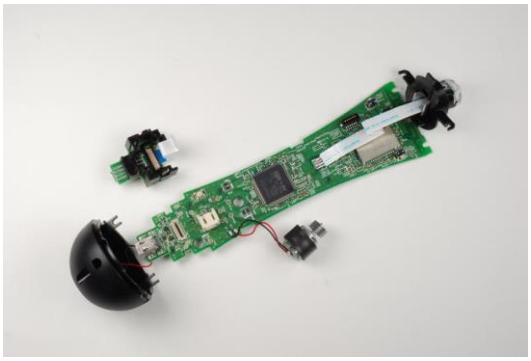


Figura 2.21 PCB PSMove

2.3 PSMove

Como este proyecto se dedica al estudio de este controlador vamos a explicar con detalle cada una de sus partes más importante y sus características principales.

La unidad está hecha de plástico y se mantiene unida con cuatro tornillos (Figura 2.22). La parte superior de la unidad tiene una esfera de goma que es fácilmente compresible. La parte inferior tiene varios puertos IO. La unidad, en sí misma, tiene una forma que facilita su sujeción. Debido a que estas unidades están diseñadas para moverse y girarse, es bastante resistente [14].

Figura 2.22 Mando PSMove

El controlador tiene varios botones, incluidos los elementos básicos de PlayStation: X, círculo, triángulo y cuadrado. La parte inferior de la unidad contiene un disparador (Figura 2.23) y los lados contienen los botones de inicio / selección.
El controlador también tiene dos botones adicionales: uno se encuentra en el centro de los cuatro botones principales en la parte superior y el otro es un botón más pequeño con el logotipo de PlayStation a continuación.

Figura 2.23 Trigger

Figura 2.24 Modelo

Figura 2.25 Puertos del Controlador

Si vemos el interior del controlador (Figura 2.26), solo se mantiene unido con cuatro tornillos ubicados en la parte posterior del controlador en las esquinas. En el interior, se revelan todas las partes principales del controlador, incluido un motor de vibración, una batería de iones de litio, circuitos, módulos y cables planos de conexión entre las partes. Curiosamente, la esfera está montada en la mitad de la carcasa, mientras que el resto de los componentes están montados en el otro lado y los dos están conectados eléctricamente a través de un cable plano.

Figura 2.26 Controlador Abierto

La batería que se encuentra en la unidad es una batería de iones de litio con una capacidad de 1380mAh a 3.7V (Figura 2.27). La batería, en sí misma, está increíblemente bien construida con acolchado de goma en el exterior para evitar vibraciones. Teniendo en cuenta lo pequeños que se han convertido los componentes electrónicos, la decisión de diseño de usar una batería grande de alta capacidad tiene sentido (ya que hay una gran cantidad de espacio vacío en el controlador). Además de eso, un controlador puede ser más fácil de usar si tiene algo de peso detrás y, por lo tanto, la batería también actúa como lastre para equilibrar el mando.

Figura 2.27 Batería de Litio

También revela el motor de vibración. Así la acción de un jugador hace que el jugador sienta una retroalimentación palpable del controlador. Un ejemplo clásico es disparar un arma en un juego: la mayoría de los juegos de disparos modernos activan los motores de vibración en los controladores cuando el jugador dispara su arma. Esto le da un efecto de realismo y ayuda a los jugadores a saber cuándo dispara su arma. Los motores de vibración también se pueden utilizar para situaciones como explosiones e incluso latidos cardíacos.

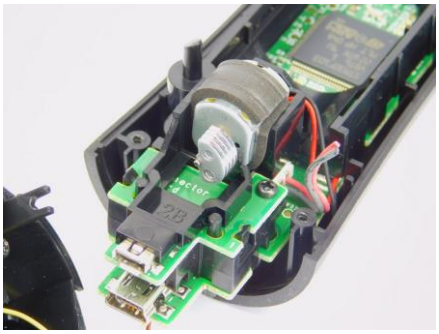


Figura 2.28 Motor de vibración e I/O USB puertos

La parte superior de la PCB principal es completamente visible sin la necesidad de quitar piezas o carcasas adicionales. La primera característica que destaca es el gran microcontrolador que aparece al quitar la batería. Este IC es un microcontrolador STM32F103 estándar que contiene una CPU basada en ARM, flash de 128 KB, USB, CAN, temporizadores, ADC y muchos periféricos. El paquete que se muestra en la PCB es un LQFP de 100 pines y generalmente se vende por 5€ - 9€ cada uno.

Figura 2.29 Microcontrolador STM32

Más abajo en la PCB (cerca de la esfera) se encuentra el conector de cable plano, un pequeño circuito integrado con la identificación AKM8974 y un módulo con un escudo de metal con la identificación ALPS 701A12B (Figura 2.30). El protector de metal se quita fácilmente con un destornillador pequeño de cabeza plana que revela algunas partes, incluida una EEPROM serie (24C32) y un Bluetooth Silicon BC4REA16(Figura 2.31). La EEPROM en serie probablemente almacenará información como nombres de red y contraseñas. Una razón para usar módulos cerrados como se ve aquí es para ayudar con el control EMC, ya que la carcasa metálica ayuda a absorber las señales EM emitidas para los circuitos, como el reloj y las líneas de datos. El AKM8974 es una brújula absoluta de eje magnético de 3 ejes que utiliza I2C para comunicarse con el controlador principal.

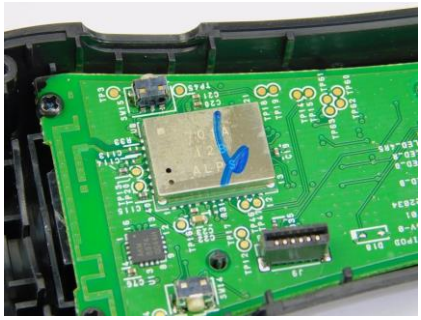


Figura 2.30 Escudo Metal

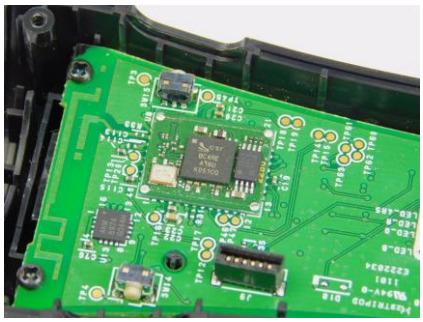


Figura 2.31 Módulo Bluetooth

La parte posterior de la PCB (cerca de la esfera) contiene los interruptores y pads de contacto para los botones.
Más abajo en la PCB, se pueden encontrar varios circuitos integrados, así como un módulo que contiene un escudo de metal. El primer circuito integrado, el KXSC4, es un acelerómetro de 3 ejes que es útil para la detección de gestos (Figura 2.31).
Como se verá en el apartado 2.5, los acelerómetros no se pueden usar para la posición absoluta o incluso la velocidad actual, ya que solo detectan un cambio en la velocidad.

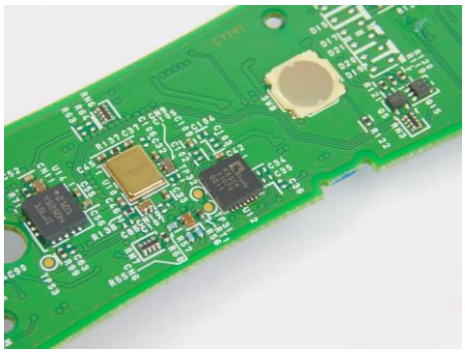


Figura 2.31 Acelerómetro de 3 ejes KXSC4

Un módulo realmente interesante se encuentra cerca, que está encerrado en un escudo de metal que parece estar chapado en oro (Figura 2.32).

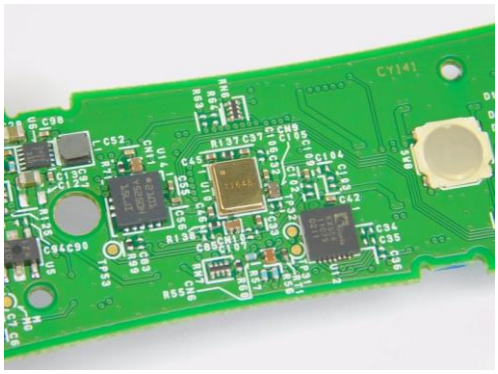


Figura 2.32 Escudo de Metal que cubre el Giroscopio

El segundo IC debajo del módulo es el Y5250H, que es un giroscopio del eje Z (Figura 2.33). Si bien no se puede encontrar una hoja de datos para esta parte, con frecuencia se asocia con el controlador PlayStation Move, lo que sugiere que este número de parte podría ser específico para la línea de producción. Algunos recursos en línea identifican esta parte como el STM LY5250 (un acelerómetro).

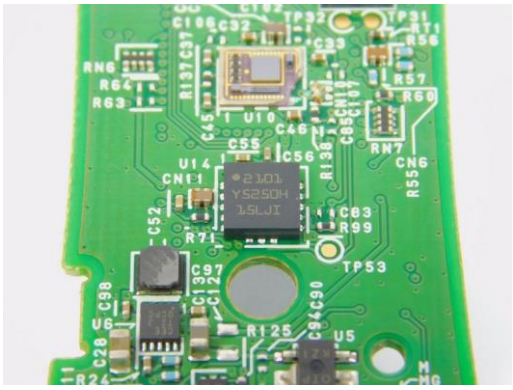


Figura 2.33 Giroscopio de un eje

La esfera (Figura 2.34) que se encuentra en la parte superior del controlador es principalmente espacio que contiene un solo LED multicolor. Curiosamente, se puede ver una gruesa pared de metal que proporciona el espacio necesario para que el cable plano acceda al LED. La razón de este escudo de metal (Figura 2.36) puede deberse al control de EMC y he aquí por qué. Para mejorar la eficiencia y crear patrones de color interesantes, las líneas RGB cambiarán y cambiarán rápidamente los niveles de voltaje. Tal cambio rápido (incluso si la frecuencia es baja) crea emisiones EM que pueden violar fácilmente las especificaciones de la FCC y la CE. Dado que las líneas RGB también cruzan los módulos de RF y la antena, existe una buena posibilidad de que las líneas RGB capten interferencias y luego vuelvan a emitir las señales más adelante en la línea. Por lo tanto, el protector de metal alrededor del cable plano ayuda a absorber la radiación emitida por el cable plano y, por lo tanto, ayuda a cumplir con las especificaciones de la FCC y CE sobre emisiones EM.

Figura 2.34 Esfera

Figura 2.35 Esfera eliminada

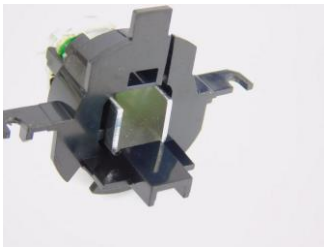


Figura 2.36 Escudo de Metal

2.4 PlayStation Eye

Como bien hemos comentado, para seguir nuestro controlador de movimiento en el espacio 3D, se usa la cámara PlayStation Eye (Figura 2.36). La cámara PS Eye es una muy buena opción como cámara, debido a su bajo costo, su conectividad USB 2.0 y la velocidad de fotogramas de 60 FPS a 640x480 y 120 FPS a 320x240. Además de una

buena velocidad de fotogramas, la cámara PS Eye también permite acceder a la duración de la exposición en los usos de la cámara. Desactivar la exposición automática y disminuir la duración de la exposición es importante para obtener buenos resultados de seguimiento y evitar problemas causados por el desenfoque de movimiento en otras cámaras. La cámara se lanzó en 2007, independientemente del controlador PSMove, pero la PS Eye se puede usar para otras aplicaciones sin Move Controlador de movimiento.

Figura 2.36 Cámara PlayStation Eye

Características:

- Obtener la posición XY y el radio de esfera de los controladores rastreados
- LED de actualización automática de controladores conectados
- Recuperar la imagen de la cámara como imagen RGB de 24 bits

Las funciones de fusión de sensores están disponibles cuando está disponible la orientación

- Obtener la matriz de proyección para renderizado 3D en la parte superior de la imagen de la cámara
- Obtener la matriz de vista de modelo para renderizado 3D con el centro de esfera del controlador como origen coordinado
- Obtener la posición 3D del controlador

2.5 Sensores Inerciales

Hasta aquí se han visto unos productos que permiten dotar de movimiento a un proyecto interactivo, determinando la posición, velocidad y orientación. Existen muchos sensores que pueden determinar esos elementos como brújulas magnéticas, GPS, barómetros, encoder ópticos y magnéticos o pulsadores de fin de carrera.
Pero entre este tipo de sensores destacan y nos centraremos en acelerómetros, giroscopios y magnetómetros.

2.5.1 Acelerómetro

Cómo su nombre indica, un acelerómetro es un dispositivo que permite la aceleración a la que está sometido [15].
Recordamos que la aceleración (ecuación 2.1) es la variación de la velocidad respecto del tiempo o, expresado matemáticamente,

$$\vec{a} = \frac{\partial \vec{v}}{\partial t} = \frac{\partial^2 \vec{r}}{\partial^2 t} \quad (2.1)$$

Asimismo, recordamos la primera ley de Newton (ecuación 2.2)

$$\vec{F} = m.\vec{a} \quad (2.2)$$

Es decir, que cualquier cuerpo con una masa *m* requiere una cierta fuerza para variar su velocidad, lo que es equivalente a que cualquier cuerpo sometido a una aceleración experimentará una cierta fuerza.
Los acelerómetros disponibles normalmente son de 3 ejes, es decir, son capaces de medir la aceleración a la que está sometido el sensor en las direcciones X, Y y Z independientemente, lo que permite saber simultáneamente la magnitud y dirección de la aceleración medida.
La capacidad de medir la aceleración de un sistema proporciona en sí misma funcionalidades interesantes, como registrar vibraciones o golpes.
Pero no es la única función que podemos obtener de un acelerómetro. El sensor se ve afectado por la gravedad terrestre, que supone una aceleración de aproximadamente 9.81 m/s^2 en la superficie de la tierra, que por supuesto es registrada constantemente por el sensor en el eje Z.

La medición de la gravedad puede emplearse para determinar la orientación del sensor. Dado que podemos registrar la aceleración en tres ejes, en ausencia de otras aceleraciones, podemos determinar la orientación del sensor con un poco de trigonometría.

En el caso de 2D, asumiendo que ponemos el sensor horizontal en su plano X-Y, con Z apuntando hacia arriba, y únicamente giramos en el eje Y, la ecuación 2.3 para el ángulo resulta (ver figura 2.37):

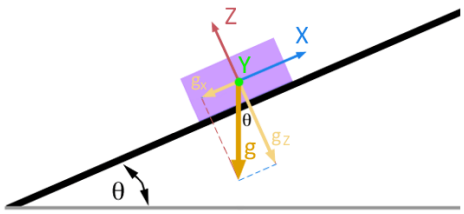


Figura 2.37

$$\theta = atan\frac{Ax}{Az} \quad (2.3)$$

Para el caso 3D las ecuaciones 2.4 (ver figura 2.38):

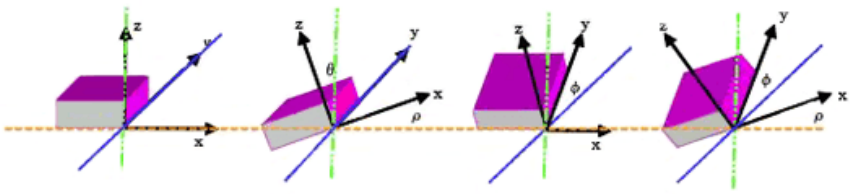


Figura 2.38

Con sus ecuaciones:

$$\theta x = atan\frac{Ax}{\sqrt{Ay^2 + Az^2}}$$

$$\theta y = atan\frac{Ay}{\sqrt{Ax^2 + Az^2}} \qquad (2.4)$$

$$\theta z = atan\frac{\sqrt{Ax^2 + Ay^2}}{\mathrm{Az}}$$

Como aspectos negativos, son dispositivos muy sensibles a las vibraciones, por lo que la medición presentará ruido de alta frecuencia. En general, deberemos filtrar la señal antes de poderla usar. En el caso más simple, simplemente hacer el promedio de varias mediciones será suficiente.

2.5.2 Giroscopio

Un giroscopio (también llamados giróscopo) es un dispositivo que permite medir el ángulo de rotación girado por un determinado mecanismo [16].

A diferencia de los acelerómetros, los giroscopios son dispositivos puramente diferenciales, es decir, no existe una referencia absoluta si no que siempre medimos ángulos relativos a una referencia arbitraria**.**

Al igual que en el caso de los acelerómetros, lo normal es que los giroscopios que empleemos sean de 3 ejes, es decir, que registran de forma independiente la rotación en X, Y, y Z**,** lo cual permite determinar la magnitud y dirección de la rotación (Figura 2.39).

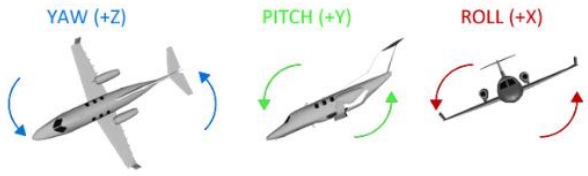


Figura 2.39 Rotación de los 3 ejes acelerómetro

Una de las consecuencias de emplear la fuerza Coriolis es que los giroscopios vibratorios, a diferencia de otros tipos de giroscopios**,** no registran el ángulo girado

sino la velocidad angular, que es la relación de variación del ángulo respecto del tiempo.

$$w = \frac{\partial \vec{\theta}}{\partial t}$$

Para obtener el ángulo de posición del sensor es necesario realizar la integración respecto del tiempo, algo que habitualmente hace la electrónica interna del sensor.

$$\theta gyro = wgyro * \triangle t$$

Efectivamente, el mayor problema que encontramos con los giroscopios de vibración es que a medio y largo plazo tienen deriva, es decir, que la medida se va desviando progresivamente del valor real (incluso con el sensor estático).

Por el contrario, los giroscopios son sensores de respuesta rápida y elevada precisión en tiempos cortos. Además, responden bien a cambios bruscos, y son relativamente inmunes al ruido, siempre en rangos temporales cortos.

Como podemos comprobar el acelerómetro y giroscopio tienen características de medición opuestas por lo que complementan muy bien entre sí.

2.5.3 Magnetómetro

Detecta el campo magnético terrestre de la misma forma que lo hace una brújula tradicional [17].
Con este sensor podemos leer las componentes del campo magnético presente, de esta forma conociendo la dirección del campo magnético terrestre podemos calcular la orientación con respecto al norte magnético de la tierra, esto siempre y cuando nuestro sensor no este expuesto a algún campo magnético externo u algún objeto metálico que altere el campo magnético terrestre.
El ángulo de una dirección contando en el sentido de las agujas del reloj a partir del norte magnético se denomina Theta en la Figura 2.40.

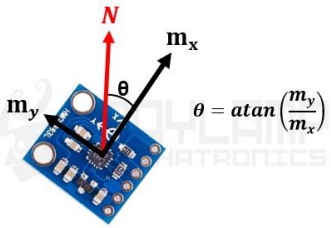


Figura 2.40 El ángulo theta determina la orientación del Norte Magnético.

Este ángulo nos determina la orientación del Norte Magnético, pero existe una diferencia entre el norte geográfico y el norte magnético, a esta diferencia se le conoce como declinación magnética, que en Zaragoza vale +0° 31’ (Figura 2.41)

Zaragoza Aragon
Latitude: 41° 39' 21.8" N
Longitude: 0° 52' 38.4" W
ARRABAL
Magnetic Declination: +0° 31'
Declination is POSITIVE (EAST)
Inclination: 56° 47'
Magnetic field strength: 45581.1 n

Figura 2.41 Declinación Magnética en Zaragoza

Estos sensores como ya hemos visto se implementan en el controlador PSMove. En general, el giroscopio se puede utilizar para la detección de rotación a corto plazo mientras el acelerómetro y el magnetómetro juntos se pueden usar para tener una orientación de referencia estable hacia el cual la rotación se puede reajustar a lo largo del tiempo.

Un caso de uso adicional para sensores inerciales es la estimación de posición cuando el seguimiento de la esfera y el algoritmo no puede determinar la posición 3D del controlador (por ejemplo, porque el controlador es oscurecido en la imagen de la cámara).

Con la información de las medidas proporcionadas por los tres tipos de sensores que componen una unidad inercial (acelerómetro, giroscopio y magnetómetro) descritos anteriormente, se puede representar la orientación de un objeto. Una de las formas principales de representar dicha orientación y la más intuitiva, son los Ángulos de Euler. Otra de las opciones para realizar la representación de un objeto, son los Cuaterniones, que son más complejos que los Ángulos de Euler, pero solucionan los problemas que se producen con estos últimos. Estos dos tipos de orientación se describen a continuación.

2.5.4 Ángulos de Euler

Los Ángulos de Euler constituyen un conjunto de tres coordenadas angulares que se utilizan para representar la orientación espacial de cualquier sistema de referencia de ejes ortogonales como una composición de tres rotaciones elementales a partir de una rotación estándar conocida [18].
Los ángulos de Euler (phi, theta, psi) corresponden con los ángulos convencionales de roll (ϕ), pitch (θ), yaw (ψ) que se utilizan en navegación para especificar la actitud de un móvil(Figura 2.42). El ángulo de roll (ϕ) es el giro de las alas del avión, el ángulo de pitch (θ) la inclinación del morro y el ángulo de yaw (ψ) es el giro del morro del avión respecto al norte.

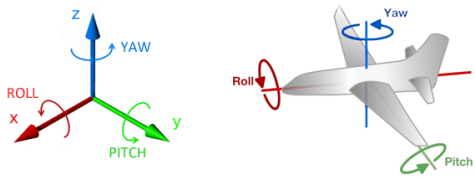


Figura 42: Representación de los ángulos de pitch, roll y yaw

Pero con los Ángulos de Euler se produce un problema llamado Gimbal Lock (Bloqueo de ejes), esto se produce cuando se pierde un grado de libertad. Es el caso cuando dos de los tres ejes necesarios para describir las rotaciones en tres dimensiones son llevados a la misma dirección (Figura 43)

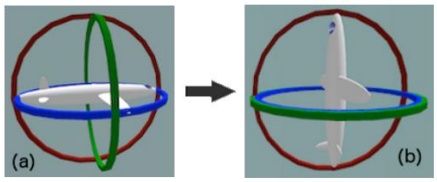


Figura 43: Gimbal Lock

2.5.5 Cuaterniones

Un cuaternión es un vector de cuatro elementos que se puede utilizar para codificar cualquier rotación en un sistema de coordenadas en tres dimensiones. Estos cuatro elementos son un elemento real y tres elementos complejos, y que pueden ser utilizados para mucho más que las rotaciones. Con los Cuaterniones se puede obtener la información necesaria para representar la posición de un sensor de orientación.
Se ha elegido la representación de cuaternión en PSMove, porque se usa como representación predeterminada en muchos paquetes de software que se ocupan de representaciones de orientación 3D, y porque evita el bloqueo de ejes de los ángulos de Euler.
El desarrollo técnico del concepto y manejo con cuaterniones excede de los objetivos de este trabajo, pero un par de páginas de consulta básicas se pueden encontrar en [19] y [20].

2.6 Elección del mando PSMove

Finalmente, el mando PSMove ha sido elegido para este trabajo porque es, probamente, el mejor controlador que consigue una alta precisión a la hora de jugar debido a los sensores ya comentados y sobre todo al sensor de campo magnético terrestre para localizar mejor los movimientos del jugador, además el mando no sólo detecta si mueves la mano arriba y abajo, adelante y atrás, sino también la distancia a la que se localiza el jugador de la pantalla.
Otro motivo para realizar su elección es debido a la conexión con el ordenador ya que el seguimiento de este controlador se hace con la cámara PlayStation Eye que lleva incorporado un cable USB y no es necesario buscar ningún otro tipo de adaptador.

3. Conexión y Calibración PSMove

Una vez vistas las principales partes y características del mando de control que se va a utilizar en este proyecto, se va a abordar cómo se realiza el conexionado y calibración del mando PSMove con un ordenador.

Para ello desempeñan un papel muy importante 2 librerías llamadas PsmoveApi y PsmoveService [21].

Estas librerías proporcionan métodos de software para configurar y controlar el mando, además de guardar toda la información que se quiera del controlador, como emparejarlo vía Bluetooth, calibración, posicionamiento e incluso comprobación de botones, luces…

Para el correcto funcionamiento es necesario instalar cualquiera de las 2 librerías ya que la funcionalidad es la misma, pero se diferencian en el modo visual a la hora de guardar la información y que nombra diferentes a las funciones y métodos del controlador.

Nosotros nos centraremos en PsmoveApi porque ha sido la librería que hemos estado trabajando, aunque en este Anexo I está la información detallada de las 2 librerías [22] y [23].

3.1 Características

Las características fundamentales de la librería PsmoveApi son:

- Emparejamiento del mando PSMove con el Ordenador.
- Ajuste del LED e intensidad del sensor de vibración.
- Lectura de los estados del botón y el estado del disparado analógico Trigger.
- Lectura de valores de los sensores (Acelerómetro, giroscopio, Magnetómetro).
- Leer los datos de calibración suministrados de fábrica a través de USB durante el emparejamiento.
- Leer información variada de los estados del dispositivo (temperatura, nivel de carga de la batería…)

3.2 Funcionamiento

PsmoveApi tienes dos ejecutables: *psmove* y *test_psmove*.

3.2.1. PSMove

El primero de ellos recoge toda la información de las características que se acaban de comentar.

El segundo corresponde a la detección del posicionamiento, que en este caso no funciona si no se tiene la cámara conectada. Aquí es importante destacar que al utilizas la cámara PS3EYE será necesario instalar un driver específico (Ver Anexo I.IV).

Una vez realizado toda la lectura y calibración del controlador, esa información se guarda en 3 ficheros dentro de PsmoveApi llamados: Psmove.h, PsmoveTracker.h, Psmove_Fusion.h

Dentro de esos ficheros se guardan funciones y métodos específicos que se han configurado previamente con los dos ejecutables mencionados anteriormente.

Vamos hablar de algunas de las principales funciones de cada fichero:

- Psmove.h guarda los valores internos del mando, es decir es posible elegir el tipo de conexión, leer los valores de acelerómetro, giroscopio, magnetómetro, modificar la intensidad de luz y ruido…
- PsmoveTracker.h son funciones especializadas solo para la cámara, es decir consigue la posición y radio del controlador.
- PsmoveFusion.h proporciona la definición de las funciones necesarias para obtener un posicionamiento en 3D.

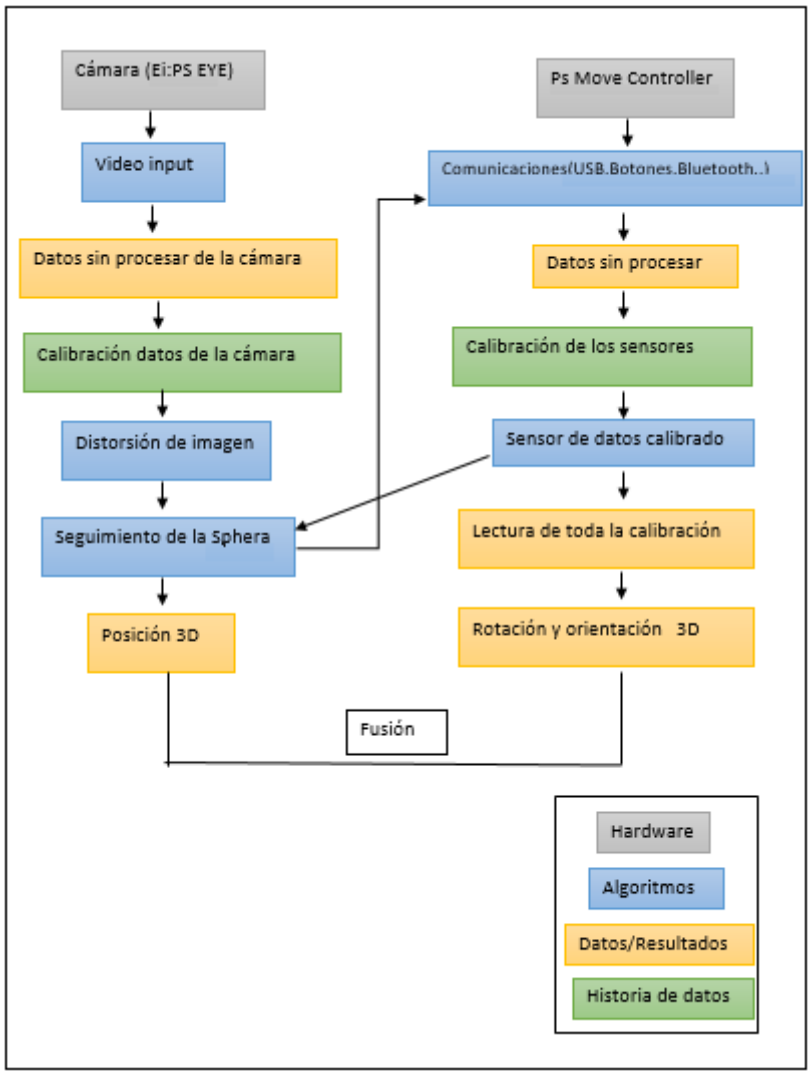
3.3.2 Test_psmove

Test_psmove se centra en el seguimiento de esfera basado en color, en lugar de depender de la detección de bordes y los bordes de la esfera, se tiene en cuenta el color para filtrar otros elementos en la imagen de la cámara. Además, se colocan puntos con diferentes colores en la esfera para permitir el seguimiento de la orientación 3D de la esfera. El enfoque de seguimiento de la esfera basado en el color tiene la ventaja de que la oclusión parcial se maneja bien, ya que al menos la mitad de la esfera aún es visible en la imagen de la cámara. Este enfoque no utiliza la transformación de 2D en absoluto, y se basa únicamente en el color, funciona con una sola cámara y en tiempo real.

Para seguir la orientación de la esfera, se colocan 32 puntos de colores (16 rojos y 16 verdes) en la esfera azul en un diseño especial que evita que dos puntos se fusionen en la imagen de la cámara al dejar suficiente espacio entre cada punto.
Para ubicar la esfera y los puntos, la imagen de la cámara se convierte del espacio de color RGB al espacio de color HSV, lo que permite el seguimiento basado en el tono de los colores que es estable incluso bajo diferentes condiciones de iluminación.
En general, este enfoque proporciona una buena manera de detectar esferas de colores usando el espacio de color HSV sin depender de transformaciones de trama 2D. El seguimiento de orientación usando puntos de colores también funciona bien, pero en el caso del PSMove se dispone de sensores inerciales que es posible usarlos para el seguimiento de la orientación, por lo que solo la primera parte (detección de posición 3D) es relevante para este proyecto.

En esta siguiente tabla explicamos en modo esquemático los procesos que tienen que seguir cada uno de los controladores. (Cámara y PSMove) para el funcionamiento correcto.

Tabla 2: Funcionamiento cámara y PSMove



4. Aplicaciones desarrolladas

Para comprobar el correcto funcionamiento del mando y sus posibilidades reales se han desarrollado un conjunto de pequeños videojuegos, que se han implementado con Unity. Unity es un motor de desarrollo o motor de juegos**.** El término motor de videojuego hace referencia a un software que posee una serie de rutinas de programación que permiten el diseño, la creación y el funcionamiento de un entorno interactivo; es decir, de un videojuego.

Los proyectos en Unity se basan en la utilización de Assets (paquetes de recursos), los cuales contienen materiales, efectos, scripts, sprites, etc... Unity dispone de la Asset Store, una tienda donde pueden adquirirse recursos ya creados por otros desarrolladores, gratuitos o de pago, que permiten reducir el tiempo de desarrollo. Para los iniciados en este motor gráfico, como en este caso, Unity facilita ayuda mediante una extensa comunidad en la que puede encontrarse prácticamente resuelto cualquier tipo de problema, una gran ayuda a la hora de empezar a utilizar un software desconocido y tan potente.

La Interfaz de Unity es intuitiva y sencilla, no ha sido difícil familiarizarse con ella en un periodo corto de tiempo. Esta se divide en varias secciones (Interfaz de Unity):

1. Project: en esta sección se muestra la estructura del proyecto en Unity, en ella se muestran los Assets que incluyen los recursos a usar en las escenas del videojuego.
2. Hierarchy: objetos de la escena actual.
3. Inspector: una de las partes más importantes de Unity aquí se muestran las características del objeto seleccionado, aquí se añaden los atributos y scripts que dan vida a los objetos.
4. Scene: pantallas que el usuario visualizará, Unity tiene las Scenes, donde se pueden añadir los recursos para construir los niveles o pantallas del videojuego.
5. Game: sección que muestra cómo se verá el juego al compilarse, hay que tener cuidado con el AspectRatio o resolución de la pantalla elegida, dependiendo de la plataforma sobre la que se desarrolle el proyecto, se mostrarán unas u otras.
6. Console: muestra los mensajes de error/debug.
7. Game controller:
   Play: ejecuta/detiene el juego.
   Pause: pausa el juego, permitiendo observar la escena sin necesidad de detener el juego por completo.

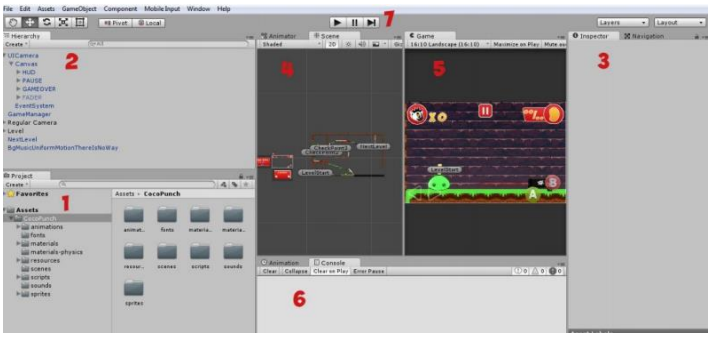


Figura 4 Interfaz de Unity. 1 Project, 2 Hierarchy, 3 inspector
4 Scene, 5 Game, 6 Console,7 Game Controller.

Antes de hablar de las aplicaciones se van a describir las principales características del código de nuestro controlador.
En primer lugar, el código está dividido en 2 scripts, UniMoveController.cs y UniMoveTest.cs.
El primero de ellos (UniMoveController.cs) llama a las funciones de PsmoveApi que son aquellas donde se guardan los datos que hemos configurado de nuestro controlador (emparejamiento, luz, orientación, posición, numero de controladores…). Es importante que la librería haya guardado toda la información, de lo contrario los movimientos serán erróneos o no funcionaran.
El segundo script (UniMoveTest.cs) corresponde al comportamiento del juego desarrollado, es decir, por ejemplo, dar acciones a un jugador
En este trabajo se han desarrollado tres pequeñas aplicaciones dónde se puede ver el alcance y comportamiento que tiene el mando PSMove [24].

4.1 SpaceShip

El primer prototipo está pensado en el comportamiento de la orientación, por tanto, esta demo no utiliza la cámara PS Eye por lo que no necesitaremos posicionamiento.
La demo consta de una nave espacial que puede rotar en 360° atacada por una serie de meteoritos contra los que hay que disparar con el objetivo de tener la mayor puntuación hasta que un meteorito choca con la nave.
Para hacerlo un poco más cercano a un juego real, se he programado un menú (Figura 4.1) de 3 niveles donde cada nivel incrementa la dificultad apareciendo más meteoritos y con mayor velocidad.

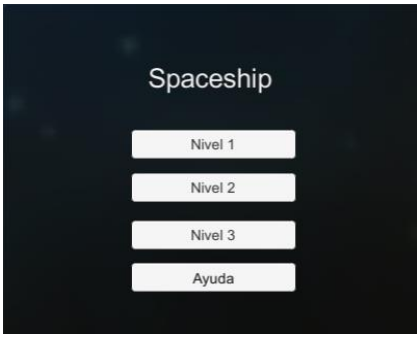


Figura 4.1 menú

Todos los elementos gráficos que aparecen en la demo están descargados de la tienda Assets de Unity. En esta aplicación se han programado varias partes.
El primer objetivo que se abordó fue que la nave siguiera los movimientos exactos del controlador. Para ello se creó un prefab[1] y un Instantiate[2] (que genera un clon y replica de aquello que se quiera).

```
GameObject moveController = GameObject.Instantiate(moveControllerPrefab,
                                Vector3.right * count * 0 +  Vector3.left
* i * 28, Quaternion.identity) as GameObject;
```

Se vio que era muy complicado que la nave acertara con el disparo al meteorito en un plano 3D, por lo que el planteamiento cambió a 2D y se hizo una pequeña modificación al vector de orientación, bloqueando uno de los ejes y por tanto la rotación de 360° quedó restringida a un solo eje.

```
moveObj.transform.localRotation=
Quaternion.Euler(move.Orientation.eulerAngles.y, -90, -90);
```

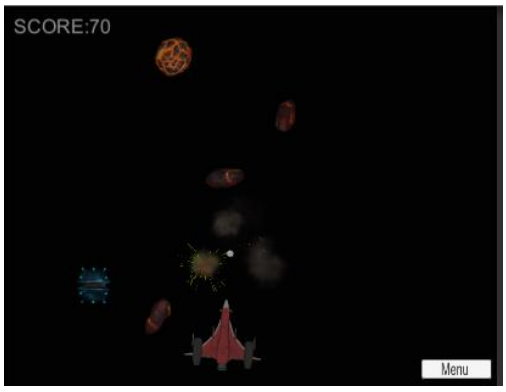


Figura 4.2(nivel 1)

Figura 4.3 (Nivel 2)

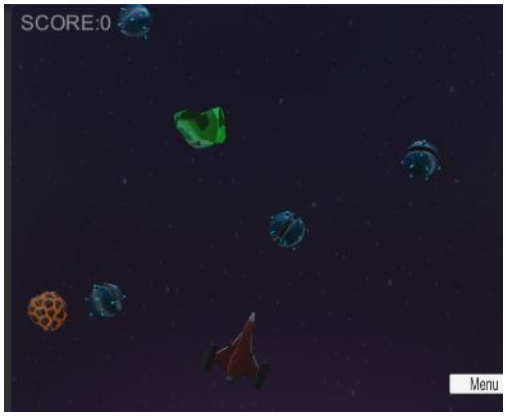


Figura 4.4(Nivel3)

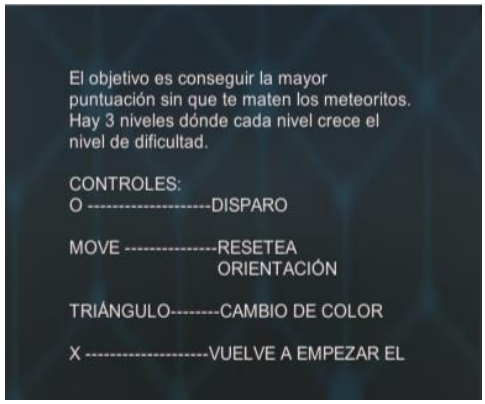


Figura 4.5(Ayuda)

Al inicio del juego, la orientación que aparece es una predefinida por defecto. Dependiendo de la posición del mando puede que no se detecte correctamente. Para

[1] **Objetos reutilizables**, y creados con una serie de características dentro de la vista proyecto, que serán instanciados en el videojuego cada vez que se estime oportuno y tantas veces como sea necesario
[2] Función de Unity donde crea una nueva copia de un objeto existente

resolver esta situación se incluyó la posibilidad de hacer un reseteo pulsando un botón del mando y realizar de nuevo la calibración de la orientación, este proceso dura muy poco y puede hacerse mientras se está jugando.
Con esta demo lo que se quería comprobar era el funcionamiento de rotación y orientación. Se ha comprobado que se hace de manera correcta y sin prácticamente errores. Si los videojuegos que se diseñen son de este estilo no habrá ningún problema a la hora de controlarlos con el mando.

4.2 Pinta y Colorea

Con esta pequeña aplicación es posible comprobar el rendimiento del posicionamiento con la cámara.
Pintar en la plataforma Unity es posible gracias a una opción llamada line Redender (Figura 4.5) donde a partir de una matriz de dos o más puntos en el espacio 3D se dibuja una línea recta entre cada uno. Para darle color a la línea basta con cambiar el material y el ancho de la línea, que es configurable (Figura 4.6).

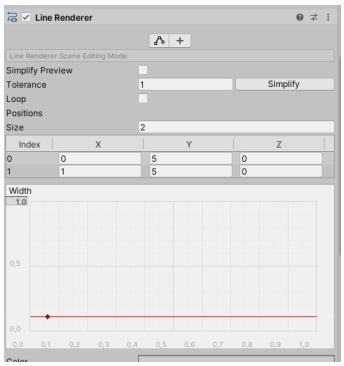


Figura 4.6 Line Redenderer

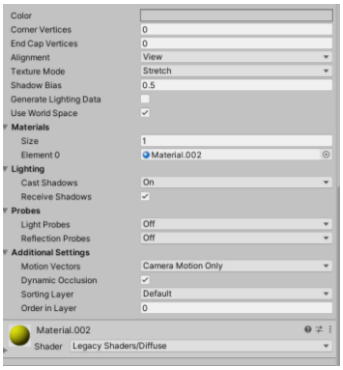


Figura 4.7 Configuración color y material

A diferencia del código anterior de la primera aplicación, en este caso es necesario realizar una función abrir la cámara, después una fusión cámara-controlador y, por último, habrá que procesar la información y actualizar los valores de seguimiento.

```
tracker = psmove_tracker_new();//Abrir cámara

fusion = psmove_fusion_new(tracker, 1.0f, 1000.0f);//Fusíon

psmove_tracker_update_image(tracker);//Actualizar   y   procesar   imagen
controlador

psmove_tracker_update(tracker, handle);//Actualizar y procesar
```

Para que el mando realice un seguimiento correcto lo primero de todo es que la variable *position* de PsmoveApi tenga guardada la información. Ese posicionamiento inicial se detecta gracias a la luz de la esfera del controlador.

Antes de poner en marcha la aplicación se recomienda siempre comprobar que tanto la cámara como el mando PSMove estén conectados. Para ello será necesario abrir el ejecutable test_tracker y realizar la comprobación de que todo funciona

correctamente porque si no es así, al arrancar la aplicación de pintar, el programa Unity se cerrará automáticamente o se bloqueará.

Si nos centramos en diseño de la aplicación empezamos con un pequeño menú (Figura 4.7) donde es posible elegir el color que nos da la opción. También desde Unity es posible cambiar el color y el grosor de la línea, además puede ser borrada la última línea que se crea.

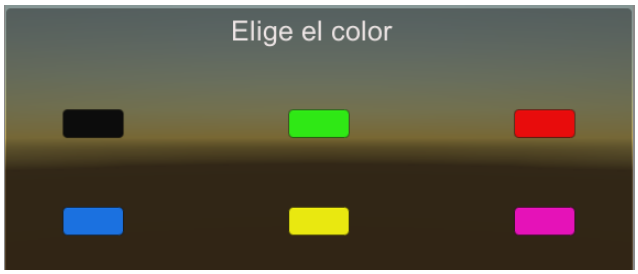


Figura 4.8 Menú

Después de elegir el color cambiará la escena (Figura 4.8) donde aparecerá un puntero en un fondo blanco y se puede empezar a dibujar de forma libre. Además, para hacerlo más divertido también está configurado para 2 jugadores (Figura 2.9).
Desde esa escena se puede cambiar de color o regresar al menú (Figura 4.10).

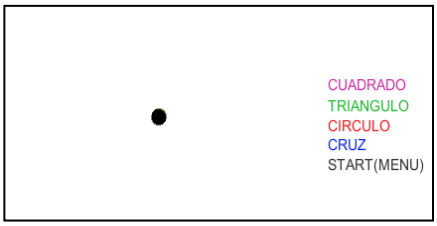


Figura 4.9 Escena dibujar

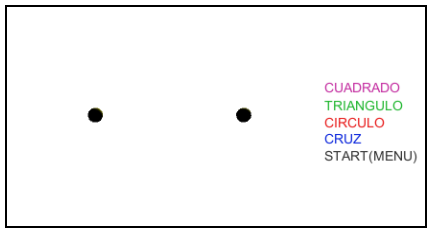


Figura 4.10 2 dibujantes

Figura 4.11 Dibujo 1

Figura 4.12 Dibujo 2

En esta demo se quería comprobar solo el posicionamiento y seguimiento con la cámara que, en este caso, también ha proporcionado resultados positivos.

4.3 Ninja Fruit

En esta última aplicación se ha querido juntar todas las acciones de las dos primeras demos, para comprobar tanto rotación como posicionamiento del mando.

El juego consiste en un cuchillo que se mueve, rota 360° y se mueve en todas las posiciones con el objetivo de intentar cortar el máximo número de frutas que van cayendo a lo largo de toda la escena de la aplicación.
Primero aparece un menú básico (Figura 4.11) con diferentes niveles en función del número de frutas.

Figura 4.13 Menú

Además, incluye una ayuda donde nos muestra los controles para cambiar de nivel (Figura 2.12).

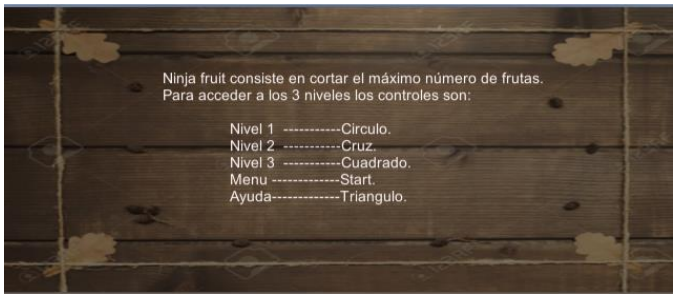


Figura 2.14 Controles

Figura 2.15 Corte Nivel 1

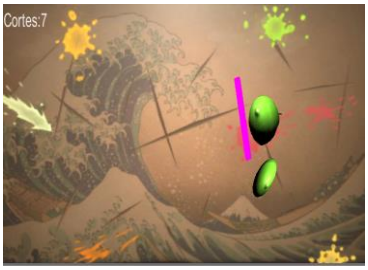


Figura 2.16 Corte Nivel 2

Figura 2.17 Corte Nivel 3

Se ha comprobado que el juego permite realizar los cortes de las frutas y, por tanto, utilizar todas las características de posicionamiento del mando. Con esta demo damos por finalizadas todas las acciones que se han estudiado y trabajado de nuestro controlador. Podemos decir que estamos bastante satisfechos ya que su funcionamiento era el esperado.

5 Conclusiones

Este trabajo se plantea desde el desarrollo de un trabajo de la asignatura de Sistemas Electrónicos de Audio y Video (optativa del 7º semestre del Grado de Ingeniería de Tecnologías y Servicios de Telecomunicación). En ese trabajo se comprobó que el mando PSMove no proporcionaba el resultado esperado a la hora de interactuar con juegos desarrollados en Unity.
En general, no se dispone de mucha información sobre este controlador para adaptarlo a un entorno Windows, pero se encontró alguna Tesis de referencia que permitió abordar los aspectos de configuración y manejo.
Como conclusiones podemos decir que:

- Se ha realizado con éxito la configuración y puesta a punto del mando PSMove para su manejo en entornos Windows.
- Se han identificado y puesto en marcha las librerías necesarias para hacerlo (PsmoveApi).
- Se ha caracterizado el manejo de los ejecutables asociados a esa librería (*psmove* y *test_psmove*).
- Se ha generado un manual de instalación y manejo (Anexo I, II) para que el proceso pueda replicarse de manera sencilla.
- Se han desarrollado tres aplicaciones interactivas en Unity que permiten validar la utilidad y capacidad del mando PSMove en entornos Windows.

Es importante remarcar que un problema que suele aparecer es que, dependiendo de la versión Windows de la que se disponga, puede fallar el emparejamiento por Bluetooth del mando al equipo.
En mi caso se ha utilizado la versión 4.0.7 de PsmoveApi con un ordenador Windows de versión 1909
De esta manera creo que se han cumplido los objetivos de este trabajo ya que se ha conseguido poner en marcha el control del mando PSMove en un entorno Windows.

5.1 Trabajo Futuro

Para ponerle punto final a este trabajo se plantean algunas ideas de mejora o que no se han implementado y que podría implementarse en un futuro proyecto.

PsmoveApi solo funciona con hardware estándar de Sony. El hardware personalizado podría darnos más control sobre el protocolo de comunicación y evitaría los problemas de emparejamiento en Windows. Una desventaja sería que el hardware debe estar diseñado y producido, y que grandes partes del software pueden necesitar ser reescritas.
Se podría estudiar cómo podemos ampliar el rango de visión de la cámara PlayStation Eye, ya que cuando te pasas de ese rango se pierde la conexión.

La elección del color solo puede ser configurada manualmente, estaría muy bien que la cámara pudiera analizar la imagen (al principio o incluso durante el seguimiento) y

seleccionar colores que no se ven en el marco de la cámara. Así la configuración del color se haría de manera dinámica y sin necesidad de ajustarla.

Por último, para hacer los juegos más dinámicos podemos pensar en la estructura de nuestro controlador la incorporación de un sistema de reconocimiento de voz.

Bibliografía

1. **Barton, Matt.** Atari 2600
Última consulta: mayo 2020
https://www.gamasutra.com/view/feature/3551/a_history_of_gaming_platforms_.php.

2. **Pastor, Javier.** Evolución Tecnológica
Última consulta: mayo 2020
https://www.xataka.com/historia-tecnologica/17-mandos-que-demuestran-lo-mucho-que-ha-cambiado-nuestra-forma-de-jugar-a-videojuegos-en-consolas

3. Super Nintendo
Última consulta: mayo 2020
https://es.wikipedia.org/wiki/Super_Nintendo.

4**.**Virtual Boy
Última consulta: mayo 2020
https://en.wikipedia.org/wiki/Virtual_Boy.

5. Nintendo 64
Última consulta: mayo 2020
https://es.wikipedia.org/wiki/Nintendo_64.

6. PlayStation3
Última consulta: mayo 2020
https://es.wikipedia.org/wiki/PlayStation_3.

7**.** PlayStation2
Última consulta: mayo 2020
https://es.wikipedia.org/wiki/PlayStation_2#DualShock_2.

8**. MasterGeek.** PlayStation4
Última consulta: junio 2020
https://www.ngeeks.com/dualshock-4-especificaciones-tecnicas-y-nuevos-detalles/.

9**.** Nintendo.
Última consulta: junio 2020
https://www.nintendo.es/Atencion-al-cliente/Nintendo-DS/Informacion-sobre-el-producto/Especificaciones/Informacion-sobre-el-producto-619794.html.

10**. Fuentes, Sacha.** Wii
Última consulta: junio 2020
https://www.xataka.com/videojuegos/especial-controles-de-videojuegos-wiimote.

11**. Palma, Álvaro.** Kinect xbox one
Última consulta: julio 2020
https://generacionxbox.com/kinect-para-xbox-one/.

12**.** Sensores Kinect
Última consulta: julio 2020

https://www.elotrolado.net/wiki/Kinect#:~:text=Caracter%C3%ADsticas%20y%20Especificaciones%20T%C3%A9cnicas%20de%20Kinect,-Sensores%2FC%C3%A1maras&text=C%C3%A1mara%20VGA%20con%20resoluci%C3%B3n%20de,Micr%C3%B3fono%20multi%2Darreglo..

13. **Greenwald, Will.** Nintendo Wii Remote vs PlayStation Move: How do they work? Última consulta: julio 2020 https://www.tested.com/tech/gaming/925-nintendo-wii-remote-vs-playstation-move-how-do-they-work/.

14. **Mitchell, Robin.** PSMove Última consulta: julio 2020 https://www.allaboutcircuits.com/news/teardown-tuesday-playstation-move-controller/.

15. **Llamas, Luis.** Acelerómetro Última consulta: julio 2020 https://www.luisllamas.es/como-usar-un-acelerometro-arduino/.

16. **Lamas,Luis Giroscopio** Última consulta: julio 2020 https://www.luisllamas.es/como-usar-un-giroscopio-arduino/.

17.Magnetrómetro Última consulta: julio 2020 https://naylampmechatronics.com/blog/49_tutorial-magnetometro-hmc5883l.html.

18. **Llamas, Luis.** Euler Última consulta: julio 2020 https://www.luisllamas.es/medir-la-inclinacion-imu-arduino-filtro-complementario/.

19. Cuaterniones Ultima consulta: septiembre 2020 https://es.wikipedia.org/wiki/Cuaternión

20. Cuaterniones Ultima consulta: septiembre 2020 https://es.wikipedia.org/wiki/Cuaterniones_y_rotación_en_el_espacio

21. **Cboulay.** PsmoveService Última consulta: julio 2020 https://github.com/psmoveservice/PSMoveService.

22. **Perl, Thomas.** *"Cross-Platform Tracking of a 6DoF Motion Controller. Using Computer Vision and Sensor Fusion."* Vienna University of Technology, Faculty of Informatics, 2012.

23.Versiones PsmoveApi Última consulta: septiembre 2020 https://github.com/thp/psmoveapi/releases

24. **Perl, Thomas.** Documentación y librerias
Última consulta: septiembre 2020
http://thp.io/2010/psmove/