ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Grado en Ingenieria Informatica

Sistema de captura de movimiento con Intel Edison

Motion capture system with Intel Edison

Realizado por
Alan Nicolás Martellotti
Tutorizado por
Juan José Ortega Daza
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, junio de 2018

Fecha defensa:
El Secretario del Tribunal

Resumen

Los objetivos de este trabajo de fin de grado se basan en diseñar y simular un sistema de captación de movimientos a través de diversos sensores interconectados con una placa de procesamiento Intel Edison [1], y el tratamiento de estos datos en un entorno de desarrollo externo, en este caso Unity [2], así como el estudio de la viabilidad y uso en diversos campos.

A lo largo del proyecto se profundizará en cómo calcular y diseñar un sistema que permita el cálculo de la posición en dos ejes, giro en tres ejes y flexión de las falanges medias y proximales de una mano, además de la interacción con programas de desarrollo externos los cuales mostrarán los movimientos leídos por los sensores a través de modelos 3D o avatares.

Para el montaje se hará uso del sensor MPU6050 [3], que permitirá calcular las diversas rotaciones en los tres ejes cardinales, su elección se justificará en los capítulos posteriores, al igual que la del sensor de flexión SPRK Flex [4] y de la cámara Kinect, utilizada para el cálculo de la posición en dos ejes.

La interconexión de los sensores anteriores se llevará a cabo a través del microcontrolador Intel Edison con extensión de placa para Arduino [5].

Posteriormente se utilizará la plataforma de desarrollo Unity, que se encargará de recoger los dato obtenidos por los sensores y crear simulaciones con modelos 3D o avatares con las que el usuario pueda interactuar.

Por último, se llevará a cabo un estudio de los ámbitos a los que se pueden aplicar este sistema de captación de movimiento.

Palabras Clave

Intel Edison, Arduino, Acelerómetro, Giroscopio, MPU6050, Posición, Kinect, Flexión, Unity.

Summary

The objectives of this End-of-Degree Project are based on the designing and simulation of a system capable of detecting and capturing movements through various interconnected sensors with Intel Edison's [1] developing platform,

and the treatment of the received data in an IDE, in this case, Unity [2], as well as the study of its viability and usage in diverse fields.

Throughout the project, I will go deep into calculations and design of a system that is able to calculate the position in two axis, rotation in three axis and flexion of the middle and proximal phalanges, in addition to the

interaction with external programs, which will display the movements read by the sensors through 3D models or avatars.

For the assembly of the system, we will make use of the sensor MPU6050 [3], which will let us calculate the rotations in the three-cardinal axis. Its election will be justified in the posterior chapters, so will be for the flexion sensor 'SPRK Flex' [4] and for the Kinect camera, used for the calculations of the position in two axes.

The interconnection of the mentioned sensors will be done through the microcontroller Intel Edison, with board extension for Arduino [5].

Posterior to it, we will make use of the developing platform Unity, which will be retrieving all the data from the sensors and create simulations with 3D models or avatars which the user can interact with.

Lastly, a study of the field in which this system can be applied too will be conducted

Key words

Intel Edison, Arduino, Accelerometer, Gyroscope, MPU6050, Position, Kinect, Bending, Unity

Tabla de contenido

Capítulo 1. Introducción

El trabajo de fin de grado denominado “Sistema de captura de movimiento con Intel Edison” se ha desarrollado para la titulación de Graduado en Ingeniería informática para la Escuela de Ingenierías informáticas de la Universidad de Málaga.

El director del presente proyecto ha sido el profesor Juan José Ortega Daza, profesor del departamento de Lenguajes y Ciencias de la Computación de la escuela de Ingeniería Informática de la Universidad de Málaga.

En este capítulo se presenta una descripción general del proyecto realizado, las motivaciones que llevaron a la creación del mismo, y una enumeración de las tareas realizadas en profundidad.

1.1 Contenidos de la memoria

El actual documento está separado en siete capítulos, estos se enumeran a continuación:

Capítulo 1: Introducción

Se describen los aspectos principales del proyecto, a través de una breve descripción de los sensores y programas a utilizar, además se introduce la motivación del planteamiento de este trabajo y una breve descripción del contenido de este documento.

Capítulo 2: Estudio del arte

Se realiza un análisis sobre los distintos métodos existentes para la determinación de la posición, orientación y flexión de los sistemas de captación actuales.

Capítulo 3: Tecnologías a usar

Se estudia qué tecnologías son aplicadas a lo largo del proyecto, valorando sistemas similares y explicando la elección de los sistemas utilizados en este trabajo.

Capítulo 4: Determinación de ángulos de giro, posición y flexión

En este capítulo se estudia el cálculo para la obtención de los ángulos de giro, la posición de la mano y la flexión de la misma para poder utilizar los datos obtenidos en un sistema externo, además de la evaluación de los datos generados por los cálculos anteriores.

Capítulo 5: Montaje y pruebas

Se procederá al montaje de pruebas individuales y completas, en ellas se mostrarán, a través de una interfaz con avatares 3D, los diversos movimientos capturados a través de los sensores y la interacción con el usuario final.

Capítulo 6: Conclusiones y posibles líneas futuras

Se realizará una reflexión de los resultados obtenidos, comprobando la exactitud y precisión a la hora de representar los movimientos del mundo real en un programa.

También se enumeran las principales conclusiones extraídas, los problemas encontrados y las posibles líneas de investigación o líneas de actuación a seguir en proyectos futuros similares.

Bibliografía

Enumeración de proyectos, webs, documentación y manuales que se han utilizado a lo largo del trabajo para documentarse acerca del mismo, así como obtener los datos e información necesaria para su realización.

Anexos

Todos los anexos se encuentran en el CD adjunto a este trabajo. Contienen las demos utilizadas en la memoria, los códigos fuente realizados para la medición de los datos y para la virtualización sobre un sistema externo de desarrollo, así como videos explicativos de las demostraciones técnicas creadas.

1.2 Descripción general

El trabajo actual plantea la creación de un sistema de captación de movimientos de bajo coste el cual sea capaz de simular una mano, para la posterior emulación de los datos adquiridos en distintos programas externos y aplicaciones.

Además, se estudiará la posibilidad de determinar la posición y orientación de una mano en un entorno de tres dimensiones mediante sensores interconectados a través de la placa Intel Edison [1]. Para tal propósito se ha utilizado el sensor MPU6050 6-axis [3], el cual se compone de un acelerómetro y giroscopio de tres ejes cada uno, un sensor de flexión SPRK-Flex [4] que dispone de un grado de flexión de 0 a 180 grados aproximadamente, un sensor Kinect [6] para el cálculo de la posición en los ejes X e Y, y un microcontrolador Intel Edison con extensión para placas Arduino [5].

La elección de la plataforma de desarrollo Unity [2] como sistema de virtualización de los datos se debe a su rapidez a la hora de poder desarrollar, el amplio número de librerías y documentación en red.

Se estudiará a lo largo de esta memoria las soluciones anteriores, actuales y en desarrollo, los sensores escogidos y sus variantes, el cálculo de los movimientos individualmente, junto con su precisión y viabilidad, el montaje con la plataforma Unity y los numerosos ámbitos a los que se puede aplicar dicha tecnología.

Finalmente se procederá a la creación de un modelo físico de un guante, este dispondrá de un flexómetro colocado entre la falange proximal y media del dedo corazón, el sensor MPU6050 [3] acoplado en la parte superior de la palma, el cual nos permitirá calcular la orientación y la cámara Kinect, a una distancia y altura de mínimo un metro.

1.3 Motivación

La motivación de este proyecto radica en la implementación de un sistema de bajo coste capaz de detectar los movimientos con un margen de error reducido y la posterior utilización de dichos datos en la creación de diversas aplicaciones para poder estudiar tales movimientos en ámbitos como la medicina, fisioterapia, gamificación [7] , aprendizaje lúdico, ocio, etc.

Como se ha podido observar, actualmente se utilizan sensores en amplitud de campos, todos estos campos a los que puede ser aplicable este sistema de detección de movimientos serán explicados en el capítulo 6 “Conclusiones y posibles líneas futuras”.

Actualmente muchos de los objetos cotidianos habituales, cuentan con sensores que añaden funciones complementarias y facilitan el uso, optimizan los recursos y convierten datos externos en operaciones internas fundamentales.

Los Smartphones, como ejemplo principal, disponen de algunos sensores, como son: acelerómetros, giroscopios, magnetómetros, GPS, sensores de proximidad, de luminosidad, de temperatura, etc. que permiten ubicar la posición, detectar sobrecalentamientos, guiar a través de mapas, ajustar el brillo automáticamente y demás funcionalidades similares.

Existen soluciones similares que permiten captar el movimiento producido por una mano, estos son algunos de los ejemplos que han existido, existen actualmente o se encuentran en desarrollo:

- **Nintendo Power Globe** [8]**:** Uno de los primeros sistemas capaces de capturar movimientos gestuales. Este guante posee dos altavoces ultrasónicos que actúan como transmisores y tres micrófonos ultrasónicos que actúan de receptores. A través de pulsos de 40 KHz, se triangulariza la posición de dichos sonidos para determinar los ejes X, Y, Z, lo que le permite calcular la dirección y el ángulo. Cuenta además con sensores de flexión que le permiten conocer el ángulo al flexionar la mano.

- **Nintendo Wii** [9]**:** El mando consta de sensores de aceleración, giroscopio y de infrarrojos, los cuales le permiten obtener la posición, giro y dónde está apuntando el usuario con el mando.

- **HTC Vive, Oculus Rift, SteamVr** [10]**:** Los nuevos sistemas de realidad virtual cuentan también con mandos que permiten obtener la posición, giro y cuentan con botones que permiten al usuario moverse a más distancia e interactuar con diversas aplicaciones y programas.

- **Acceleglove** [11]: Aún en desarrollo por la empresa americana AnthroTronix en colaboración con el Departamento de Defensa de Estados Unidos, este guante consta de seis acelerómetros, distribuidos en cada dedo y en la palma de la mano. Su finalidad está en el control de maquinaria a distancia y de traducción del lenguaje de signos.

- **Leap Motion** [12]**:** Este dispositivo ilumina la zona de cobertura mediante infrarrojos emitidos por tres leds frontales de sus dos cámaras, y estudiando la reflexión de la luz recogida, es capaz de detectar los movimientos, rotación y flexión de la mano en tres ejes. Actualmente es la solución más viable para detección de gestos y movimientos de la mano, aunque depende mucho de circunstancias externas y del espacio donde se utilice.

Dado que la cantidad de soluciones actuales e intentos pasados justifican el auge en este tipo de sistemas, el presente proyecto estudia el montaje y la viabilidad de un sistema de bajo coste y piezas habituales, capaz del reconocimiento de la posición en eje X y eje Y, el giro en tres ejes, y la flexión de los dedos, capaz de combinar las soluciones anteriores para diseñar un modelo económico que se pueda adaptar a múltiples ámbitos como la fisioterapia, medicina, videojuegos, etc.

Capítulo 2. Estado del arte de los sensores de movimiento

2.1 Introducción al capítulo

En este apartado se introduce qué es un sensor y los distintos tipos que existen en el mercado, así como el análisis de métodos actuales para la determinación de la posición, orientación y flexión.

Finalmente se realiza un repaso de las técnicas más usadas para la captación de movimientos.

2.2 Sensores de posición

Un sensor, transmisor, detector, transductor o emisor, es el elemento principal de cualquier sistema de medición. Se encarga de recoger datos del mundo real y generar una salida digital o analógica del mismo para poder evaluar tales datos desde cualquier aplicación, programa o sistema. Dichos dispositivos son muy importantes en diversos campos como la robótica, sistemas de navegación, animación, vehículos terrestres, etc.

2.3 Conceptos de medición de la posición

A la hora de seleccionar el sensor a utilizar, es importante tener en cuenta una serie de términos y conceptos, ya que una selección incorrecta puede terminar influyendo negativamente sobre las mediciones del proyecto. Se establecen las siguientes definiciones [13]:

Exactitud: Permite medir la veracidad de la salida.

Resolución: Medida más pequeña de incremento o decremento que se puede medir.

Precisión: Estabilidad de la medición al volver al mismo punto, también conocido como grado de repetibilidad.

Linealidad: Diferencia entre la salida del sensor y la posición real.

2.4 Tipos de sensores

La medición de la posición es la segunda propiedad más comúnmente utilizada, solo superada por la temperatura. Existen diversos tipos de sensores capaces de calcular la posición, pero dado que el estudio del actual proyecto se basa en los movimientos de una mano, se descartan aquellos sensores manuales y de gran volumen. Podemos clasificar los sensores de la siguiente forma:

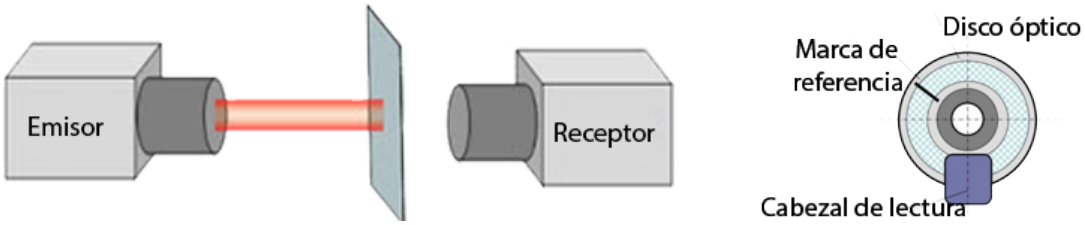
1. Sensores ópticos
2. Sensores magnéticos
3. Sensores capacitivos
4. Sensores inductivos tradicionales
5. Sensores de ultrasonidos
6. Sensores inerciales, o Unidad de Medición Inercial(IMU)

2.4.1 Sensores ópticos

Conocidos comúnmente como encoders, estos sensores suelen utilizarse normalmente para la medición de la posición. Cuentan con dos partes, un emisor y un receptor; un halo de luz brilla desde un disco de vidrio o plástico, y el fotorreceptor mide la señal de posición [14].

Existen dos posibles mecanismos para la recogida de datos externos. En el primer mecanismo, la luz se transmite de un extremo a otro y se monitoriza el cambio de algunas de sus características, ya sea la intensidad, la longitud de onda o similar.

En el segundo escenario los emisores disponen de una codificación en el disco a través de luz alterna o sectores oscuros, de manera que los pulsos son generados con el giro del disco, los receptores se encargan de recoger los pulsos y a través de la velocidad de giro del emisor, pueden calcular la posición angular. A través de dos receptores podemos calcular la dirección del movimiento.



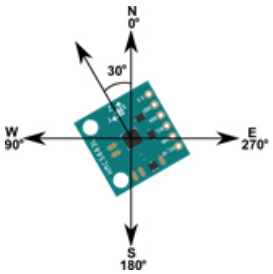
Ejemplo de sensores ópticos, imagen perteneciente a [15]

- **Puntos fuertes:** Alta resolución, buena precisión.
- **Puntos débiles:** Fallos sin advertencia previa, temperaturas extremas, suciedad.

2.4.2 Sensores magnéticos

Todos los sensores magnéticos utilizan un principio parecido, basado en el desplazamiento de un imán respecto a un detector magnético. Los más habituales para Arduino suelen contar con tres ejes, de manera que se pueden leer los componentes del campo magnético presente para calcular la orientación con respecto al norte magnético de la tierra [15].

Estos sensores no se suelen usar para aplicaciones de alta precisión debido a la histéresis magnética [16] , que es la tendencia de un material a conservar una de sus propiedades, y a su imposibilidad de estar cerca de fuentes de corriente continua o acero.



- **Puntos fuertes:** Robusto.
- **Puntos débiles:** Histéresis y no puede estar cerca de fuentes de acero o corriente continua.

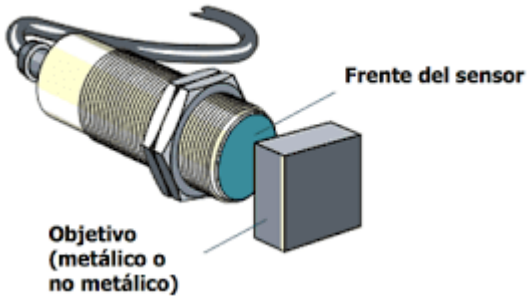
2.4.3 Sensores capacitivos

Estos tipos de sensores miden la separación entre dos placas conductoras separadas por un aislante, estas mediciones suelen ser inferiores a 1mm para la medición de la deformación, carga y presión.

Existen dos métodos para la medición:

- **Modificación de la constante dieléctrica.** El material a medir está conectado al material dieléctrico entre las placas, cuando este se mueve, esta constante varía, y a través de este cambio en la capacitancia, es posible obtener la posición de dicho material.
- **Modificación de la zona de solapamiento**. El material a medir se conecta a una de las placas, y a través de este movimiento la zona de solapamiento varía, obteniéndose la posición a través de la capacitancia.

Estos sensores suelen tener mala reputación dadas sus tolerancias de instalación estrictas y se suelen evitar a no ser que exista una necesidad de mediciones de alta precisión en aplicaciones muy estables y cíclicas [17].

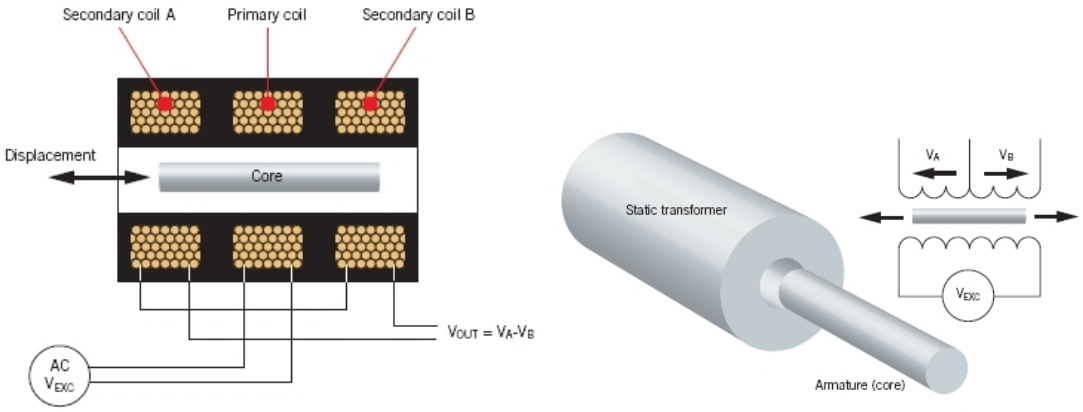


Sensor capacitivo, imagen perteneciente a [18]

- **Puntos fuertes:** Baja potencia necesaria.
- **Puntos débiles:** Propenso a fallos de temperatura y humedad, instalación compleja.

2.4.4 Sensores inductivos tradicionales o LVDT

También conocidos como Transformadores Diferenciales Linealmente Variables (LVDT) [19], estos transformadores utilizan una construcción de al menos tres objetos conductores, de forma que, al desplazarse el primario entre los secundarios, varía el acoplamiento electromagnético, y esta relación indica la posición del objeto primario con respecto a los devanados secundarios. Se suele utilizar para medir movimientos desde micras hasta centímetros, y dado que los sensores inductivos pueden separar la electrónica del área de detección, hacen que sea posible utilizarlos en entornos hostiles.



LVDT, imagen perteneciente a [19]

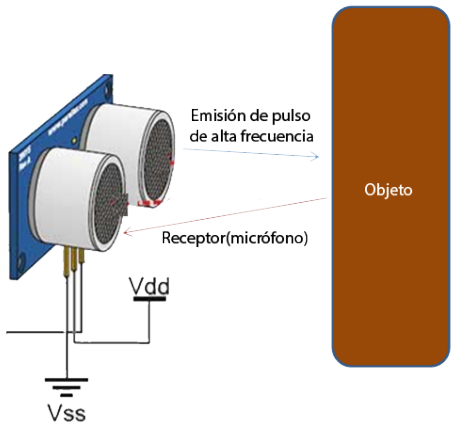
- **Puntos fuertes:** Alta precisión, funcionamiento en entornos hostiles.
- **Puntos débiles:** Masa y coste elevados.

2.4.5 Sensores de ultrasonidos

Estos sensores se basan en el envío de un pulso de alta frecuencia, el cual rebota y es reflejado hacia el sensor, que dispone de un receptor, micrófono en este caso, capaz de medir la distancia a través del tiempo entre pulsos, dado el conocimiento previo de la velocidad del sonido [20].

Los sensores de ultrasonidos son económicos y su precisión oscila entre 20 cm a 200 cm, con una resolución de 0,3 cm, aunque solo permiten calcular la distancia y

posición a través de los rebotes de los sonidos, por lo que son propensos a errores según el entorno y el ruido.



Sensor de ultrasonido, imagen perteneciente a [20]

- **Puntos fuertes:** Coste reducido, facilidad de uso.
- **Puntos débiles:** Sensible al entorno, solo detecta distancias a través de rebotes.

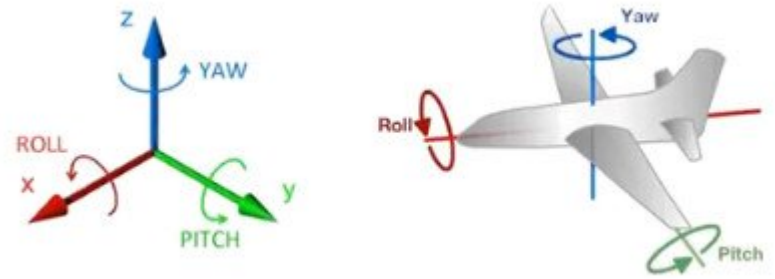
2.4.6 Sensores inerciales, o Unidad de Medición Inercial(IMU)

Compuestos por acelerómetros, giroscopios y magnetómetros**,** se enumeran las características más importantes de cada uno de estos componentes:

- **Acelerómetro**: Instrumento capaz de medir aceleraciones, dicha aceleración es la asociada con el fenómeno de cambio de peso experimentado por una masa que se encuentra en el marco de referencia del dispositivo. Este tipo de aceleración es diferente dependiendo del entorno, por ejemplo, el valor obtenido en posición de reposo es diferente del conseguido en movimiento, ya que las masas poseen un peso a pesar de la falta de velocidad que la que se puede medir en una caída gravitacional, ya que en este caso la masa tendrá un valor de 0. Esto quiere decir que, calcular la aceleración lineal requiere de ciertos cálculos y filtros, como se detalla en capítulos posteriores.

- **Giroscopio**: Dispositivo mecánico capaz de medir la orientación, utilizando los principios de la conservación del momento angular. Dichos dispositivos utilizan un MEMS [21] (MicroElectroMechanical Systems) para medir la velocidad angular usando el efecto Coriolis [22], que es la aceleración relativa que sufre dicho dispositivo dentro de un sistema de referencia no inercial con respecto al eje de giro. Una vez obtenida la velocidad angular es posible calcular el desplazamiento angular a través del tiempo que ha transcurrido en la medición de dicha velocidad, lo que proporciona la posición angular si se sabe dónde se inició el giro.

- **Magnetómetro**: Componente que mide la fuerza y/o dirección ejercida por los campos magnéticos de la tierra. Dado que son componentes que dependen del campo magnético, son propensos a fallar por fuentes externas. La utilidad de este radica en ubicar el norte terrestre y es un elemento opcional que a menudo poseen algunos sensores inerciales.



Distintos tipos de rotaciones y ejemplos visuales, imagen perteneciente a [23]

De todos los sensores anteriormente citados, se utilizarán los sensores inerciales o IMU [24], ya que permiten calcular los ángulos de giro y aceleraciones, además de ser los más económicos y habituales del mercado, por lo que se estudiarán más en profundidad en el capítulo 3.

2.5 Fallos comunes al escoger sensores

Algunos de los errores más comunes a la hora de elegir un sensor de posición son los siguientes:

No tener en cuenta el coste del fallo del sensor: Dado que en este caso particular se usan sensores de bajo coste, se debe evaluar si el error obtenido por tales sensores es más caro de solucionar que el disponer de un sensor menos económico, pero más exacto.

No elegir bien el tipo de sensor con respecto al entorno: Hay que tener en cuenta que cada técnica de medición tiene sus puntos fuertes y débiles, y hay que elegir cada sensor dependiendo de las necesidades. Por lo general, dependiendo del entorno no hay elegir:

- **En aplicaciones donde se necesita una gran precisión**, no se deben escoger sensores magnéticos, a no ser que se pueda erradicar los campos magnéticos y disponer de un montaje mecánico sobre dicho sensor.
- **En sistemas con vibraciones constantes**, no se utilizarán potenciómetros. Esta circunstancia se debe a que los contactos eléctricos deslizantes están sujetos al desgaste y fallo, causados por la cantidad de movimientos producidos por el sistema.
- **Para entornos sucios o húmedos**, no se debe escoger potenciómetros, sensores ópticos o capacitivos.
- **En aplicaciones con temperaturas de funcionamiento extremas,** no debemos utilizar sensores ópticos, magnéticos, o capacitivos.
- **No leer el datasheet o manual:** La industria de los sensores de posición se encuentra en un momento de auge, lo que provoca la competitividad entre las empresas fabricantes. Esto ha llevado a algunos fabricantes a actuar de forma más comercial en relación a los datos de la especificación. Un ejemplo puede ser la mención de la resolución de conteos por revolución, pero no de la precisión, o de la alta resolución de la que dispone dicho sensor, pero no de la repetibilidad, lo que puede llevar a tener un sensor de medición con un ruido más alto de lo esperado en su salida.

Capítulo 3. Tecnologías utilizadas para escenarios 3D

Para afrontar el montaje, se ha escogido Intel Edison como placa de desarrollo, los sensores MPU6050, cámara Kinect, Flexómetro SPRK-Flex y la plataforma Unity para el montaje. En los siguientes apartados se tratan los motivos de las elecciones de Intel Edison frente a otras alternativas, del IMU MPU6050, Flexómetro SPRK-Flex, Kinect y de Unity como plataforma de desarrollo.

3.1 Intel Edison, elección y comparación

Intel Edison es un módulo de computación fabricado por Intel como un sistema de desarrollo para dispositivos y para sensores, conocido comúnmente con el nombre de Internet de las Cosas, de ahora en adelante IoT [25] (Internet of Things).

Esta plataforma de desarrollo está diseñada para desarrollar prototipos rápidamente y producir aplicaciones que tengan integración con IoT y con cualquier sistema externo al que se pueda acceder mediante una conexión a internet y/o cable. Existen dos IDE (Integrated Development Environment) y lenguajes a elegir para esta plataforma [26].

- **JavaScript y Node.js:** La utilización del lenguaje JavaScript, tanto para el desarrollo de software como del servidor con Node.js, sirve para crear interfaces web y permitir que los sensores puedan comunicarse entre ellos de una manera sencilla, a la par de permitir una forma simple de conectividad en la nube. Su IDE es Intel XDK.

- **Arduino:** Dado que Intel Edison provee una placa de extensión para Arduino, es posible utilizar los conectores tanto de ambos. Basado en el lenguaje C++, su IDE recomendado es Arduino IDE.

Cabe destacar que dada su conexión WIFI y Bluetooth, es capaz de leer datos de sensores y transmitirlos en tiempo real a cualquier dispositivo externo o aplicación en la nube, así como recibir información desde cualquier dispositivo o aplicación externa y aplicar cambios a los sensores o motores conectado a la placa.

Una alternativa al uso de la placa Intel Edison, es el uso de dos sistemas: Raspberry PI 3 y Arduino. Si estos dos sistemas se combinan, permite recibir y enviar los datos de los sensores gracias a Arduino y utilizarlos en cualquier aplicación interna o externa a través de la conexión wifi o Bluetooth que proporciona Raspberry PI 3. Esta última cuenta con una gran variedad de opciones de conectividad y compatibilidad para desarrollar en cualquier plataforma, además de salidas de video y audio que permiten desarrollar sin necesidad de un computador externo.

A continuación, se presentan las principales características y especificaciones técnicas de las placas anteriormente descritas.

Placa	Intel Edison con Arduino Board [1]	Raspberry 3 y Arduino [27]
Sistema operativo	Yocto Linux/Arduino Linux	ARM7 compatible Linux / Win 10 (Pocas aplicaciones compatibles) / Android.
SoC(System On a Chip)	22 nm Intel SoC que incluye un procesador dual-core Intel Atom de 500 Mhz y un microcontrolador Intel Quark de 32-bit de 100 MHz	1.2GHz 64-bit ARM Cortex A53 Quad Core Processor
Ram(Random Access Memory)	1 GB LPDDR3 POP, 2 canales de 32 bits a 800 MT/sec (MegaBytes per second)	1GB LPDDR3 SDRAM
Almacenamiento Flash	4 GB eMMC (embedded MultiMediaCard)	Externo, tarjeta SD o dispositivo USB 2.0

Bluetooth	Bluetooth 4.0	Bluetooth 4.1
WiFi	Broadcom 43340 802.11 a/b/g/n, antena integrada en la placa.	10/100 Ethernet(RJ-45) vía hub USB, Wifi 802.11n,
Placa E/S	Compatible con Arduino Uno, con 4 PWM (Pulse Width Modulation) pines en vez de 6.	2 PWM, uno compartido con el audio.
Pines digitales	20 pines de entrada/salidas digitales, incluyendo 4 pines como salidas PWM.	14, incluyendo 6 como salidas PWM.
Entradas analogicas	Hasta 6 entradas de 10-12 bits programables. A0 – A5 pueden ser usadas como entradas digitales y operar a 5v o 3,3v.	6 entradas analógicas de 10 bits programables.
Transmisor-Receptor Asíncrono Universal	1 UART (dispositivo que controla los puertos y dispositivos serie).	GPIO 14 y 15 pueden asignarse como UART.
Inter-Integrated Circuit	1 I^2C (comunicación interna entre diferentes partes de un circuito).	1 I^2C
In-Circuit Serial Programming	1 ICSP de 6 pines(SPI) (nos permite programar dispositivos lógicos programables, microcontroladores y otros circuitos desde nuestro ordenador usando la placa Intel Edison).	Desconocido. Puede soportar OpenOCD para Cortex-A7.

Cabe destacar que Intel Edison funciona en dos modos, lo que convierte dicha placa en una elección polivalente para cualquier tipo de proyecto, sus dos modos son:

- **Host:** Permite que la placa actúe como un ordenador, cualquier periférico externo como un teclado, webcam o ratón puede ser conectado. Gracias a esto es posible transmitir contenido a través de programas ftp, como Putty o similar, y usar la placa como servidor o host remoto.

- **Dispositivo:** La placa actúa como un computador periférico hacia el ordenador principal. Suministrando una conexión de 5V, permitirá leer y escribir en la memoria flash y programar a través de Arduino IDE o de Intel XDK IDE a través de USB o Wifi.

Puesto que el objetivo del proyecto se basa en la lectura de datos y aplicación de éstos en un sistema de emulación, desarrollado en la plataforma Unity, se necesita una lectura simple y rápida de los datos.

Como se ha estudiado, ambas soluciones son igual de válidas, pero dado la reciente incorporación de Intel Edison al mercado en comparación con Raspberry, y que la primera placa dispone de las conexiones de los sensores y la posibilidad de transmitir datos externamente en un solo dispositivo (considerando que Raspberry debe estar conectado también a una placa Arduino), elegiremos la placa Intel Edison Board para Arduino, con Arduino IDE como entorno de desarrollo.

A la hora de escoger el lenguaje de desarrollo, se ha optado por C++, dada la ágil respuesta que dispone al recoger datos en tiempo real y transmitirlos a través del puerto COM, lo que evita problemas de latencia. Asimismo, la elección está apoyada gracias a la extensa documentación que se puede encontrar en diversas webs y libros sobre los sensores, su conexión y la compatibilidad de la mayoría de ellos con el entorno Arduino IDE, que es más extensa que la compatibilidad de tales sensores en Raspberry.

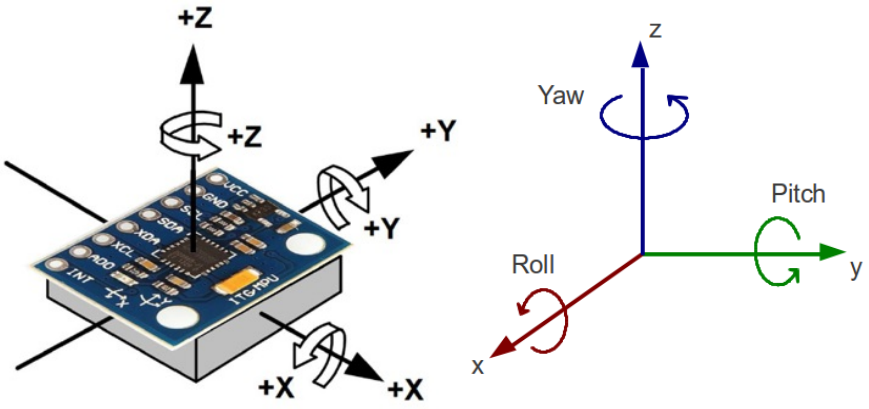
3.2 MPU6050

Como se describe anteriormente, el sensor MPU6050 es una IMU (Unidad de Medida Inercial), los diferentes tipos de IMU que existen en el mercado se diferencian por el tipo de sensores que poseen, generalmente acelerómetro, giroscopio y magnetómetro. Para este proyecto, cuyo principal cometido es el cálculo del ángulo de giro y posición, se ha elegido el MPU6050, ya que a diferencia del MPU6090, dispone de acelerómetro y giroscopio, pero no de magnetómetro, pues no es necesario conocer la dirección con respecto al eje gravitacional de la tierra.

Actualmente los MPU-60X0 [24] son los IMU más utilizados en los dispositivos móviles y tablets actuales, ya que permiten añadir experiencias muy variadas al usuario, las cuales van desde reconocimiento de gestos, mejora en la experiencia de juego, precisión en realidad aumentada a la hora del cálculo de posición, captura panorámica de fotografías, captura de movimientos (Motion tracking) y navegación entre otras.

La serie MPU-60X0 fue de las primeras en integrar la detección de movimientos en seis ejes combinando el acelerómetro y giroscopio, ambos de tres ejes, además de un procesador de movimiento digital (DMP) que permite guardar datos para transmitirlos o hacer uso de ellos a posteriori.

El sensor MPU6050 permite el cálculo de los ángulos de giro en los tres ejes, además de la posición, la cual se comentará en profundidad en el capítulo 4. Su coste varía en el rango de 1 a 3 euros.



Modelo 3D del acelerómetro MPU6050 y ángulos de giro

Antes de definir las características, caben destacar los siguientes conceptos sobre las especificaciones:

- **Rango de trabajo:** Máxima medida que soporta de lectura. Para los acelerómetros la medida es la gravedad estándar (9.80665 m/s2), en el caso de los giroscopios la medida se define en grados/segundos puesto que lo obtenido es la velocidad angular.
- **Sensibilidad:** Relación de la variación de magnitud que se produce entre entrada y salida. A mayor rango de trabajo, menor sensibilidad.
- **Ancho de banda de respuesta:** Frecuencia a la que funciona el sensor, que viene definida por el reloj que utilice el procesador del sensor, por el conversor analógico-digital y por la velocidad de transmisión. Medida en bits por segundos.

Dentro de las especificaciones técnicas más importantes caben destacar:

MPU6050	
Voltaje de entrada	2.3V - 3.4V
Salidas	I^2C digital
Acelerómetro	• Salida digital de 3 ejes con rango de trabajo de ±2g, ±4g, ±8g y ±16g. • Corriente de funcionamiento normal del acelerómetro:500 μA. También permite un funcionamiento de baja potencia a 1.25Hz, 20μA a 5 Hz, 60μA a 20 Hz y 110μA a 40Hz.
Giroscopio	• Salida digital X-, Y- y Z-, con una sensibilidad de ±250, ±500, ±1000, ±2000 º/s. • Sensor de temperatura digital con un error de ±1%(calibrado de fábrica), lo que mejora la estabilidad y reduce la necesidad de calibración manual.

Datos adicionales	• Algoritmos internos para la calibración del dispositivo en el propio sensor. • Procesador integrado (DMP- Digital Motion Processing). • Permite al usuario programar filtros para el giroscopio • acelerómetro y temperatura. • Buffer FIFO de 1024 bytes para la recolección de datos.

3.3 Kinect

Como se verá más en profundidad en el capítulo 4, el cálculo de la posición en los ejes X, Y, Z a través del MPU6050 contiene un margen de error lo suficientemente alto para plantear una alternativa al mismo, de modo que se ha seleccionado otro sensor de bajo coste, con un margen de error menor para nuestro proyecto, la cámara Kinect.

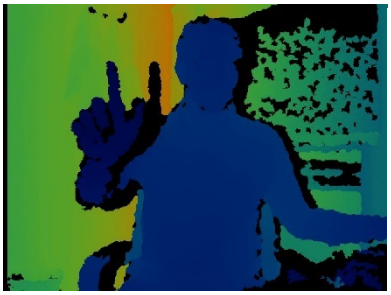
Kinect es una cámara fabricada por Microsoft para su consola Xbox 360. Actualmente existe en el mercado la versión 2.0, pero se hará uso de las primeras versiones dado que la baja acogida en el ámbito de los videojuegos ha provocado que su coste se redujera a un coste mínimo, además de contar con un SDK, librerías y API mejor documentadas y con un mayor número de ejemplos, tanto teóricos como prácticos. El coste actual de una cámara Kinect oscila entre 10 y 25 euros.

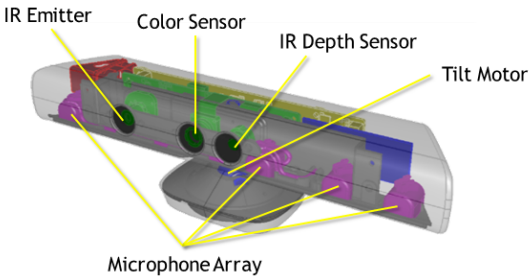
Este controlador permite un reconocimiento completo del cuerpo, cara y voz. Dispone de emisor de infrarrojos que permite calcular la profundidad a través de la distorsión producida en el patrón de las imágenes reflejadas.

Se programa a través de una librería llamada OpenNI [28], de modo que permite a desarrolladores la abstracción del hardware y algoritmos de análisis de escenas, reconocimiento de gestos, de pose y demás, simplificando el uso de dicho sensor.

Cabe destacar que contiene un acelerómetro con un rango de trabajo de 2g que permite determinar la orientación actual de la cámara Kinect, cuatro micrófonos para

capturar el sonido, lo que permite reconocer la posición desde donde se emite el sonido y una cámara RGB que captura imágenes en color a una resolución de 1280x960.





Las especificaciones son las siguientes:

Kinect	
Rango de inclinación vertical	±27 grados.
Ángulo de visión	43 grados vertical y 57 grados horizontal.
Fotogramas por segundo, Imagen y color	30 FPS, RGB, 1280x960 resolución.
Formato de audio	16-kHz, 24-bit mono con Modulación de código por pulso (PCM).
Entrada de audio	4 micrófonos con convertidores analógicos-digital y preprocesamiento interno para la cancelación de eco y supresión de ruido.
Características del acelerómetro	Acelerómetro de 2G/4G/8G de rango de trabajo configurado a 2G.

3.4 Flexómetro

El sensor de flexión o flexómetro SPRK-Flex [4] permitirá el cálculo del ángulo de flexión generado por la falange proximal y media a través de la resistividad del mismo. Este sensor está compuesto por materiales de carbono, lo que le permite flexionarse sin romperse, la curvatura formada provoca la variación de una resistencia eléctrica, de manera que, a más curvatura, mayor valor de resistencia.

Estos dispositivos se usan en dos casos habitualmente, el primero para calcular la nivelación y el segundo para medir el ángulo de flexión, que permite medir de 0 a 90 grados.

A lo largo de este proyecto se utilizará el segundo caso para el cálculo del movimiento de los dedos de una mano.

El uso inicial de este tipo de dispositivo fue para los airbags, pero se han utilizado en diversos campos como medicina, periféricos, terapias físicas, etc. En el mercado se pueden encontrar dos tipos de tamaños, de 5.5 cm y de 11.5 cm, con un coste oscilante entre 8 y 15 euros.

Las especificaciones son las siguientes:

SPRK-Flex	
Resistencia en plano	25K Ohm
Rango de temperatura	-35º a 80º
Tolerancia de la resistencia	±30%
Rango de resistencia al doblado	45K Ohms a 125K Ohms
Consumo	0.50 Watts en continua, 1 Watt en pico

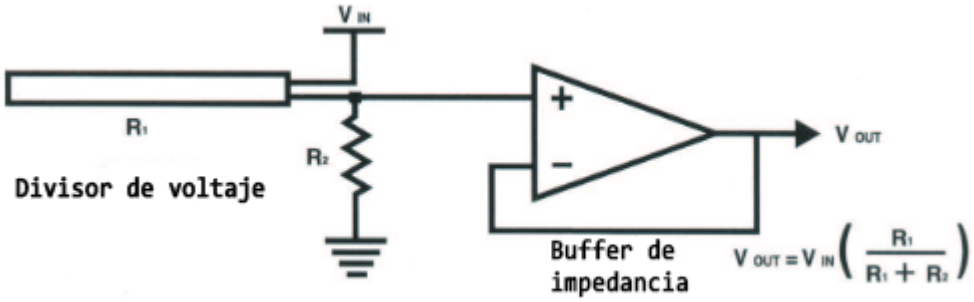


Diagrama de circuito del sensor SPRK Flex

3.5 Unity

Unity es una plataforma de desarrollo de aplicaciones multiplataforma, con motor gráfico OpenGL para Windows, Mac y Linux, Direct3D para Windows, y OpenGL ES para Android e iOS. Esta plataforma incluye un sistema físico propio, a través del motor PhysX, permite la interconexión a través del puerto serial con librerías nativas y cuenta con una solución de control de versiones incluida en el mismo, la cual utiliza una base de datos PostgreSQL.

La elección de esta plataforma de desarrollo se justifica dadas las siguientes características:

- Producción de aplicaciones multiplataformas a través del mismo programa, sin necesidad de desarrollo nativo individual.

- Lenguaje de alto nivel C#, con documentación y una API extensa.

- Posibilidad de uso conjuntamente con programas de diseño 3D tales como Blender, 3ds Max, Maya, Adobe, SketchUp, etc. Asimismo, destaca el soporte para el mapeo de terrenos, reflejos, sombras y demás elementos visuales.

- Compatibilidad propia con sistemas de realidad virtual y realidad aumentada.

- Gran cantidad de ejemplos y librerías con licencias GNU.

Capítulo 4. Algoritmos de cálculo del movimiento 3D

Aquí se procederá a estudiar cómo conseguir todos los movimientos y cómo pasarlos a un programa externo, en este caso Unity, para poder interactuar con ellos. Cada apartado dispone de una demo evolutiva, de manera que el último apartado contendrá una demostración de todos los posibles movimientos.

Se separará este capítulo en las siguientes partes:

- **Datos generados por MPU6050 y calibración:** Se determinan las mediciones sin procesar generadas por el sensor, cómo se leen y cómo se pueden mejorar calibrando este dispositivo.
- **Ángulos de giro:** En este apartado se calcula el ángulo producido por el IMU MPU6050 en los ángulos X, Y, Z.
- **Posición:** Se explican las diferentes posibilidades de cálculo de la posición.
- **Flexión:** A través del sensor de flexión se determinará el ángulo generado por la flexión de la mano.

Antes de empezar cada sección, se deben tener en cuenta algunos conceptos iniciales. Lo primero a recordar es que la aceleración es la variación de la velocidad en el tiempo.

a = dV/dt

Así como la segunda ley de Newton, “La aceleración de un objeto es directamente proporcional a la fuerza neta que actúa sobre él e inversamente proporcional a su masa.”

$$\sum \vec{F} = m\vec{a}$$

Este concepto se utiliza para medir la aceleración en el sensor MPU6050, que dispone de placas capacitivas unidas a resortes minúsculos que se mueven internamente conforme a las fuerzas de aceleración que actúan sobre el mismo, a este sistema se le llama masa-resorte.

En segundo lugar, se debe de tener en cuenta que dicho cálculo se puede obtener debido a que el sensor MPU6050 internamente posee un MEMS (MicroElectroMechanical Systems). Aunque no exista movimiento, siempre podrá

calcular dicho parámetro ya que el acelerómetro siempre estará ligado a la aceleración de la gravedad. A través de esta variable se pueden conocer otras, tales como la velocidad, si es integrada con el tiempo, y la posición, si se dispone de la posición inicial.

Una vez aclarados los conceptos, se procede a evaluar los resultados obtenido por el sensor utilizando la librería propia del mismo desarrollada por Jeff Rowberg [29].

4.1 Datos generados por MPU6050 y calibración

Se ha de tener en cuenta que el modo Sleep debe estar deshabilitado para que los registros del acelerómetro y del giroscopio puedan ser leídos, simplificando el código de la documentación oficial del sensor, adjunto en **"medicionesSinProcesarMPU6050.ino"**, y conectando la placa Intel Edison con Arduino Board con el sensor MPU6050, con las conexiones siguientes:

MPU6050	Arduino	Descripción
Vcc	5V	Cabe destacar que para múltiples sensores MPU6050 se puede dividir el voltaje en 3.3V para su conexión.
GND	GND	Toma de tierra, 0V.
SCL	A5	Conexiones estándar del modo I2C.
SDA	A4	

Puesto que los datos a leer serán los rangos de trabajos correspondientes al acelerómetro y al giroscopio por defecto **(2g y 250º/s)**, se hará uso de las siguientes ecuaciones para convertir dichas lecturas en un valor de aceleración (m/s^2) o velocidad angular(º)

ax = ax_raw * (9.81/16384.0);

gx = gx_raw * (250.0/32768.0);

Para entender las ecuaciones anteriores se ha de tener en cuenta que la escala de salida del sensor MPU6050 por defecto es de **[-32768, +32768]** para cada uno de los seis ejes, tanto acelerómetro como giroscopio. Como se mencionó anteriormente, por defecto se utiliza **+/-2g** y **+/-250º/s**, por lo que la lectura en un caso ideal, es decir, perfectamente nivelado y sin ningún tipo de movimiento ni fuerza externa, debería de otorgar unos valores de:

- 0 para los ejes X e Y.
- 1g para el eje Z, el cual corresponde a 16384 con la sensibilidad 2g predeterminada.
- 0 para las lecturas del giroscopio para los tres ejes.

De lo anterior se deduce que, si el filtro se encuentra bien calibrado, los datos obtenidos en el eje Z de la aceleración deben aproximarse a 9.81.

En el código adjunto con el nombre **“CalibracionMPU6050.ino”** se puede encontrar el algoritmo usado para la medición de dichas medidas. Aplicando dicho algoritmo se obtienen las siguientes mediciones sin procesar, con el sensor en posición horizontal, sobre una mesa de madera, sin ruido externo, sin modificación de temperatura por luces cercanas, y a una temperatura media de 21º.

Acelerómetro (m/s^2)			Giroscopio (º)		
X	Y	Z	X	Y	Z
0.90	-0.29	10.23	2.13	-2.27	-0.79
0.83	-0.23	10.28	2.21	-2.14	-0.75
0.81	-0.23	10.18	2.33	-2.21	-0.79
0.83	-0.27	10.27	2.15	-2.54	-0.66
0.89	-0.26	10.30	2.40	-2.23	-0.74
0.81	-0.23	10.26	2.27	-2.30	-0.55

Como se puede observar, lo datos anteriores contienen un alto error si se observa la medición del eje Z en la aceleración, el cual dista de 9.81 como medida ideal. A través de la media de las medidas anteriores sobre dicho eje podemos estimar el error en ±0.44.

Esto se debe a numerosas razones, como que la sensibilidad por defecto es muy alta, y que el sensor devuelve 16 bits, pero el número actual de bits es menor que 16, además de otras como la posición inicial, temperatura exterior, etc, de manera que se debe encontrar una manera de calibrar los datos anteriores.

Esta calibración debe hacerse dado que el sensor no sabe en qué posición se encuentra, por lo que se necesita calcular el offset para compensar los errores de medición una primera vez.

Se procede a generar un nuevo programa capaz de calcular el offset necesario para poder calibrar el sensor y que permita obtener mediciones más precisas. Para ello se generará un algoritmo que se encargue de eliminar el error constantemente hasta conseguir la medida deseada, la cual es 0 en todos los ejes, excepto la aceleración de Z, que debería ser g=9.81.

Colocando el sensor en posición horizontal, primero se visualiza por pantalla el offset actual para ver si se ha sido calibrado anteriormente o hace falta calibrar. A continuación, se procederá a leer los datos del acelerómetro y giroscopio, y a través de un filtro que se encarga de estabilizar las lecturas, se calcula el promedio constantemente de la siguiente manera:

filtro_ax = filtro_ax -(filtro_ax/32)+ax;

promedio_ax = filtro_ax/32;

filtro_gx = filtro_gx - (filtro_gx/8)+gx;

promedio_gx = filtro_gx/8;

Se utilizan treinta y dos medidas para la aceleración puesto que es una variable con más ruido que el rango de trabajo del giroscopio, con lo que 8 ocho lecturas es suficiente para una estimación del promedio. Una vez que el promedio “promedio_gz” se acerque lo suficiente al valor de la sensibilidad 2g por defecto (16384), se obtiene el offset para la calibración, que se aplica posteriormente. Al resetear la placa Intel Edison y con el sensor MPU6050 en posición horizontal y estable, se obtienen unas medidas ya filtradas como las siguientes:

Acelerómetro			Giroscopio		
X	Y	Z	X	Y	Z
0.11	-0.07	9.96	0.24	-0.23	-0.06
0.07	-0.05	9.87	0.26	-0.34	0.00
0.09	-0.03	10.0	0.20	-0.36	-0.03
0.03	-0.09	9.94	0.20	-0.21	-0.05

0.17	-0.04	9.77	0.21	-0.42	-0.01
0.09	-0.03	9.99	0.27	-0.48	-0.08
0.06	-0.06	9.88	0.26	0.03	0.17
0.13	-0.03	10.0	0.11	-0.64	0.02
0.11	-0.08	9.88	0.26	-0.21	0.02
0.12	-0.03	9.89	0.16	-0.12	0.06
0.07	-0.11	9.88	0.07	-0.09	-0.03
0.12	-0.06	9.97	0.33	-0.33	-0.08
0.09	-0.09	9.93	0.22	-0.38	-0.15
0.15	-0.06	9.93	0.29	-0.31	0.01
0.10	-0.06	9.96	0.26	-0.34	0.17
0.12	-0.05	9.93	0.24	-0.50	0.08
0.15	-0.07	9.92	0.09	-0.18	0.01
0.14	-0.06	9.96	0.31	-0.14	-0.08
0.09	-0.06	9.94	0.31	-0.54	0.12
0.13	-0.12	9.94	0.31	-0.54	0.12

Con un Offset aplicado con los siguientes valores:

Offset					
Acelerómetro			Giroscopio		
X	Y	Z	X	Y	Z
296	813	16382	0	8	-1

A través de la media de las primeras veinte lecturas, observando el eje Z de aceleración, se obtiene 9.927 sobre una medida ideal de 9.81, con lo que se observa que el error absoluto una vez filtrado es de **±0.117**, es decir, un porcentaje de error de **1.192%.**

4.1 Determinación de ángulos de giro

Existen tres maneras para el cálculo de los ángulos de giro, la primera consiste en leer los datos de aceleración del acelerómetro, la segunda es a través del giroscopio y por último una combinación de ambas lecturas.

4.1.1 Cálculo a través de la aceleración

Partiendo del conocimiento que la gravedad siempre es vertical, se pueden calcular los ángulos resultantes de la inclinación del plano del sensor.

Es importante tener en cuenta que los resultados del acelerómetro proporcionan ángulos de orientación precisos siempre que la gravedad sea la única fuerza que actúe sobre el sensor.

Sin embargo, al mover y girar el sensor, se le aplican fuerzas externas, lo que provoca una fluctuación en las mediciones, devolviendo un resultado con mucho ruido y perturbaciones breves pero significativas. Las ecuaciones correspondientes para estos cálculos son las que siguen:

$$\rho = \arctan\left(\frac{a_x}{\sqrt{(a_y^2 + a_z^2)}}\right)$$

$$\phi = \arctan\left(\frac{a_y}{\sqrt{(a_x^2 + a_z^2)}}\right)$$

$$\theta = \arctan\left(\frac{\sqrt{(a_x^2 + a_y^2)}}{a_z}\right)$$

Puesto que el actual trabajo consiste en un sistema de captación de movimiento incorporado en un guante, existirán más fuerzas que la gravedad, lo que provoca que este tipo de medición no sea la más recomendada.

4.1.2 Cálculo a través del giroscopio

Partiendo del conocimiento que el giroscopio mide la velocidad angular, se debe proceder a calcular la posición inicial del sensor para poder obtener un ángulo inicial conocido. Posteriormente se medirá la velocidad angular (ω) alrededor de los ejes X, Y y Z a intervalos medidos (Δt). De esta manera se obtiene que el cambio de ángulo es **ω × Δt**, sumando a este el ángulo inicial se consigue el ángulo actual.

$$\theta_x = \theta_{x0} + \omega_x \Delta t$$
$$\theta_y = \theta_{y0} + \omega_y \Delta t$$

El problema de este cálculo es que se basa en la suma de muchos pequeños intervalos computados (integración). La suma repetida de incrementos de ω × Δt dará como resultado pequeños errores sistemáticos que se magnifican con el tiempo, por lo que, aunque el giroscopio proporcione datos precisos sobre el cambio de orientación a corto plazo, los resultados se desviarán en escalas de tiempo más largas.

4.1.3 Cálculo a través de ambos

El uso de otras fuerzas además de la gravedad en el acelerómetro y las acumulaciones de errores generadas por la integración en el tiempo del giroscopio, conocidas como drift, hacen que el error de medición sea más alto de lo que se busca conseguir. La solución a estos problemas consiste en unir los datos recogidos por el acelerómetro y el giroscopio de tal manera que las mediciones de uno ayuden a cancelar el error del otro.

El método estándar se basa en combinar estas dos entradas de datos con un algún tipo de filtro, habitualmente se utiliza el filtro Kalman, el cual tiene un coste computacional elevado, o utilizando un filtro complementario, el cual permite combinar ambas entradas contando con una implementación más simple y óptima para sistemas de bajo nivel de cómputo.

4.1.3.1 Filtro de Kalman:

El filtro de Kalman [30], o filtro de estimación lineal cuadrática (LQE) permite obtener lecturas más precisas mediante la estimación de una distribución de probabilidad conjunta sobre las variables para cada intervalo de tiempo.

El algoritmo del filtro de Kalman es recursivo [31], por lo que puede ejecutarse en tiempo real utilizando solo las medidas de entrada actuales, el estado previamente calculado y su matriz de incertidumbre.

El filtro de Kalman se separa en dos etapas.

- **Predicción**: Se recogen las lecturas actuales y se producen unas estimaciones de las variables de estado actuales junto a su incertidumbre.
- **Corrección**: Donde se combinan las mediciones con los valores reales medidos.

Para resumir, el filtro consiste en leer los datos del sensor y conociendo previamente los valores reales, por cada iteración ajustar dicho error para acercar las lecturas iniciales a las observadas en el mundo real, teniendo en cuenta que el sistema debe ser lineal.

Para los cálculos del filtro de Kalman, se usan dos entradas, las cuales son los datos leídos por el acelerómetro y el giroscopio. Se usa la siguiente fórmula que representa la forma general de un modelo lineal:

$$x_{k+1} = Ax_k + B\mu_k$$

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & -dt \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}_{k} + \begin{pmatrix} dt \\ 0 \end{pmatrix} \mu_k$$

Para este propósito se necesita rellenar la matriz A en B, y elegir un estado x. La variable μ representa la entrada, en este caso son los datos leídos del acelerómetro y giroscopio, para integrar estos, se deben normalizar primero, incluyendo el tiempo entre las dos medidas, de la siguiente manera:

$$\alpha_k = \alpha_{k-1} - (\beta)dt + \mu_k dt$$

Siendo:

C	Matriz de estado, en este caso $(1 \quad 0)'$.
μ	Lectura de los datos de la última medición.
x = Ax + Bμ	Actualización del estado x del modelo.
y	Lectura del ángulo calculado por el acelerómetro.
Inn	Diferencia entre el valor medido y el predicho, llamado Innovación.
s = CPC' + Sz	Cálculo de la covarianza.
K = APC'*inv(s)	Cálculo de la ganancia Kalman.
x = x+K*Inn	Corrección de la predicción de estado.
P = APA' - KCPA' + Sw	Cálculo de la covarianza del error de predicción.
$S_z = E(z_k z_k^T)$	Cálculo de la covarianza del ruido generado en la medición.
$S_w = E(\begin{bmatrix}\alpha & \mu\end{bmatrix}' \cdot \begin{bmatrix}\alpha & \mu\end{bmatrix})$	Matriz de covarianza del ruido de las mediciones.

La matriz C es la que extrae la salida de la matriz de estado, de manera que se puede saber Alpha (**α = Cx).**

Teniendo en cuenta que solo se utilizan los elementos diagonales de la matrix $\boldsymbol{S_w}$, solo se necesita conocer $E(\alpha^2)$ y $E(\mu^2)$, que se calculan observando el ruido en **α** que procede del giroscopio y es multiplicado por dt^2. Con lo que se obtiene:

$$E(\alpha^2) = E(\mu^2)dt^2$$

Para calcular estos factores se deben hacer una serie de pruebas ensayo-error para poder ajustar dichos valores, en este trabajo son:

E(α^2) = 0.001

E(β^2) = 0.003

Sz = 0.3 radianes (17.2 º)

Se puede observar en el código adjunto, propiedad de **KasBot V1** [32], la implementación del algoritmo para Arduino. Al implementarlo y llevar a cabo las filtraciones en Arduino se experimenta una ralentización del sistema dado su limitada capacidad de computo.

Las principales desventajas del filtro Kalman para el actual trabajo son las siguientes:

- Dado que la placa de desarrollo Intel Edison utiliza Arduino como base y su capacidad computacional es baja, es muy difícil implementar dicho filtro en este tipo de sistemas. En este caso al cambiar de ángulo rápidamente, el filtro tarda más de lo normal en completar los cálculos dado su complejidad recursiva, haciendo que el sistema se ralentice de una manera significativa.

- Este filtro debe ser utilizado si previamente se ha detectado cuáles son las fuentes de ruido externos y hayan sido previamente modeladas. Asimismo, los parámetros necesarios para el cálculo del filtro son difíciles de entender y de implementar.

4.1.3.2 Filtro de complemento:

Como se ha estudiado anteriormente, el filtro Kalman es uno de los más utilizados habitualmente en la actualidad, pero necesita de un sistema con un cálculo de cómputo elevado. Para solventar este problema, se utiliza el filtro complementario o Complementary Filter [33].

La ecuación filtro de complemento es la siguiente:

$$\Theta = \alpha(\theta_{giroscopio} + \omega_{giroscopio}dt) + \beta(\theta_{acelerometro})$$

Siendo Θ el ángulo filtrado, **α = 0.97** y **β = 0.3** unas ponderaciones del 97% y 3% respectivamente, las cuales pueden variar siempre que sumen 100% y **dt** una diferencia de tiempo, que se ha establecido en 0.05 segundos.

Con la anterior fórmula, se consigue amortiguar las variaciones bruscas de aceleración y reducir las rotaciones rápidas generadas por el giroscopio.

Dado que la complejidad de dicho algoritmo es menor que la complejidad del filtro de Kalman (ya que es un algoritmo recursivo), se consigue un filtro que ajusta los datos de una manera cercana a la realidad y que no provoca ralentización del sistema en el cálculo del giro.

Antes de aplicar el algoritmo de filtro de complemento debemos tener en cuenta que los sensores a usar deben de estar calibrados, una vez conseguido, el algoritmo consta de las siguientes partes:

1. Obtención de lecturas: Se consiguen las medidas sin procesar, tanto del acelerómetro como del giroscopio.

2. Conseguir el diferencial de tiempo en segundos.

3. Calcular la aceleración de los ejes X y Z utilizando las fórmulas del apartado 4.1.1 y los ángulos de giro con las fórmulas del apartado 4.1.2.

4. Aplicando la fórmula anterior, se obtiene el ángulo en X e Y filtrados. En el caso de Z, solo se necesita actualizar los datos del giroscopio en el tiempo y pasarlos a grados dividiendo por 180.0/3.14.

5. Actualizar la posición anterior de los ángulos calculado.

En el código adjunto **"FiltroComplementrio.ino"**, se puede observar el algoritmo utilizado y sus medidas para diversas rotaciones, asimismo se podrá visualizar la precisión obtenida por este filtro en el capítulo 5, en la demostración titulada **"DEMO 2 - Flexión"**

4.2 Determinación de posición

Existen diversas maneras para calcular la posición de un cuerpo en un sistema de tres ejes, pero puesto que este proyecto se basa en una mano virtual, interesa especialmente dos ejes, X, de izquierda a derecha y Z, de arriba a abajo. Para ello se estudia el cálculo de posición que es posible llevar a cabo a través del sensor que hemos utilizado para los ángulos y a través de otro sensor de bajo coste, el Kinect de Microsoft.

4.2.1 MPU6050

Es posible calcular la posición relativa del sistema utilizando la aceleración, la cual sí se integra con el tiempo para obtener la velocidad, y si se integra nuevamente con respecto al tiempo, es posible obtener la posición [34] [35].

$$\int\int_0^t a(t)dtdt$$

Dado que la aceleración en el eje X e Y es conocida, recordando que no es posible calcular la elevación ya que la gravedad es nula, es posible integrar numéricamente el tiempo sobre cada eje de manera individual.

Utilizando la regla del trapecio, método de integración para aproximar el valor de una integral definida, es posible calcular la posición. Para ello se calcula la aceleración, transformando posteriormente los datos a medidas internacionales, y se aplica la siguiente fórmula, teniendo en cuenta que previamente el sensor debe haber sido calibrado:

$$\vec{v}_k = \vec{v}_{k-1} + \frac{(t_k - t_{k-1})(\vec{a}_k - \vec{a}_{k-1})}{2}$$

$$\vec{p}_k = \vec{p}_{k-1} + \frac{(t_k - t_{k-1})(\vec{v}_k - \vec{v}_{k-1})}{2}$$

El algoritmo a estudiar consta de los siguientes pasos:

- Se inicializa el bus I^2C, el sensor MPU6050 y el puerto de comunicación serie.

- Se obtienen las lecturas del acelerómetro y del giroscopio, convirtiendo estas a unidades internacionales y asignándolas como variables anteriores.

- Se efectúa una espera entre medidas y se calcula la velocidad, ya que se dispone de la aceleración anterior, y a través de esta, se calcula la posición y se imprimen los resultados.

Aplicando dicho algoritmo y midiendo el desplazamiento de 0 a 10 cm reales sobre el eje X, las medidas obtenidas son las correspondientes a la siguiente tabla:

Cm Real	2	4	6	8	10	8	6	4	2	0
Cm Mpu6050	2	5	6	7	11	9	7	5	3	2

Con lo que es posible observar que el error es lo suficientemente alto como para buscar otra alternativa, además del hecho de no disponer de la posición en el eje Z (elevación), lo que provoca que el guante sea poco interactivo, dado que el movimiento de la mano solo se efectuaría en el eje X e Y (similar a un ratón de ordenador), pero en la pantalla se visualizará el desplazamiento de la mano en el eje X, Z.

4.2.2 Kinect

La segunda solución de bajo coste planteada para conseguir el cálculo de posición es el uso de la cámara Kinect, la cual permite calcular la posición en el eje X y Z, a la vez que visualizar tales datos en el entorno de desarrollo Unity.

Kinect provee de librerías y documentación que permite al usuario abstraerse de la programación y gestión de reconocimientos, de manera que creando un nuevo objeto de la clase principal del asset de nombre “Kinect with MS-SDK”, de la empresa **RF Solutions** [36] que se encuentra en la tienda virtual de Unity, es posible obtener un catálogo de funciones y métodos que proveen al desarrollador de herramientas para gestionar qué partes del cuerpo captar, el tipo de movimientos, profundidad, sonido captado y demás funciones.

En el capítulo siguiente se explica en detalle la implementación de la misma sobre Unity.

4.3 Determinación de flexión

Un sensor de flexión es capaz de obtener el grado a través de la resistividad del material del que está fabricado al doblarse el mismo.

Para determinar el ángulo producido, se conecta el sensor a la placa Arduino de manera que la salida analógica capte los datos que produce el mismo y con una resistencia entre el pin GND del sensor y el mismo de la placa Intel Edison.

La resistencia colocada en el circuito es de 1k Ohm, con lo que la salida recibida sin filtrar y en posición de reposo no es fiel al ángulo que realmente está midiendo, por lo que se debe calibrar antes de transmitir los datos a un sistema externo. Para este propósito se calcula la media de las N primeras medidas (50 en este trabajo), en posición de reposo, y posteriormente se imprime a través del puerto serie el valor producido por el valor absoluto del valor leído menos la media de calibración, de manera que se obtiene un valor de 0º a 180º.

Capítulo 5. Montaje y pruebas

Se realizan cuatro demostraciones técnicas en la plataforma de desarrollo Unity, tres individuales para mostrar cada uno de los movimientos recogidos, y una final que contendrá todos ellos. Dichos programas, junto con un video demostrativo pueden ser encontrados adjuntos a esta memoria adjunto con el nombre **"CalibracionMPU6050.ino".**

Una vez calibrado, se hará uso de un algoritmo común para la transmisión de datos por el puerto serial. Para ello se transmiten los datos desde Arduino a través del comando **Serial.print("datos")**, recordando añadir **Serial.flush()** para esperar a que la transmisión de datos de salida termine, además de eliminar cualquier dato del buffer de entrada.

Se ha de considerar que la velocidad de cómputo de un ordenador es mucho mayor que la capacidad de Intel Edison, por lo que se limita ambos códigos para que la lectura sea paralela, para ello se utiliza **stream.ReadTimeout = 50** en Unity y **delay(50)** en el código para la placa Arduino de Intel Edison.

Por otro lado, desde el código de Unity se siguen los siguientes pasos:

- En **File>Build settings>Player Settings>** se coloca **.NET2** de nivel de compatibilidad de API, para poder hacer uso de la librería System.IO.Ports.
- Se Inicializa un objeto de la clase **SerialPort** especificando el puerto y el baudaje, que es el número de unidades de señal por segundo. En el caso de que el puerto fuese mayor que 9, se debe añadir "\\.\" para la correcta creación.
- Se abre el stream con **stream.Open()** y se establece el timeOut de lectura.
- Por último, se recogen los datos leídos con el método **stream.ReadLine()** y se parsean para poder evaluarse por separado.

Teniendo en cuenta lo anterior, que se aplica en el común de los casos, se procede a explicar una demostración técnica y visual de cada uno de los apartados tratados en el capítulo 4, además de una demostración final de uso del conjunto de todos los movimientos.

5.1 Montaje de MPU6050, inclinación y rotación con respecto a los tres ejes cardinales.

Como se ha comentado con anterioridad, a través del sensor MPU6050 obtenemos los ángulos de giro y rotación para los tres ejes. El primer montaje será un ejemplo de uso del mismo en un entorno virtual, para ello se conectará el sensor MPU6050 a la placa Intel Edison a través del Arduino Board, conectando Vcc a 5V, Gnd con Gnd, SCl a la entrada analógica 5 y SDA a la entrada analógica 4.

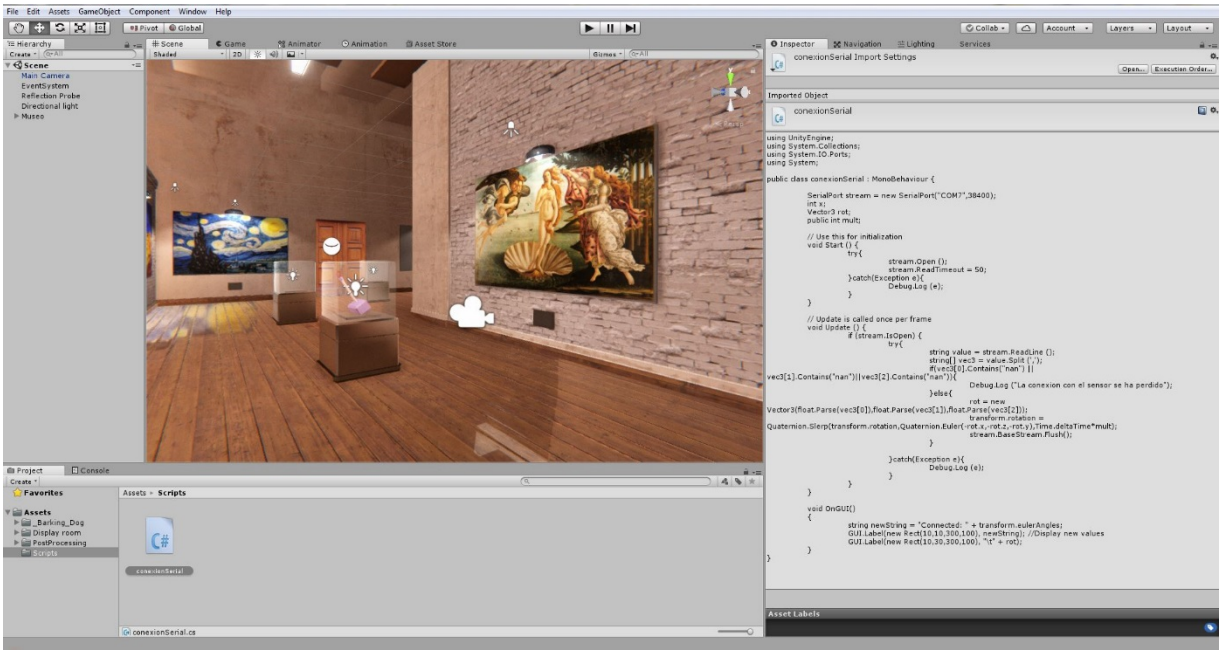
Teniendo en cuenta que el sensor debe haber sido previamente calibrado a través del programa **"CalibracionMPU6050.ino"**, se procede a subir a la placa Intel Edison el programa anteriormente estudiado de filtro complementario, con el nombre **"FiltroComplementrio.ino"**

Una vez subido a la placa, se procede a cerrar el programa Arduino IDE para dejar el puerto COM asignado libre.

Posteriormente se creará en Unity una escena que contendrá un museo virtual, utilizando modelos adquiridos para la misma y una librería propia de Unity, llamada Post Processing Stack para la creación y optimización de luces y efectos visuales. Se añade además una cámara principal a la escena de Unity, la cual contendrá el código que leerá los datos del programa que se encuentra ejecutándose en la placa Intel Edison.

Este código contendrá también un parseador capaz de reconocer las lecturas enviadas a través del puerto serial, separadas por comas y terminadas en retorno de carro, de manera que asignará cada lectura de cada eje al objeto que contenga dicho código, en este caso la cámara principal. De esta manera se logra leer, con un delay de 50 ms, las lecturas del sensor MPU6050 y aplicarlas directamente al giro y rotación de la cámara a través del escenario creado en Unity.

Como se puede observar en el ejemplo adjunto **“DEMO 1 - Acelerómetro”**, cambiando la ubicación del sensor, por ejemplo, a nivel de los ojos en este caso específico, es posible obtener un sistema similar a la realidad virtual, logrando que los movimientos realizados con el sensor se apliquen directamente a la visualización del entorno.



Demostración técnica 1, implementación de rotación y giro sobre la cámara principal

5.2 Montaje de SPRK-Flex y captura de flexión

En este montaje se creará una mano articulada que podrá abrir y cerrar los dedos a través de la flexión producida por estos, para ello se conecta el flexómetro utilizando una resistencia de 1KOhm.

Como se explicó anteriormente, un sensor de flexión es una resistencia variable que nos devuelve unos datos en cuanto esta resistencia aumenta. Al tener una resistencia entre el GND de la placa Intel y el sensor, debemos de tener en cuenta que los datos producidos por el sensor pueden varias, por lo que tenemos que adaptar dichos datos a medidas reales, de 0º a 180º. Por lo que el algoritmo de lectura en Arduino, adjunto con el nombre **“Flexometro.ino”**, es de la siguiente manera:

- Se leen los 50 primeros datos producidos por el sensor en posición sin flexionar.

- Se calcula la media de dichos números.

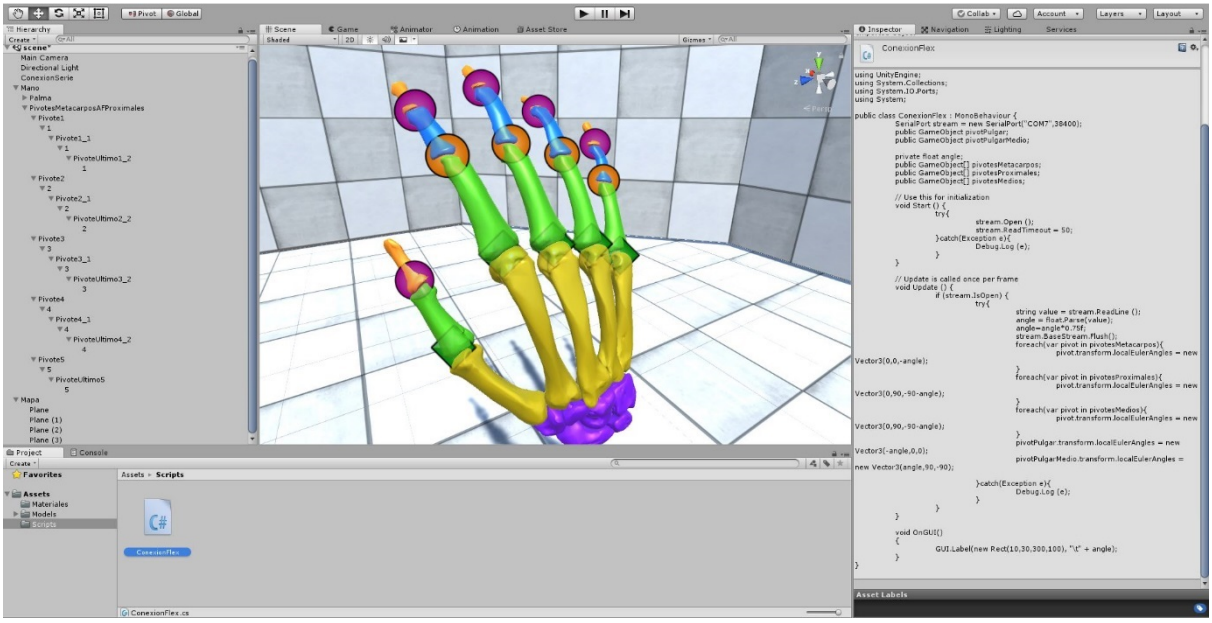
- Se leen los datos nuevamente restando la media anterior y devolviendo su valor absoluto (para evitar lecturas negativas en posición de reposo).

Cabe destacar que es posible añadir un cálculo de error, creando una condición que detecte la diferencia de valor anterior y el actual es un diferencial de tiempo pequeño. Este caso se encuentra comentado puesto que es útil si utilizamos resistencias más pequeñas, ya que se pueden encontrar picos en las lecturas de las mismas.

En la parte de Unity se procede a diseñar una mano, a través de un modelo con licencia GNU obtenido de **archive3D** [37], con una estructura jerárquica para poder controlar la rotación de cada uno de los huesos. La mano se compone de los metacarpos (amarillo), falanges proximales (verde), falanges medias (azul) y falanges distales (naranja). Para poder acoplar los datos recibidos por Arduino en dicha mano, se debe montar una jerarquía de manera que los metacarpos contengan a las falanges proximales, estas a las medias y por último a las distales, permitiendo que el movimiento de una infiera en el movimiento de todos los nodos hijos.

El valor que es medido a través del flexómetro puede variar de 0 grados (sin flexión) hasta 180 (flexión máxima), pero puesto que la falange proximal no puede sobrepasar los 90º físicamente, lo máximo que se podrá leer por el flexómetro una vez acoplado a nuestra mano está entre 0 y dicho valor, así que será aplicado directamente a la rotación del segundo hueso. Asimismo, se aplica también este ángulo de rotación a las falanges medias y distales, sumando posteriormente la posición anterior, puesto que, en la flexión de la mano, cada falange tendrá un ángulo de flexión mayor al anterior.

Aplicando el ángulo leído al eje Y para los cuatro dedos centrales (índice, corazón, anular y meñique), y en el eje X para el pulgar, recordando que dicho hueso no contiene falange media, es posible observar el resultado en el contenido adjunto en **"DEMO 2 – Flexión"**



Demostración técnica 2, calculo de flexión a través de SPRK Flex

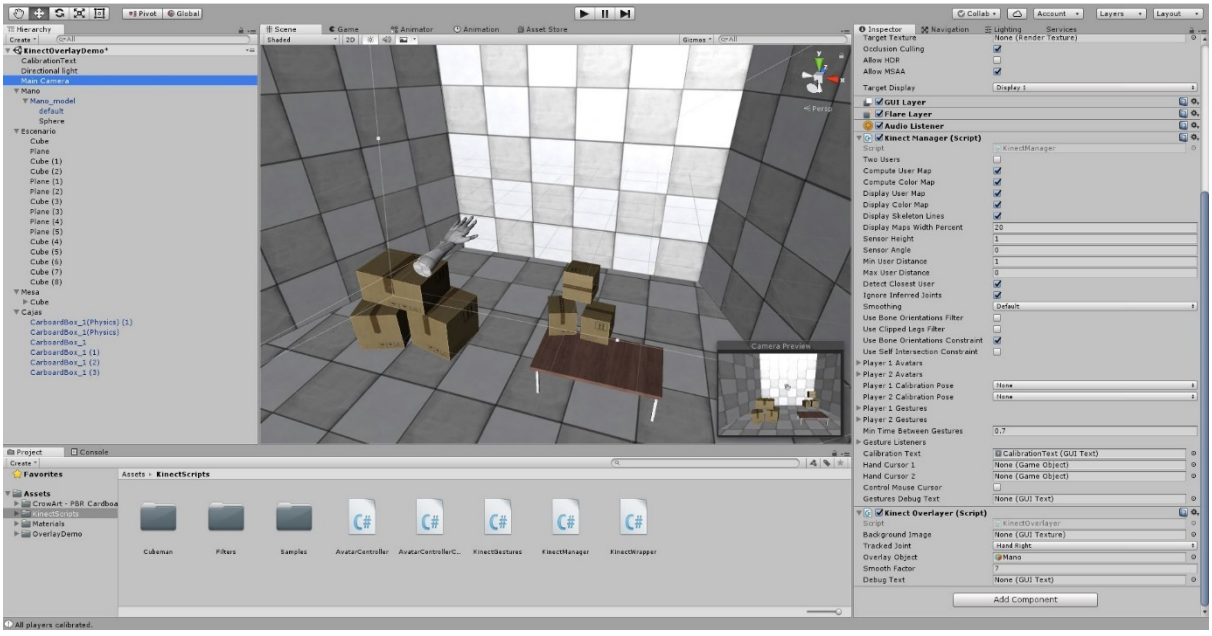
5.3 Montaje de Kinect y determinación de posición en dos ejes

Se ha utilizado la cámara Kinect para el cálculo de la posición en los ejes X e Y debido a su reducido coste y a su kit de desarrollo Open Source. Para poder utilizar dicha cámara debemos instalar primero su SDK y posteriormente descargar un asset gratuito con licencia GNU de la empresa RF Solutions, el cual aporta documentación, un api estable y una interfaz con la que despreocuparse del reconocimiento facial, body-tracking, eliminación del fondo y demás, dejando un nivel de abstracción al desarrollador en el cual solo se debe configurar que parte del cuerpo reconocer, que gestos hacer y cómo incorporarlos en una escena de Unity.

Se procede a configurar un entorno de pruebas, con varios objetos con cuerpo rígido y mesh colliders para la interacción con la mano que se desplazará en el eje X e Y, se añade también un objeto con un modelo de una mano. Este también dispondrá de un collider pero no de un cuerpo rígido, ya que el desplazamiento será controlado por la clase **"KinectManager.cs"**

Se continúa aplicando dicho código a la cámara principal del proyecto y configurando las diferentes opciones que es posible obtener de la cámara, siendo la más importante la que define que parte del cuerpo seguir, conocido como tracking. Se añade además el código **"KinectOverlayer.cs"** y se define el elemento que será el que se moverá cuando detecte nuestra la mano derecha del usuario, además de la suavidad de movimiento del objeto.

Se puede observar una demostración técnica de lo explicado anteriormente en el archivo adjunto **"DEMO 3 – posición"**



Demostración técnica 3, cálculo de posición a través de Kinect

5.4 Montaje de los tres sensores y control de una nave en un sistema de tres dimensiones.

Una vez descrito el montaje de cada sensor por separado, se continua en un proyecto que combina todos los anteriores para aplicarlos al movimiento de una nave espacial en un juego que se encuentra en los tutoriales de la documentación oficial de Unity [38].

Para ese caso, se procede a modificar el código del tutorial de manera que la cámara sea capaz de seguir a la nave desde una perspectiva de tercera persona, permitiendo que se puedan apreciar en esta los movimientos de rotación producidos por el sensor MPU6050, la flexión producida por el sensor SPRK Flex para efectuar los disparos y los movimientos sobre los ejes X e Y que se producen a través de la cámara Kinect.

Para tal cometido, se utiliza el código adjunto con el nombre **"UnionMPU6050Flexometro.ino"**, el cual es una adición del código **"FiltroComplementrio.ino"** donde se añade como parámetro de salida al array de string el ángulo de flexión calculado previamente en el algoritmo **"Flexometro.ino".**

Desde la parte de Unity se procede a modificar la clase que controla el manejo de la nave a la vez que el disparo y rotación.

Para la rotación, sólo se añaden los 3 primeros parámetros que son recibidos en el array (correspondientes al giro del eje X, Y, Z) al cuerpo rígido que contiene la nave, permitiéndole modificar la posición de rotación por cada lectura.

Para el disparo y el movimiento, se modifican dentro de la clase **"ConexionSerial.cs" (**Aplicada al objeto Nave) los métodos de disparo y de movimiento, de manera que los activadores de dichos eventos son los leídos por el sensor de flexión y la cámara Kinect.

Se puede visualizar el resultado de esta composición en el video y proyecto adjunto con el nombre **“DEMO 4 - Unión”.**

Captura de la demo técnica que une posición(Kinect), rotación(MPU6050) y disparo con flexión(SPRK-Flex)

Capítulo 6. Conclusiones y posibles líneas futuras

6.1 Conclusiones

El objetivo de este proyecto era estudiar el montaje de un sistema de captación de movimientos con sensores de bajo coste y estudiar su viabilidad, asimismo estudiar posibles líneas futuras y ámbitos a los que se pueden aplicar dichos movimientos.

Como se puede observar en el capítulo 4, los errores recogidos en el cálculo de la posición hacen inviable el uso del sensor MPU6050 para sistemas donde la precisión es una parte importante. Por el contrario, el cálculo obtenido después de aplicar el filtro complementario es casi perfecto, de manera que es viable el uso de este sensor en cálculos de giros.

Una de las desventajas encontradas en el sensor MPU6050 es la imposibilidad de calcular el ángulo del eje Z con precisión. Pero dado que este trabajo consiste en la captación de movimientos a través de un modelo de un guante, la imprecisión de este eje no impide su uso.

Con respecto al uso de la cámara Kinect, se han encontrado diversos problemas de compatibilidad con el sistema operativo Windows 7, recogidos en la web de la documentación del mismo, dichos problemas aún no han sido solucionados de manera que se ha de disponer de Windows 10 o Mac para el correcto uso de la librería correspondiente.

El sensor SPRK Flex ha demostrado ser un sensor capaz de medir con precisión los ángulos de flexión entre 0 y 180 grados, siempre que se filtren los valores de la resistencia previamente.

6.2 Ámbitos aplicables

Este tipo de detección de movimientos puede ser aplicadas a diversos ámbitos. A través de ejemplos se demostrará el posible uso de los mismos:

- Medicina: Los datos obtenidos a través de este sistema podrá permitir el estudio posterior de los movimientos, lo que ayudará a diagnosticar diferentes síntomas o patrones en los movimientos de ciertos pacientes con enfermedades psicomotrices o neuromotoras.
- Fisioterapia: Usando los movimientos captados en tiempo real se podrá generar programas donde el paciente podrá interactuar con objetos mediante tal sistema y de esta manera paliar síntomas de múltiples dolencias, tanto agudas como crónicas, por medio del ejercicio.
- Prótesis: Es posible copiar los movimientos generados por una mano en otra protésica, la cual a través de sensores sea capaz de generar los mismos movimientos. Esto permitirá a usuarios con disfuncionalidades nuevas formas de interacción.
- Gamificación/Ocio: La capacidad de poder emular movimientos de una mano en un sistema virtual permite al jugador una sensación de inmersión dentro de un mundo virtual donde él pueda interactuar con objetos del mismo.

6.3 Futuras líneas de actuación y problemas encontrados

Con respecto a las futuras líneas de actuación, se puede destacar el uso de los filtros. Se ha demostrado que el filtro de complemento tiene unos resultados muy similares a la realidad, aunque no se ha podido comprobar el filtro Kalman debido a la falta de computo de la placa Intel Edison, hay que volver a comentar que es uno de los filtros más usados a día de hoy dada su precisión, en sistemas capaces de llevar a cabo dicho cómputo recursivo.

Sobre la Kinect hay un aspecto importante a incluir en esta memoria, y es la actualización de la misma a su versión 2.0. la documentación asociada a esta es la misma que la primera versión de Kinect, y mejora significativamente el rendimiento de esta, además de la resolución, color y profundidad de la misma [39].

El último sistema por comentar es Intel Edison, que se ha descontinuado en junio de 2017, dado que su comercialización y documentación, tanto profesional como de la comunidad, fue muy reducida. Dicha placa tiene unas buenas características, aunque tiene un fallo comprobado en la conexión USB de muchas de estas placas, lo que provoca la continua desconexión de ésta al ordenador.

Bibliografía

[1] «Intel Edison Documentación,» 2017. [En línea]. Available: https://software.intel.com/es-es/iot/hardware/edison/documentation.

[2] «Unity3D Documentación,» [En línea]. Available: https://docs.unity3d.com/es/current/Manual/index.html.

[3] «MPU6050 Datasheet,» [En línea]. Available: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[4] «SPRK Flex Datasheet,» [En línea]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf.

[5] «Arduino Uno Documentación,» [En línea]. Available: https://datasheet.octopart.com/A000066-Arduino-datasheet-38879526.pdf.

[6] K. D. y. API. [En línea]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271(v=ieb.10).

[7] «Gamificación, definición,» [En línea]. Available: https://rua.ua.es/dspace/bitstream/10045/39195/1/Gamificacio%CC%81n%20(definicio%CC%81n).pdf.

[8] «Nintendo Power Globe, Reach out and touch your data,» [En línea]. Available: https://www.microsoft.com/buxtoncollection/a/pdf/BYTE%20Glove%20Comparision.pdf.

[9] «Wiimote, how it works,» [En línea]. Available: http://wiibrew.org/wiki/Wiimote#SDP_information.

[10] «HTC Vive Guidelines for developers,» [En línea]. Available: https://dl.vive.com/Tracker/Guideline/HTC_Vive_Tracker_Developer_Guidelines_v1.3.pdf.

[11] «Acceleglove, web,» [En línea]. Available: http://metamotion.com/images/AcceleGloveUserGuide.pdf.

[12] «Leap Motion documentación,» [En línea]. Available: https://developer.leapmotion.com/documentation.

[13] «Definiciones de posición,» [En línea]. Available: https://www.zettlex.com/es/articles/precision-resolucion-repetibilidad/.

[14] «Sensores Ópticos,» [En línea]. Available: https://www.elprocus.com/optical-sensors-types-basics-and-applications/.

[15] «Imagen Optical Sensors,» [En línea]. Available: https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=2ahUKEwjcwsK4kPXbAhWGShQKHbYMD-

oQjxx6BAgBEAI&url=https%3A%2F%2Fwww.elprocus.com%2Foptical-sensors-types-basics-and-applications%2F&psig=AOvVaw0IU6OS3keZOjbRy6xNjrT1&ust=1530233040598644.

[16] «Ejemplo de sensor Magnético,» [En línea]. Available: https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf.

[17] «Histéresis Magnética,» [En línea]. Available: https://es.wikipedia.org/wiki/Hist%C3%A9resis.

[18] «Sensores capacitivos documentación,» [En línea]. Available: http://media.automation24.com/datasheet/es/KI6000.pdf.

[19] «Sensor Capacitivo Imagen,» [En línea]. Available: https://www.emaze.com/@ATTLRCOI/Technology-Upgrade-copy1.

[20] «Sensores Inductivos LVDT,» [En línea]. Available: https://www.solartronmetrology.com/service-and-support/knowledge-base/lvdt-half-bridge-and-digital-transducer-theory.

[21] «Sensores de ultrasonidos,» [En línea]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.

[22] «MEMS,» [En línea]. Available: https://en.wikipedia.org/wiki/Microelectromechanical_systems.

[23] «Efecto Coriolis,» [En línea]. Available: https://es.wikipedia.org/wiki/Efecto_Coriolis.

[24] «images de ejemplo rotaciones,» [En línea]. Available: https://javighawk.wordpress.com/2014/10/16/pid-controller-back-to-the-bxdrone/.

[25] «Documentación IMU 60X0,» [En línea]. Available: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[26] «IoT,» [En línea]. Available: https://en.wikipedia.org/wiki/Internet_of_things.

[27] «Intel Edison IDE's recomendados,» [En línea]. Available: (https://software.intel.com/es-es/iot/hardware/edison/documentation).

[28] «Raspberry Datasheet,» [En línea]. Available: https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM_1p0.pdf.

[29] «Kinect, a brief explanation,» [En línea]. Available: http://www.cs.upc.edu/~virtual/RVA/CourseSlides/Kinect.pdf.

[30] «Libreria de MPU650 de Jeff Rowberg,» [En línea]. Available: (https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050).

[31] «Kalman web oficial,» [En línea]. Available: http://www.cs.unc.edu/~welch/kalman/.

[32] «Explicacion de filtro de Kalman,» [En línea]. Available: http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/.

[33] «Git de KasBot V1,» [En línea]. Available: https://github.com/jdorweiler/BalancingRobotDC/blob/master/kalman.ino.

[34] «Complementary filter ad Kalman,» [En línea]. Available: http://robottini.altervista.org/kalman-filter-vs-complementary-filter.

[35] «Position Estimation using IMU,» [En línea]. Available: http://iieng.org/images/proceedings_pdf/AE0516306.pdf.

[36] «Using Accelerometers to Estimate Position and Velocity,» [En línea]. Available: http://www.chrobotics.com/library/accel-position-velocity.

[37] «RF Solutions,» [En línea]. Available: https://www.rfsolutions.co.uk/.

[38] «Archive 3D, models free with GNU Licence,» [En línea]. Available: https://archive3d.net/.

[39] «Space Shooter Tutorial,» [En línea]. Available: https://unity3d.com/es/learn/tutorials/s/space-shooter-tutorial.

[40] «Kinect with SDK, another point of view,» [En línea]. Available: https://rfilkov.com/2013/12/16/kinect-with-ms-sdk/.