Facultad de Ciencias

SISTEMA DE POSICIONAMIENTO CON FUSIÓN DE SENSORES PARA ENTORNOS XR

(Positioning system with sensor fusion for XR environments)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Hugo González Allende

Director: Eugenio Villar Bonet

Junio – 2022

Tabla de contenido

1. Resumen 3

1.1. Palabras clave 3

2. Abstract 3

2.1. Keywords 3

3. Introducción 4

3.1. Tecnología XR 4

3.2. Evolución del mercado XR 5

3.3. Estado del arte 7

3.3.1. Azure Kinect 7

3.3.2. Oculus Rift 9

3.2.3. Meta Quest 2 10

3.3.3. Ultra Wideband 13

3.4. Motivación 16

4. Proyecto 16

4.1. Hardware utilizado 16

4.2. Estado inicial del proyecto 17

4.3. La Deriva 18

4.4. Desarrollo 19

4.4.1. Sistema óptico 20

4.4.2. Sistema inercial 21

4.4.3. Sistema de comunicación 26

4.4.4. Fusión 27

4.4.5. Implementación Unity 27

4.5. Metodología de modelado 30

5. Resultados 34

6. Conclusiones 40

7. Trabajo futuro 40

8. Bibliografía 41

1. Resumen

Se propone una implementación para la problemática del posicionamiento preciso en interiores con especial énfasis en aplicaciones de realidad mixta, virtual o aumentada (XR) mediante un sistema inercial complementado por otro sistema óptico para, junto con otras técnicas, lograr contener y corregir la deriva y mitigar otros errores de estimación. El objetivo de conseguir una posición lo suficientemente precisa para este tipo de aplicaciones, utilizando conceptos menos costosos computacionalmente que los empleados en el estado del arte.

1.1. Palabras clave

UMI, inercial, posicionamiento, realidad virtual, realidad aumentada, realidad mixta, Unity, sensor, fusión sensórica, cámara, deriva, estado del arte.

2. Abstract

An implementation is proposed for the problematic of precise positioning indoors with special emphasis on mixed, virtual or augmented reality (XR) applications through an inertial system complemented by an optical system to, alongside other techniques, manage to contain and correct drift and mitigate other estimation error. The aim is to achieve a sufficiently precise positions for this type of application, using concepts that are less computationally expensive than those used in the state of the art.

2.1. Keywords

IMU, inertial, positioning, virtual reality, augmented reality, mixed reality, Unity, sensor, sensor fusion, camera, drift, state of the art.

3. Introducción

3.1. Tecnología XR

La tecnología XR o tecnología de realidad extendida, comprende todas las tecnologías especializadas en la inmersión del usuario existentes como la realidad virtual, la realidad aumentada y la realidad mixta, y las que están por existir en este ámbito.

La realidad virtual es aquella que genera un entorno envolvente 360º completamente virtual, sin interacción del medio físico más allá del propio usuario. El ejemplo más conocido de este tipo de tecnología es, sin lugar a duda, la idea que tiene el usuario general sobre el metaverso, un mundo virtual.

En cuanto a la realidad aumentada, no consiste en generar un entorno envolvente como en el caso de la realidad virtual, sino que, en este caso, son los elementos virtuales los que complementan al entorno real, pero es este último el que predomina. Tenemos un ejemplo reciente que, literalmente, fue un fenómeno global en su salida en el año 2016 en el cual se podía ver a diario millones de usuarios por la calle con sus dispositivos móviles capturando pokemons mediante la aplicación “Pokemon GO”. En esta aplicación, se podían ver estas criaturas a través del dispositivo móvil e interactuar con ellas de manera limitada.

La realidad mixta es una de las menos conocidas y exploradas por el usuario corriente, pero es de las que más potencial tiene. En este tipo de realidad extendida, se combinan los elementos reales y los virtuales difuminando la capa que los separa y haciéndolos de esta manera partícipes de una misma realidad. La mayor diferencia con la respecto a la realidad aumentada, es el nivel de interacción y el grado de inmersión, que es superior en este tipo de realidad extendida. Un ejemplo claro de visualizar de este tipo sería la representación de configuraciones de amueblado de un edificio, el usuario estaría en la vida real caminando por un edificio vacío físico real y, mediante algún tipo de visor MR (“Mixed Reality”), podría estar visualizando muebles virtuales en ese edificio real para hacerse a la idea de cómo quedaría, ayudar en la decisión de compra etc.

Entre ellas hay muchas similitudes y diferencias, sin embargo, nos vamos a centrar en el tema que nos atañe, el posicionamiento. Todos estos tipos de

realidad requieren de un sistema de posicionamiento para localizar al usuario dentro de la experiencia y/o al resto de los elementos virtuales que se añaden o recrean. Por ejemplo, el “Pokemon GO” requiere de GPS siempre activo para funcionar. El caso de realidad virtual, podemos pensar en el metaverso, requiere de algún sistema que traslade el movimiento relativo del usuario al mundo virtual, bien mediante algún tipo de mando, o bien mediante movimiento real del usuario, más adelante en este documento se verán varios métodos y el propio proyecto se centra en este tipo, aunque es extrapolable para la realidad aumentada y mixta. Para el ejemplo de realidad mixta dado, el sistema de posicionamiento sería muy similar/idéntico al de realidad virtual.

Como podemos ver, el sistema de posicionamiento es una parte crítica y vital para el correcto funcionamiento de estas tecnologías y, por lo tanto, su estudio y desarrollo tiene una gran importancia.

3.2. Evolución del mercado XR

Las tecnologías y visores XR han sufrido avances muy importantes en los últimos años que se han reflejado en el mercado.

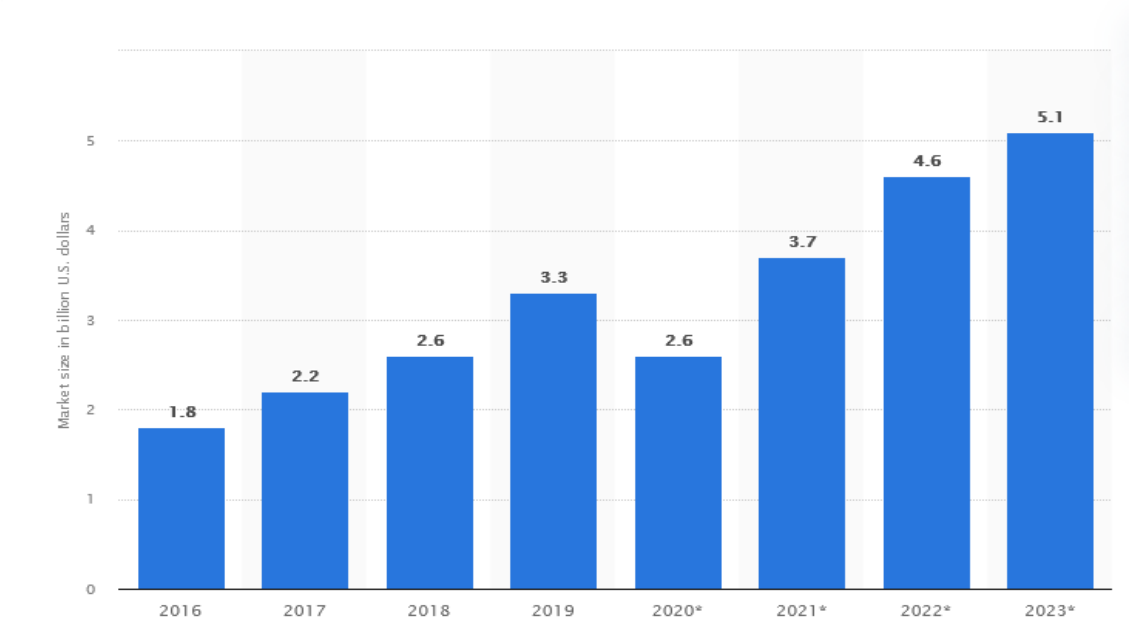

Figura 1. Tamaño de mercado XR. [1].

Como podemos ver en la Figura 1, el crecimiento ha sido relativamente lineal desde el año 2016 hasta el año 2019, a partir de ese año, ya no son datos, sino estimaciones. Se proyecta que siga aumentando en los próximos años de una manera similar, por supuesto, teniendo en cuenta las condiciones especiales

debido a la pandemia del COVID-19, que como se aprecia, ha condicionado en gran medida la predicción para el año 2020, y para los años posteriores probablemente se ha dado una predicción excesivamente optimista ya que casi nadie se imaginaba que la pandemia fuera a durar tanto tiempo y, por lo tanto, podemos esperar que los resultados reales sean inferiores a las predicciones. Pese a todo esto, la tendencia es indudablemente al alza.

Este crecimiento ha sido impulsado en gran parte debido a la maduración del medio y al mayor número de aplicaciones y mejora de la calidad de estas. Por otro lado, los propios visores XR también han mejorado sustancialmente, siendo de más fácil acceso y significativamente más potentes y convenientes.

Como observamos en la Figura 2, el mercado por número de aplicaciones es dominado por el ámbito comercial (visualizaciones inmobiliarias, de vehículos…) y doméstico (entretenimiento), aunque también es utilizado en otros sectores como el aeroespacial o sanitario, en los cuales es una herramienta muy importante para abaratar costes a la vez que proporcionar una formación de mayor calidad a futuros trabajadores mediante simulaciones más realistas e inmersivas.

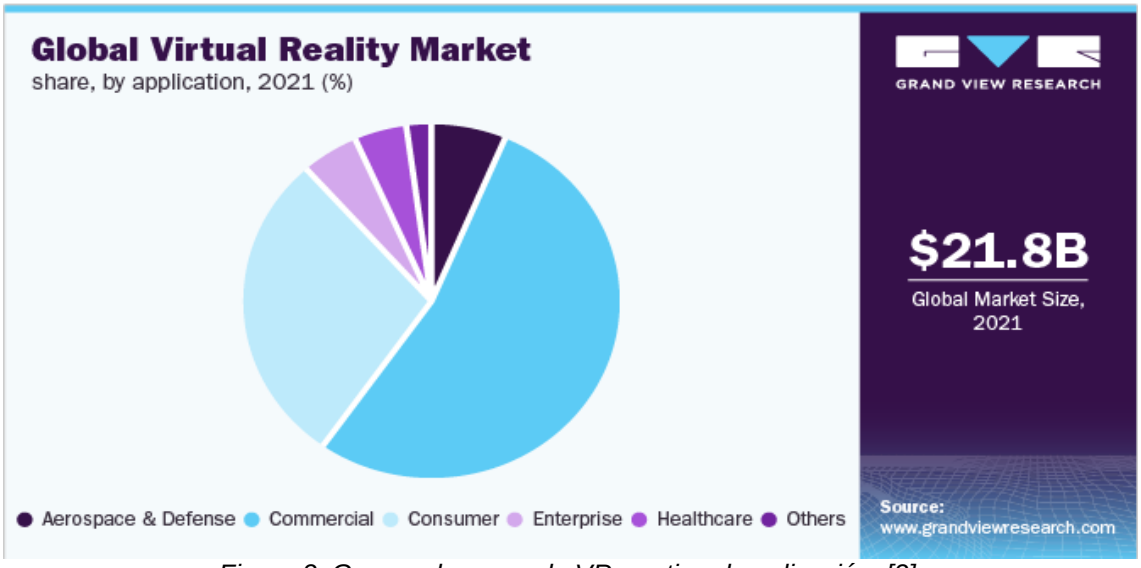

Figura 2. Grueso de mercado VR por tipo de aplicación. [2].

Otro aspecto a tener en cuenta es la importante inversión que se está haciendo en desarrollo y marketing de tecnologías relacionadas con el metaverso [3], que giran en torno al XR, siendo el posicionamiento 3D una parte clave para conseguir una buena inmersión que asegure una gran experiencia de usuario.

Actualmente el mayor exponente de metaverso implementado es VRChat que cuenta con un pico medio de alrededor de 25.000 jugadores simultáneos diarios [4]. En VRChat los usuarios pueden crear y compartir sus propios mundos, avatares, minijuegos (Figura 3) y otro tipo de contenido creado por ellos mismos, además se han realizado eventos multitudinarios como conciertos de artistas del mundo real [5] o conferencias tecnológicas [6].

Figura 3. Jugadores jugando juego de mesa en VRChat. [6].

3.3. Estado del arte

Una vez realizado el estudio de mercado, se comprende la importancia que tienen y se pronostica que tengan en el futuro las tecnologías XR y que, por lo tanto, merece la pena continuar con el desarrollo e investigación de estas. A continuación, se van a analizar algunos de los principales exponentes en cuanto a posicionamiento ya que es el componente en el cual nos centramos en este documento para entender el estado actual del arte antes de saltar a la solución desarrollada y lo que la diferencia del resto.

3.3.1. Azure Kinect

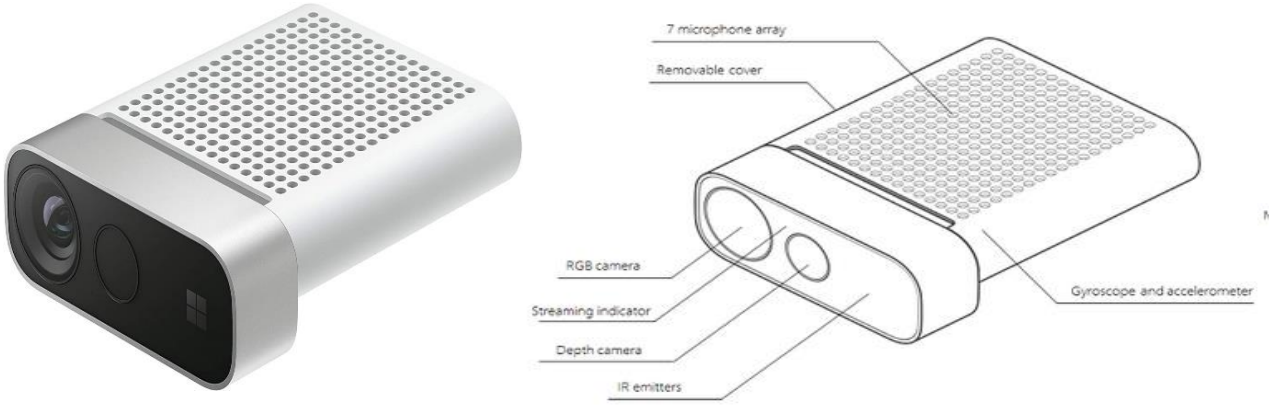

Figura 4. Azure Kinect. [7].

Mucho ha cambiado desde el Kinect original de 2010 hasta la versión actual. Como vemos en la Figura 4, su más reciente sucesor es el Azure Kinect que dispone de un kit de desarrollo a la venta desde 2020.

Según su especificación hardware oficial [7], el posicionamiento consta de una unidad inercial (giróscopo y acelerómetro), una cámara RGB, una de profundidad, y emisores de luz infrarroja.

Estos sensores trabajan en combinación para posicionar el cuerpo de una persona mediante su utilería, aunque el desarrollador puede tratar los distintos conjuntos de datos que se generan mediante sus herramientas de desarrollo.

En situaciones en las cuales el dispositivo se encuentra fijo, es decir, la unidad de medición inercial da unas lecturas prácticamente estáticas, el grueso del posicionamiento lo van a realizar las cámaras en combinación con el emisor de luz infrarroja.

La cámara de profundidad [8] utiliza el principio de tiempo de vuelo (time of flight, ToF), es decir, la luz infrarroja se proyecta sobre los objetos y vuelve a ser recogida por la cámara, calculando la distancia a cada punto midiendo el tiempo que ha transcurrido desde que se emitió la luz hasta que se recibió. Con estos datos se forma un mapa de profundidad que junto a la imagen recibida y sincronizada por la cámara RGB y modelos de inteligencia artificial en los cuales ha sido entrenado el dispositivo de fábrica, se detectan los puntos que pertenecen a un cuerpo humano y estos se unen formando un esqueleto, a partir del cual se siguen rastreando sus variaciones.

Para realizar el rastreo, es necesario que el dispositivo Kinect esté conectado a un PC que disponga de una tarjeta gráfica de Nvidia, y para que este sea lo mejor posible, es conveniente utilizar múltiples dispositivos Kinect, que disponen de utilería para sincronizarse de manera sencilla para el usuario.

Con todo esto, se obtiene un posicionamiento de todo el cuerpo, pudiendo no solamente ubicarlo en 3D si no también hacer reconocimiento de gestos y movimiento de manera nativa. En contraposición, como podemos intuir por el mecanismo utilizado, especialmente la necesidad de un PC con gráfica Nvidia dedicada y el uso de modelos de inteligencia artificial, el coste computacional no es para nada despreciable. Además, al menos sin un desarrollo adicional grande por parte del programador, el sistema se encuentra limitado a cuerpos humanos

y otros para los que se encuentren preparados los algoritmos de inteligencia artificial.

3.3.2. Oculus Rift

El exponente más conocido en el mercado doméstico. Las Oculus Rift originales utilizan el sistema de posicionamiento llamado “Constellation” y aunque esta tecnología ya no es la principal que se emplea en los modelos sucesores para el posicionamiento, ahora bajo el nombre de compañía Meta, sigue siendo una tecnología relevante ya que es la que se usa para el posicionamiento de los mandos en estos nuevos sistemas [9].

Figura 5. Oculus Rift con su sensor. [9] y [10].

Como podemos ver en la Figura 5, los puntos morados representan emisores de luz infrarroja que se encuentran escondidos dentro del visor, estos ocupan una predisposición definida, la llamada constelación [11]. Dichos emisores utilizan además unas frecuencias diferentes para ser reconocidos y todos los dispositivos que se pretendan posicionar, deben tener una de estas constelaciones. Además, son detectados mediante los sensores que podemos ver en la imagen de la derecha, que detectan la luz infrarroja y computando las variaciones de movimiento de los respectivos haces de luz en cada imagen capturada en un ordenador al cual debe ir conectado el sistema, calcula la posición.

Existen problemas graves, por ejemplo, si se le da la espalda a uno de estos sensores y no se dispone de más, o no es capaz de capturar el número suficiente de puntos para los cálculos, el sistema falla, por lo que era necesario utilizar

multitud de sensores, entorno a cuatro, simplemente para el posicionamiento 3D 360º dentro de una habitación pequeña.
Utiliza un planteamiento de sistema óptico en el cual son los marcadores los que se están moviendo y los sensores captadores los que permanecen fijos.

3.2.3. Meta Quest 2

Figura 6. Meta Quest 2.

El último producto en el mercado por parte de Meta, y con el que se ha estado trabajando físicamente en el proyecto, utiliza el sistema de posicionamiento denominado por ellos mismos como VISLAM [12], “Visual Inertial Simultaneous Localization and Mapping”.
Para entender correctamente el sistema, ya que aumenta de manera importante su complejidad con respecto a modelos anteriores de la marca, vamos a ir por partes comenzando por los mandos. Como se ha dicho, utiliza el sistema constelación explicado anteriormente, sin embargo, esto es solo parte de la solución, ya que ahora además se utilizan unidades de medición inercial (IMU) para el posicionamiento que previamente solo se usaban para la rotación. Las IMU constan de un giróscopo y un acelerómetro, que se utilizan para obtener la rotación (operaciones basadas en la física) y la posición del mando (doble integral de la aceleración), por lo que cada mando dispone de una, de esta manera se consigue la posición y rotación de los mismos con una menor latencia con respecto a sistemas ópticos, sin embargo, las IMU MEMS (sistemas microelectromecánicos) tienen problemas a la hora de realizar el cálculo de la posición, y este es el llamado deriva o “drift”.

La deriva consiste en pequeños errores de medición inherentes a la construcción de las mismas que se van propagando a lo largo del tiempo en la integración de la posición, y que, de no ser corregido de alguna manera, al cabo de pocos segundos provocará que la posición calculada esté completamente desacertada en relación con la realidad, en este problema se ahondará más en su propio apartado más adelante.

Mientras las IMU van calculando estas orientaciones y posiciones, a una frecuencia más baja, las cuatro cámaras del visor y el software de este están generando una representación en 3D del entorno de juego designado.

Figura 7. Representación 3D entorno de juego. [22].

Esto lo consigue mediante el establecimiento de puntos de interés o “landmarks”, que son patrones o puntos reconocibles del medio físico como esquinas del mobiliario, disposición de las baldosas en el suelo y muchos otros puntos de referencia para ubicarse en el entorno y localizar la zona de juego establecida desde el sistema óptico. Al mismo tiempo, las cámaras captan las constelaciones de los mandos y utilizan esta posición cuando es calculada para ir corrigiendo las posiciones dadas por los mandos mediante su sistema inercial y de esta manera contener y mitigar la deriva.

En cuanto al posicionamiento del visor, el planteamiento es similar, pero ahora utiliza estos puntos de interés de manera directa ya que, por las variaciones de esos puntos con respecto a la imagen captada por la cámara, es capaz nuevamente de calcular una posición que, en combinación con la IMU del visor, es corregida del efecto de la deriva.

En este caso, cambian la metodología con respecto a las Oculus Rift originales y son los sensores (cámaras) los que se mueven y los puntos críticos detectados los que se mantienen fijos.

Sistemas similares a estos son algunos de los más utilizados en ámbitos de conducción autónoma y robótica, aunque con componentes más sofisticados, especialmente las IMU.

El principal motivo de la inclusión de las IMU es la reducida latencia en comparación con un sistema puramente óptico. Esto es debido a que en casos de XR, la importancia de obtener baja latencia es crítica ya que incongruencias entre lo que el usuario está haciendo físicamente u ocurriendo en el entorno y lo que está viendo en el visor, ocasiona una mala experiencia de usuario, llegando a provocar síntomas de cinetosis, jaquecas etc. Para complementar y aminorar estos efectos se utilizan también patrones de predicción de movimiento, inteligencia artificial y otras técnicas de las cuales desde fuentes oficiales no se da mucho más detalle.

Con todo esto se consigue un posicionamiento y experiencia de usuario muy buena sin preparación previa del entorno (por ejemplo, instalación de marcadores), sin embargo, tiene problemas. Por un lado, se encuentra la elevada complejidad del sistema dado que hay un flujo importante de datos de varias fuentes que deben ser calculados y comunicados, también el aumento del coste computacional debido al uso de inteligencia artificial compleja, reconocimiento y detección de puntos de interés mediante algoritmos VISLAM… Todo esto es un factor importante del porqué llevan un procesador Snapdragon XR2 modificado con mejoras en inteligencia artificial y 6GB de memoria RAM, siendo que la inmensa mayoría de aplicaciones XR son bastante simplistas y no muy demandantes de requisitos.

Por otro lado, tenemos que las cámaras y el sistema de detección de puntos de interés requiere de unas condiciones de luz buenas, en las cuales ni mucha luz ni muy poca funcionan correctamente, y pasará el posicionamiento a ser solo de tres grados de libertad (orientación) si no se dan las condiciones óptimas de luz. Es decir, se limitará a la rotación ya que de esta manera no sería capaz de calcular la posición corrigiendo la deriva lo cual haría el sistema inusable. En cuanto a la rotación, también sufre de la deriva, pero es más fácilmente

corregible mediante fusión sensorial de la propia IMU con aplicación de filtros Kalman [13] u de otro tipo.

Uno de los mayores problemas a tener en cuenta fue detectado al realizar una presentación del sistema de las Meta Quest 2 a varios visitantes al laboratorio, en este caso, aparecía un mensaje de que el sistema de posicionamiento se había perdido y se cerraba la aplicación, esto parece estar debido a que había varias personas cercanas al jugador y como es natural se iban moviendo, esto hacía que los puntos captados por el mecanismo de las Quest 2, se movieran y por lo tanto, perdían su valor como puntos de referencia y el sistema se perdía. Es decir, es un sistema planteado para entornos mayormente estáticos ya que movimientos en el mismo no son bien recibidos, lo que hace que dicho sistema pueda llegar a ser inutilizable en lugares como museos o monumentos históricos con grandes afluencias de gente.

3.3.3. Ultra Wideband

La banda ultra ancha no es algo nuevo y se lleva usando en entornos militares desde la década de 1960 en su ámbito de posicionamiento radar, sin embargo, el primer uso de la banda consta del año 1897 en el transmisor “Spark-gap” inventado por Marconi [24].

Pese a su longevidad, la tecnología de posicionamiento de banda ultra ancha ha ganado mucha relevancia en los últimos años debido principalmente a dos factores.

El primero de ellos es la mejora de las implementaciones que ya llegan a unas precisiones de aproximadamente 20cm con una frecuencia de hasta 100 medidas por segundo. Por otro lado, está el abaratamiento de los dispositivos de banda ultra ancha que hace unos años eran muy caros y solo podía llegar a tener sentido en ámbitos industriales y otro tipo de entidades con amplios presupuestos, sin embargo actualmente se pueden encontrar kits de desarrollo como el MDEK1001 de Decawave por alrededor de 300 dólares, lo cual puede seguir siendo considerablemente caro para un usuario que quiera disfrutar de experiencias XR en su casa pero son precios mucho más accesibles para pequeños negocios que potencialmente podrían seguir bajando, por lo cual no debemos quitar el ojo de esta tecnología.

El posicionamiento mediante banda ultra ancha se realiza con el concepto de tiempo de vuelo similar al explicado anteriormente en el caso de Azure Kinect, sin embargo, hay varias técnicas de implementación de tiempo de vuelo que varían según el producto. Uno de los más representativos es el “Two-Way Ranging” que es por ejemplo el utilizado en los iPhone 11 para comunicarse y establecer posiciones relativas entre ellos.

En este método los dispositivos van a enviar patrones de señales con información del dispositivo al otro y mediante los tiempos que transcurren en dichas transmisiones, como podemos ver en el diagrama de la Figura 8, y el algoritmo interno de posicionamiento, se calcula la posición relativa.

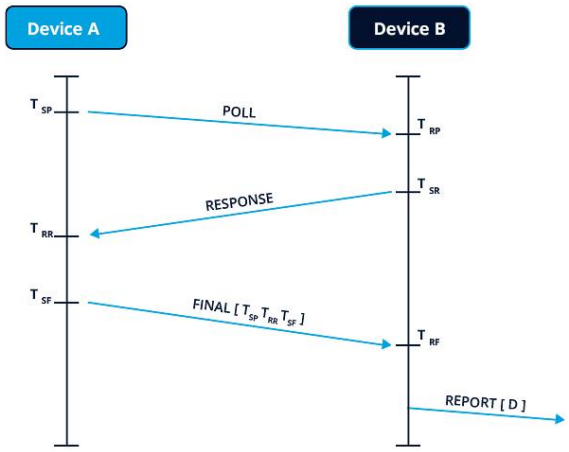

Figura 8. Two-Way Ranging. [25].

La metodología aplicada a nuestro paradigma de aplicación XR, no se realizaría entre smartphones, si no que se tendría un dispositivo que emite estas señales UWB (“UWB Tag”) en el visor XR o que el propio visor XR implementara esta capacidad para ser posicionado y estas señales fueran recibidas por un “UWB Anchor” que son dispositivos que detectan estas señales enviadas por los “Tags”, se encuentran en una posición conocida y realizan una respuesta de vuelta al tag, en ese punto y conociendo los tiempos de vuelo, el tag envía la información a un servidor de procesamiento que calcula la posición del Tag con respecto al “anchor” (o “anchors”).

Se requiere de cuatro “anchors” para el posicionamiento en una habitación de 25x25.

No hay una implementación conocida al menos de manera pública de este método de posicionamiento llevado a cabo para entornos XR en visores de las características del Meta Quest 2 por ejemplo, que requieren de una precisión prácticamente milimétrica, sin embargo, sí que hay algunas aplicaciones de realidad aumentada para smartphone que hacen uso de este sistema con buenos resultados.

El sistema no tiene rival en cuanto a precisión y latencia con respecto a métodos con características similares como es el posicionamiento mediante intensidad de señal WI-FI o métodos BLE, como podemos ver en la siguiente tabla comparativa.

	UWB	Chirp (CSS)	BLE	Wi-Fi
Location Accuracy*	10-50 cm	1-2 m	< 5 m	< 10 m
Range*	Optimal: 0-50 m Up to 200 m	Optimal: 10-500 m Up to 1000 m	Optimal: 0-25 m Up to 100 m	Optimal: 0-50 m Up to 500m
Latency*	< 1 ms to get location	< 1 ms to get location	Typically 3-5 s to get location	Typically 3-5 s to get location
Power Consumption	Low, option for embedded cell battery in select hardware options	Very low, option for embedded cell battery in select hardware options	Very low, option for embedded cell battery in select hardware options	Moderate
Cost	$$	$	$$	$$$ (Low $ with existing Wi-Fi access points)
Frequencies	3.1 – 10.6 GHz	ISM-band 2.4 GHz (2.4-2.4835)	2.4 GHz	2.4, 5 GHz
Data Rate	Up to 27 Mbps	Up to 2 Mbps	Up to 2 Mbps	Up to 1 Gbps

Figura 9. Tabla comparativa posicionamiento interiores. [26].

La latencia es completamente adecuada para un sistema de posicionamiento para entornos XR, no obstante, la precisión no llega a los extremos requeridos de unos pocos centímetros e idealmente milímetros, sin embargo, podría ser un buen sistema complementario para realizar la fusión con los sensores utilizados en el proyecto debido a que a diferencia de los sistemas basados en cámaras, no hay concepto de línea de visión y por lo tanto la orientación del dispositivo no

puede influir de manera nociva en las medidas en caso de que no pueda captar marcadores u otros dispositivos debido a sucesos relacionados con la iluminación, desenfoques u otros factores.
También, carece de deriva y el número de interferencias es muy bajo debido al uso de la banda ultra ancha para las comunicaciones.

3.4. Motivación

Como hemos visto en el análisis del estado del arte, hay múltiples sistemas de posicionamiento que se están empleando en la actualidad, sin embargo, todos ellos flaquean en algún aspecto.
De manera idónea lo que se está buscando es un sistema de posicionamiento que sea capaz de obtener una precisión lo suficientemente elevada para que el usuario sienta que efectivamente se está realizando un movimiento sinónimo o lógico en el entorno XR respecto al movimiento real. Además, se requiere que la latencia de dicho movimiento sea lo suficientemente baja para que este se realice de forma natural y fluida y produzca de esta manera la ilusión de inmersión propia de este tipo de aplicaciones. Por último y no menos importante, se busca que el sistema que cuente con estos dos factores sea lo más sencillo computacionalmente posible, obteniendo de esta manera unos consumos energéticos más eficientes que puedan aumentar la autonomía del dispositivo y conseguir al mismo tiempo un ahorro económico.

4. Proyecto

4.1. Hardware utilizado

El hardware utilizado en el proyecto desarrollado es:

- IMU: MPU-9255 9dof (acelerómetro, giróscopo y magnetómetro) de InvenSense.
- Cámara: HBV-1323 S1.0 de Huiber Vision.

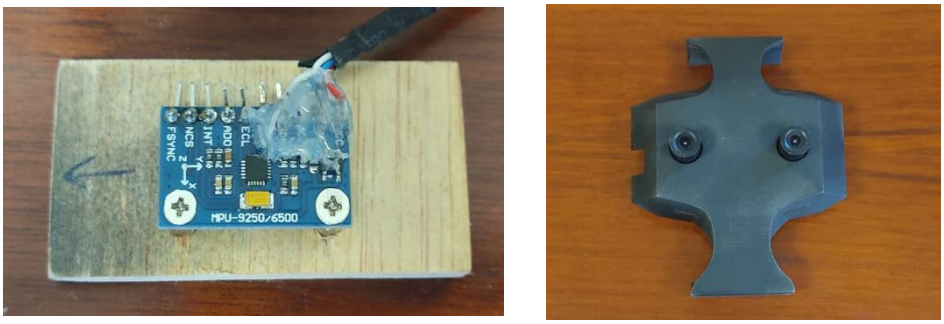

Figura 10. IMU y cámara.

- Visor XR: Meta Quest 2
- Sistema de cómputo: Raspberry Pi 4.
- Marcadores led: Parte de la patente gestada en la Universidad de Cantabria [15].

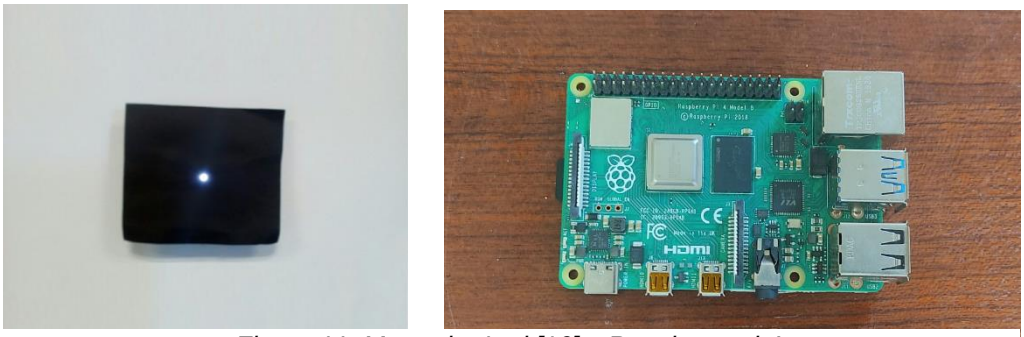

Figura 11. Marcador Led [16] y Raspberry pi 4.

- Router Linksys WRT54GL para la creación de la red local necesaria para la comunicación de la Raspberry con el destino (visor).

4.2. Estado inicial del proyecto

El proyecto de conseguir un sistema de posicionamiento con las características expuestas en la motivación aportando de esta manera al estado del arte, lleva varios años en progreso por distintos alumnos de la universidad que se ha ido desarrollando mediante sus trabajos de fin de grado. El punto de origen de este trabajo es el dejado por el antiguo alumno Raúl Gómez Varela [16] que sigue la metodología de la patente [15].

Se parte de un sistema óptico que utiliza la cámara previamente expuesta y los sensores de las Oculus Rift Dk1 que proporcionan la orientación. Mediante triangulación del visor con respecto a los marcadores leds, se consigue también la posición 3D y se imprime por pantalla el valor en milímetros para cada eje.

Este sistema tiene varios problemas, los dos principales son la elevada latencia y aún más grave, los errores de medición, que son de varios centímetros, algo que, si bien era suficiente como prueba de concepto, en este momento para la intención de implementarlo para entornos XR es inviable. Además, requería de movimientos muy suaves para que el sistema no se perdiera y dejara de funcionar. Otro gran problema es que, debido a que el sistema utiliza los sensores de las Oculus Rift Dk1, este se encontraba necesariamente ligado a las mismas y esta plataforma está ya obsoleta y descatalogada convirtiendo el consecuente desarrollo para la misma complicado y carente de sentido.

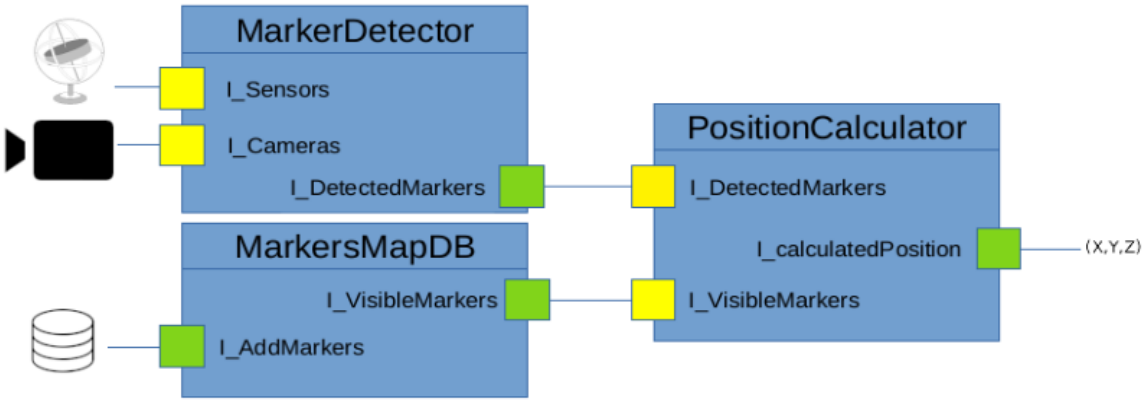

Figura 12. Diagrama del sistema óptico. [9].

Como vemos en Figura 12, los módulos que componen el sistema óptico son "MarkerDetector", "MarkersMapDB" y "PositionCalculator".

A pesar de no entrar en mucho detalle debido a que se encuentra la descripción completa de los mismos en la referencia [9], es conveniente una pequeña descripción de los módulos para entender en realidad cómo funciona el sistema del que se parte.

"MarkerDetector" es el primer módulo que entra en acción, recoge la imagen de la cámara y las lecturas de orientación de los sensores de las Oculus Rift Dk1 y calcula las coordenadas de los potenciales marcadores led que reconoce a partir de estos datos. En este momento, comienza una doble funcionalidad soportada por el módulo "MarkersMapDB" que reconoce cuales de esos potenciales marcadores leds lo son realmente y los identifica emparejándolos con los de la base de datos previamente cargada con las ubicaciones fijas de los marcadores leds. El módulo "PositionCalculator", sabiendo los leds que se están viendo y desde qué orientación, mediante operaciones trigonométricas, calcula la posición de la cámara con respecto a un punto de origen fijo, la llamada triangulación.

Se requiere que la cámara capte al menos dos marcadores para que la triangulación pueda ser computada.

4.3. La Deriva

Como se ha dejado entrever en el estado del arte, la deriva es el problema principal de las IMU por el cual no pueden ser el único componente para determinar la posición de manera precisa en intervalos largos de tiempo, lo cual

es frustrante teniendo en cuenta lo baratas que son de construir en su categoría de consumo, teniendo el modelo utilizado en el proyecto un precio inferior a un euro.

La deriva no es causada por una sola causa [14], sino que hay varias fuentes de error que son inherentes a la construcción de la IMU. Por un lado, está el sesgo de corriente continua, causado por defectos en la construcción de la IMU, muy fáciles de ocurrir debido a la miniaturización de las MEMS IMU que provocan variaciones que, por ejemplo, cuando el resultado debería estar dando cero, produzca un resultado cercano a este, pero no cero absoluto.

Por otro lado, tenemos el ruido que puede provenir de varias situaciones como la temperatura, fuerzas externas y los propios defectos electrónicos previamente mencionados que pueden producir “random walks” que dice que el siguiente dato puede ser determinado por la casualidad en un rango acotado. Por ejemplo, este puede ser entre -0.2 y 0.2, siendo la magnitud del dato real 0.0.

Además, la toma de mediciones de la IMU que no puede realizarse de manera continua en el tiempo también arrastra error con respecto al caso teórico debido a que se pierden valores por el “polling rate”.

Los errores derivados de la integración de todos estos valores con sus respectivas taras para calcular la posición en el tiempo es lo que se conoce como la deriva, que acumula dichos errores aumentando de manera exponencial en el tiempo.

Estos errores de medición pueden ser mitigados en IMUs de mayor calidad de construcción y materiales y, por ende, de mayor coste, aunque, aun así, no pueden ser totalmente suprimidos.

4.4. Desarrollo

Cada sistema tiene sus propias ventajas y desventajas, por lo que a menudo se fusionan diferentes sistemas para tapar los problemas de los otros y potenciar los puntos fuertes de cada uno.

En este caso se fusiona el sistema óptico inicial con las modificaciones y mejoras que se abordarán en el siguiente apartado, con un nuevo sistema inercial.

El sistema óptico tiene la gran ventaja de que no sufre de deriva y, por lo tanto, su precisión no se ve influenciada por el tiempo transcurrido, sin embargo, la latencia es significativamente más elevada que la del sistema inercial.

Por otro lado, el sistema inercial tiene una latencia muy baja pudiendo conseguir un magnífico tiempo de respuesta para posicionarse en referencia al movimiento del mundo físico de manera prácticamente de tiempo real para el usuario, sin embargo, este sistema tiene deriva. Este puede funcionar sin necesidad de estar captando marcadores a diferencia del sistema óptico.

Estos sistemas se complementan a la perfección, siendo el sistema inercial el "principal" responsable del posicionamiento ya que es capaz de dar un mayor número de posiciones en menos tiempo y utilizando el sistema óptico como "complementario" en la fusión y siendo su función corregir la deriva del sistema óptico en movimientos largos.

4.4.1. Sistema óptico

Para renovar el proyecto y hacer que se pueda seguir trabajando en el mismo correctamente, fue necesario independizar el sistema óptico de las Oculus Rift Dk1 y hacer que posicione únicamente con la cámara, los marcadores leds y la nueva IMU, prescindiendo de los sensores de las gafas y haciendo de esta manera que el sistema óptico [9] pueda utilizarse para cualquier tipo de visor y otros sistemas.

Esta tarea fue facilitada debido a la naturaleza modular del software del que se dispone, escrito en C++ y utilizando programación basada en objetos a los cuales se accede mediante sus interfaces. Por lo tanto, en este caso la interfaz "I_Sensors", que es la que nos interesa, en principio queda intacta (aunque se han hecho varias modificaciones para trabajar de mejor manera y añadido funcionalidad para facilitar el depurado). Consecuentemente, lo que debe ser modificado es la implementación del módulo "Sensors", prescindiendo del kit de desarrollo del visor Rift y consiguiendo la orientación con nuestra nueva IMU. Sin embargo, esto conlleva problemas, ya que la propia API de las Rift originalmente proporcionaba la orientación (alabeo, cabeceo y guiñada o "roll", "pitch" y "yaw") directamente con la respectiva fusión de sensores (giróscopo y acelerómetro) para reducir la deriva al máximo, en esta ocasión para la nueva IMU, era necesario hacer algo similar. Para ello, se utilizó una librería llamada "RTIMULib2" modificada para nuestro modelo de IMU [17] que permite conseguir la orientación con la respectiva fusión sensorial mediante filtro Kalman [13] (configurable) que utiliza los valores dados por el acelerómetro, giróscopo y

magnetómetro para mitigar la deriva y conseguir una buena orientación fusionada de manera estable. Otra parte crucial para el correcto funcionamiento de la IMU es una calibración óptima, hasta el punto de que antes de realizarla, los valores obtenidos de la orientación parecían prácticamente aleatorios, rotando de manera descontrolada. Para la calibración, se utilizó la utilería que proporciona la librería para conseguir de manera más cómoda los valores mínimos y máximos ya que el método de calibración min/max fue el elegido y utilizar de esta manera los desplazamientos adecuados para los distintos componentes de la IMU. Finalmente se comprobó el correcto funcionamiento mediante una visualización 3D en tiempo real de la librería como se aprecia en la Figura 13.

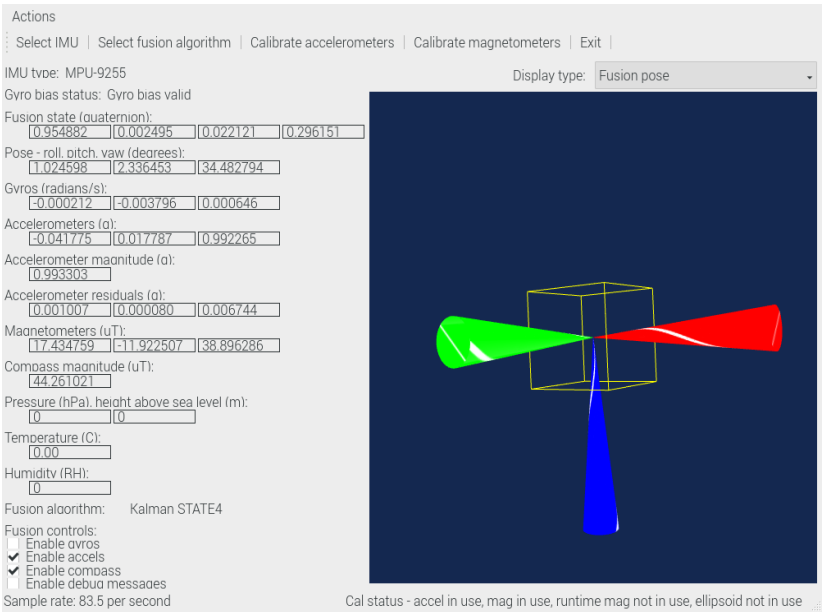

Figura 13. Representación 3D orientación.

Es importante cuadrar la nomenclatura de ejes, el rango y el sentido de movimiento para cada eje en todos los módulos que componen el sistema.

En este punto ya tenemos el sistema óptico de los compañeros funcionando de manera independiente con respecto a las Rift y con el efecto colateral de la utilización de una nueva IMU con su correcta calibración personalizada que consiguió reducir la variación de resultados del sistema original y aumentar la precisión.

4.4.2. Sistema inercial

Este es el grueso de la aportación del trabajo. De manera similar a algunos de los sistemas del estado del arte, se calcula además de la orientación, la posición

de manera puramente inercial integrando las lecturas del acelerómetro que se recogen mediante el bus I2C. Sin embargo, en este caso, se utiliza una técnica que es capaz de mitigar y reiniciar la deriva sin soporte del sistema óptico consiguiendo una posición precisa de manera puramente inercial durante periodos de tiempo muy grandes siempre y cuando dicha técnica se aplique en lapsos suficientemente cortos de tiempo.

Se ha escrito e integrado en el sistema anterior un nuevo módulo llamado "InertialPosition" con su respectivo fichero de cabeceras e interfaz en C++ con programación basada en objetos. En él, se recogen los datos necesarios de la IMU que son aceleración y orientación (cabeceo, alabeo, guiñada), y se calcula la posición inercial.

El cálculo de la posición a partir de la aceleración se realiza mediante una doble integración. De la aceleración en el tiempo se integra la velocidad, y de la velocidad en el tiempo se integra el desplazamiento, que, tras ser añadido al desplazamiento anterior con respecto a un punto de origen conocido, se convierte en posición.

$$\boldsymbol{velocidad_x = velocidadAnterior_x + \left((fixedAccel_x * \pi) * time_{elapsed}\right)}$$

$$\mathbf{posicion_x = posicionAnterior_x + \left(velocidadAnterior_x * time_{elapsed}\right) + (0.5 * fixedAccel_x * gravedad * time^2_{elapsed})}$$

Como podemos ver en las ecuaciones, La posición depende de la posición anterior, la velocidad anterior, el tiempo transcurrido con respecto a la anterior medición y la "fixedAccel". A su vez, la velocidad actual depende de la velocidad anterior, del tiempo transcurrido y de esta "fixedAccel" que se explica a continuación. Pero antes, destacar que debido a que los cálculos no son independientes en cada iteración y dependen de la anterior, producen el efecto de la deriva del que se ha hablado, en el cual cualquier fuente de error integrada en el tiempo aumenta la imprecisión de manera rápida.

Por otro lado, como podemos ver ese "fixedAccel", no es simplemente la aceleración medida por la IMU, sino que esta tiene un problema importante para nuestros cálculos y es que contiene la gravedad. Esto no ocasionaría excesivos quebraderos de cabeza si la IMU fuera a estar fija con respecto a la orientación, de manera que la gravedad quedaría entera en un solo eje de manera constante, pero para el proyecto, esa IMU debe ser capaz de rotar en todos sus ejes, como

lo haría una cabeza humana hasta cierto punto. Esto lo que provoca es que si por ejemplo, está la IMU 60º rotada en su roll o alabeo, la gravedad quedaría distribuida entre el eje vertical y el perpendicular horizontal locales de la IMU, y no sabemos a priori en que proporción para aislarlo correctamente de la aceleración causada por fuerzas externas. La solución, en este caso, es rotar el vector gravedad {0,0,-1} con respecto a la orientación calculada, siendo -1 el valor del eje vertical que apunta a la superficie terrestre, de manera que se siga manteniendo en este eje local de la IMU toda la componente gravedad y no nos altere los resultados, para ello, se utilizan las matrices de rotación [18].

$$R(\alpha,\beta,\gamma) = R_z(\alpha)\,R_y(\beta)\,R_x(\gamma) = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}.$$

Figura 14. Matriz de rotación. [18].

Tras crearla siguiendo la convención YXZ, ya que el orden importa a la hora de crear la matriz de rotación porque ocasiona giros diferentes debido a que no tiene la propiedad conmutativa al multiplicarla por el vector gravedad, obtenemos el vector gravedad rotado que ya se lo podemos restar a la aceleración obtenida de la IMU y conseguir esa “fixedAccel” individualizada de la gravedad para poder hacer los cálculos.

En este punto quedan aún tres problemas principales por solucionar. El primero es que debido a que la IMU se trata de un dispositivo real y físico, es por ende, imperfecto y tiene ruidos y perturbaciones, por lo que en estado de reposo las mediciones de la aceleración no tienen por qué ser cero lo cual es problemático ya que haría que la posición “se moviera” cuando a efectos prácticos, lo ideal sería que se mantuviera en cero absoluto. Para solventarlo, se establecen unos límites dentro de los cuales todo valor en ese rango pasa a ser cero absoluto, esto es una forma de filtro bajo, el elegido para esta IMU y aplicación en concreto es de 0.05 y de -0.05 en cada eje.

Otro problema es que, de nuevo, causado por el ruido, que el dispositivo es imperfecto y que el tiempo de recogida de datos no es completamente continuo (“polling rate”), ni la IMU tiene capacidad para que lo fuera, la aceleración al

realizar un movimiento y pasar a parado de nuevo no se anula siempre como ocurre en el caso teórico. Esto lo que ocasiona es que, tras realizar un movimiento, este puede sobre extenderse o peor aún, volver hacia atrás debido a un mayor número de aceleración en sentido opuesto al del movimiento original, ya que como sabemos la “frenada” no es más que aceleración en el sentido opuesto. Se ha realizado un pequeño programa realizado en Python [19], el cual abre un fichero previamente creado y recoge resultados de las aceleraciones durante un movimiento en el sentido positivo de un eje utilizando el sistema descrito en este documento, calcula la suma de todos los resultados y genera una gráfica representando el movimiento mediante las aceleraciones.

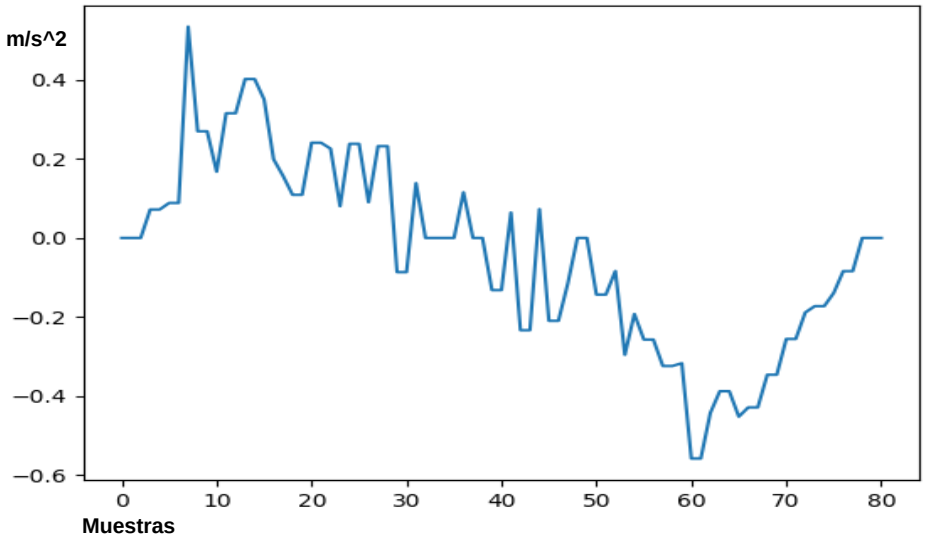

Figura 15. Aceleraciones de un movimiento. [19].

Como vemos, la Figura 15 es la resultante de la ejecución, la cual representa aceleración a lo largo del tiempo (muestras tomadas), el resultado de la adición de las aceleraciones es de, -0.041, lo que ocasionaría si no se hace ningún tratamiento, que el movimiento efectuara un efecto rebote y volviera ligeramente hacia la izquierda.

Para solucionarlo, se utiliza una política de sentidos en los ejes, de manera que, si un movimiento se inicia en un eje, no se puede mover en sentido opuesto mientras dure ese movimiento, el final del movimiento es cuando el dispositivo se para, es decir, entra en el rango definido cercano al cero durante un periodo de tiempo determinado y consecutivo.

Finalmente, llegamos a la problemática de la deriva, que como sabemos, crece a lo largo del tiempo sin límites. Sin embargo, partiendo de la premisa de que en entornos XR la mayoría del tiempo los movimientos son cortos, pasos, o pequeños desplazamientos debido a la naturaleza de las propias aplicaciones XR, que suelen estar delimitados por zonas de movimiento más o menos pequeñas dependientes del espacio físico de juego del que se dispone y, debido a que el usuario no suele hacer con confianza grandes desplazamientos, se propone que cada vez que la aceleración sea cercana a cero durante un periodo de tiempo determinado, se interprete que el usuario se ha parado, y como está inmóvil, se reinicia la velocidad y la aceleración, poniéndolas a cero. De esta manera eliminando la deriva acumulada, aunque esta existe durante el transcurso del movimiento, no es suficiente para que sea notable a ojos del usuario, al menos en aplicaciones que no requieran de precisión extrema. De esta sencilla y nulamente costosa manera podemos hacer que el sistema proporcione una posición con bastante grado de precisión de manera puramente inercial sin utilizar el sistema óptico. Todo esto en sí ya sería suficiente para un gran número de aplicaciones, como se demostrará en la parte de la implementación de entornos XR en Unity, sin embargo, ya que también se dispone del sistema óptico, este se puede utilizar para corregir situaciones en las cuales el movimiento se ha prolongado en el tiempo de manera que la deriva ha sido suficiente como para desviar al usuario de manera notable en su trayectoria y, si la posición dada por la IMU y por el sistema óptico es lo suficientemente distante entre ellas, se dispone de ambas para poder, o bien hacer una multiplexación, eligiendo la dada por el sistema óptico y reiniciar el sistema inercial, o bien hacer algún tipo de algoritmo de fusión entre ambas medidas, llegando a un punto “intermedio” y decidir si reiniciar el sistema o no y seguir fusionando. Por ejemplo, se podría dar un algoritmo de fusión complementario simple en el cual se le dieran pesos de 0.6 a la IMU y de 0.4 a la posición dada por el sistema óptico para la corrección. Estos valores y el algoritmo en sí son fácilmente modificables en el código según requiera la aplicación o el programador, aunque evidentemente se pueden utilizar mecanismos más complejos como filtros de Madgwick para conseguir mayor precisión a costa de mayor complejidad.

Como podemos ver, el tema del posicionamiento en este tipo de entornos XR es complicado y sigue siendo principal fuente de investigación de las compañías involucradas en el desarrollo de estas tecnologías. A pesar de ello, no se ha llegado a una solución totalmente efectiva de manera que muchos desarrolladores de experiencias XR optan por poner un sistema de teletransporte mediante los mandos para el movimiento en lugar del sistema de posicionamiento natural mediante el movimiento del jugador.

4.4.3. Sistema de comunicación

Para que el proyecto tenga sentido, los resultados deben ser útiles, para ello, lo ideal sería tener un sistema en el cual el cálculo de la posición y la utilización de la misma estuvieran integrados, por ejemplo, un visor XR propio con el sistema de posicionamiento corriendo en su procesador paralelamente a la aplicación gráfica XR. Sin embargo, por motivos de tiempo y de alcance del proyecto, al menos como parte del trabajo de fin de grado, se ha decidido optar por una comunicación basada en sockets TCP mediante la cual se envía desde la Raspberry Pi (que corre el sistema de posicionamiento) toda la información necesaria al programa Unity que recibe los datos; estos son la orientación, la posición dada por la IMU, la posición dada por el sistema óptico en caso de que se utilice, y un booleano indicando si la información del sistema óptico es válida, ya que esta tiene una mayor latencia que el sistema inercial y, por lo tanto, no siempre que se envíen datos de posicionamiento a la aplicación va a haber nuevos datos preparados por parte del sistema óptico.

Para poder realizar la comunicación se ha configurado una red local mediante un router sin necesidad de conexión a internet. En esta red local se establecen direcciones IP fijas para los dispositivos y se logra facilitar de esta manera la conexión, ya que no habrá necesidad de introducir las nuevas IP como ocurriría en el caso de mantenerlas de manera dinámica.

Para realizar el paso de información mediante sockets, se ha realizado la correspondiente serialización en la fuente y deserialización en el destino. La fuente es cualquier dispositivo capaz de correr programas en lenguaje C/C++, con bus I2C para la recogida de datos de la IMU y al menos un puerto USB para la cámara. El destino puede ser cualquier dispositivo capaz de ejecutar programas Unity (.apk).

Los destinos utilizados para las pruebas han sido un equipo portátil y el propio visor Meta Quest 2. El origen ha sido en todo caso la Raspberry pi.

4.4.4. Fusión

El foco principal del desarrollo se encuentra en el sistema inercial, su funcionamiento y su capacidad de auto reinicio en desplazamientos cortos para mitigar la deriva. No obstante, se ha implementado una funcionalidad de fusión básica entre el sistema óptico y el sistema inercial de manera que el sistema envía por el socket toda la información necesaria de ambos subsistemas a la aplicación Unity, como se comenta en 4.4.3.

La fusión se realiza directamente en la aplicación Unity para dotar de una mayor libertad al programador de experiencias en cómo quiere usar ambos resultados, o incluso hacer cosas diferentes con ellos.

Es necesaria esta fusión para el caso de movimientos largos en el cual la deriva podría llegar a ser lo suficientemente grande como para hacer que la medición no sea utilizable debido al error acumulado. Sin embargo, no se ha podido probar extensamente este caso debido a las limitaciones con los cables y como se encuentra el sistema físicamente conectado actualmente, por lo que se considera trabajo futuro, sin embargo, se verá un ejemplo de uso en el apartado de resultados.

La fusión de manera demostrativa para poder observar de manera gráfica y clara en la aplicación la influencia de ambos subsistemas y no con carácter definitivo, lo que hace es tomar en cuenta los desplazamientos dados por el sistema inercial, recordemos, con una mayor frecuencia que el sistema óptico, y cuando hay una medición preparada del sistema óptico (carente de deriva), se utiliza esta en lugar de la inercial. A partir de este punto, las siguientes inerciales reiniciaran sus aceleraciones y velocidad acumuladas, eliminando la deriva, y darán su desplazamiento en relación con esta última posición óptica calculada de manera que no se produzcan grandes saltos y pueda seguirse utilizando el posicionamiento durante un periodo teóricamente infinito de tiempo.

4.4.5. Implementación Unity

El siguiente punto importante del proyecto es proporcionar una aplicación XR para poner a prueba el sistema de posicionamiento en un caso de uso real, al

margen de una simple muestra de los datos de posicionamiento por pantalla, como se llevaba haciendo en iteraciones anteriores del proyecto.

Para ello, se ha implementado un entorno VR funcional en Unity y se ha probado ejecutando tanto en un equipo portátil como en el visor Meta Quest 2 como destino desactivando el sistema de posicionamiento interno propio del visor y utilizando solamente el desarrollado en el proyecto. En el programa Unity se ha implementado el lado receptor de los sockets para recibir y deserializar la información calculada por el sistema, y se ha desarrollado un “prefab” [20] como son llamados en Unity en C#, que podrían ser equivalentes a librerías en un lenguaje de programación tradicional, que se encuentran vinculadas a un objeto Unity dotándolo de funcionalidad de manera que cualquier desarrollador que quiera hacer sus propios proyectos con el sistema de posicionamiento, solo tendría que importar el “prefab”, que en este caso está vinculado a la cámara del jugador, para poder desarrollar de manera sencilla. Dicho “prefab” se puede adaptar de manera fácil para funcionar también para objetos, de manera que no solo sirva para desplazarte como jugador, sino que también se le pueda poner la IMU y/o cámaras a cualquier objeto del mundo real y su movimiento se registraría en el entorno XR sin problemas, en general, las posibilidades son muy amplias, pudiendo traer cualquier objeto del mundo real a entornos XR de manera sencilla.

La manera en la que funciona es que, al iniciar la aplicación, esta se conecta al dispositivo fuente mediante sockets, a partir de este punto, en cada marco de la imagen del entorno XR se pide y recibe un mensaje con los resultados, estos son deserializados, tratados, y sus valores son utilizados para cambiar las variables requeridas en Unity para ajustar el movimiento y la orientación. Esta tasa de imágenes por segundo es variable según el dispositivo destino en el que se ejecute la aplicación Unity, pero suele oscilar entre los 60 y los 90 por segundo ya que son entornos con una baja carga de elementos y baratos de cargar de forma general. Tras cerrar cualquiera de los dos programas, el origen o el destino, se cierran los sockets de ambos y paran de ejecutar. El cálculo de la orientación a veces puede variar en el orden de 1º o 2º debido a pequeñas perturbaciones en el medio del mundo físico que disturben los sensores de la IMU, para lo cual se añade un pequeño filtro en la implementación Unity que

evita este tambaleo. Por otro lado, la posición calculada por el sistema siempre es en el ámbito local de la propia IMU, por lo que para que corresponda al mundo virtual, hay que realizar una nueva rotación siguiendo el método descrito anteriormente mediante matrices de rotación, sin embargo, esta vez se aplica directamente en la posición y no en la aceleración. Esto probablemente es necesario en la mayoría de los casos. Por ejemplo, si no se hace esta rotación y movemos la IMU hacia adelante en el mundo físico, esta se moverá correctamente en el mundo virtual, pero si rotamos la IMU 90º hacia la derecha en su guiñada ("yaw") y volvemos a mover hacia "delante" de la IMU que ahora es hacia la derecha en la perspectiva del observador y debería ser a la derecha en el mundo virtual (al frente en la perspectiva de la cámara), vemos que el movimiento seguiría siendo erróneamente en la misma dirección que el primer movimiento, por ello la importancia de esta segunda rotación.

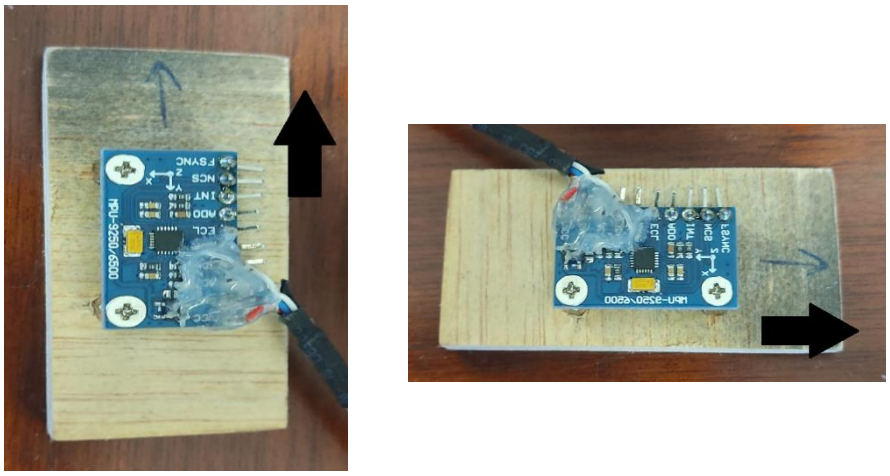

Figura 16. Ejemplificación movimiento transformación dimensión.

Se puede ver de manera visual en la Figura 16 que correspondería al mismo movimiento en el mundo virtual en caso de no ser corregido mediante la rotación correspondiente debido como se ha dicho con anterioridad a que el cálculo de la posición se realiza en el sistema de referencia local.

En la Figura 17 podemos ver el entorno Unity básico creado con 3 cubos simulando cajas, un plano actuando como suelo e iluminación direccional simple.

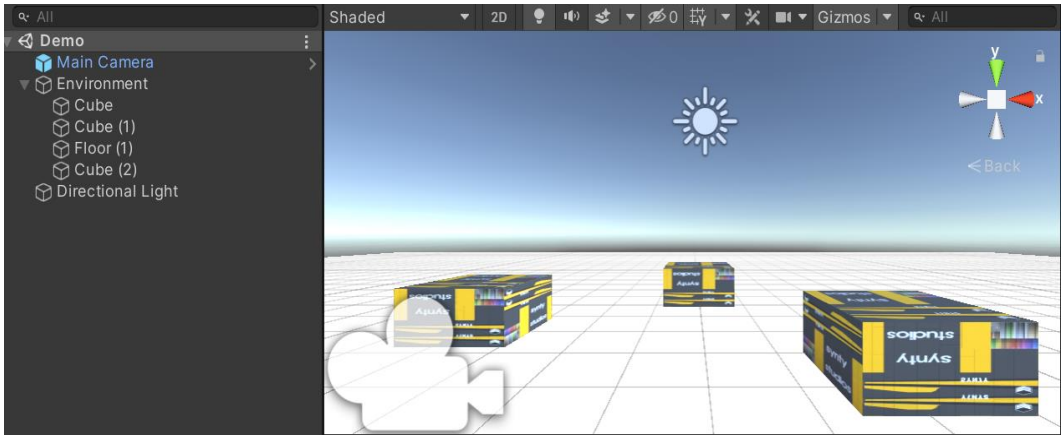

Figura 17. Entorno básico Unity.

Con el sistema desarrollado se puede rotar y mover la cámara del jugador en este entorno (videos demostrativos en la defensa).

4.5. Metodología de modelado

En el propio departamento de TEISA de la Universidad de Cantabria se ha desarrollado una metodología de modelado y diseño muy completa llamada S3D [21] que tiene como pilares la simplicidad, escalabilidad, separación de problemas, reusabilidad y flexibilidad. Si bien no se ha sido fiel a nivel metodológico a todo lo que establece el método S3D, se han intentado mantener estos pilares en el desarrollo del sistema. Algunas de las medidas tomadas para ello son la separación de funcionalidades del sistema en módulos, lo que facilita la escalabilidad, fomenta la simplicidad, la separación de problemas y la reusabilidad y flexibilidad del sistema, el uso de interfaces para acceder a dichos módulos (lo que facilita la comunicación entre los diversos módulos) y la facilidad para cambiar las implementaciones internas de los mismos etc.

Adicionalmente, desde el departamento se ha desarrollado un plugin para Eclipse que, a partir de código desarrollado siguiendo las directrices del método para la programación y un conjunto de diagramas UML, es capaz de generar el ejecutable o ejecutables del sistema, correr el programa y obtener una serie de informaciones acerca de la ejecución y el sistema en su conjunto de gran utilidad, además de configuraciones para el mismo. Estas directrices para amoldar el sistema al plugin son las que se han seguido de manera más estricta.

La utilización del plugin trae consigo un número importante de ventajas que fomentan los pilares expuestos anteriormente. Entre ellas se encuentra la posibilidad de ejecutar múltiples instancias del programa o de los módulos que lo componen de manera distribuida cómodamente, cambiar fácilmente las implementaciones de los módulos para una misma interfaz, obtener modelos de rendimiento y carga de trabajo en integración con el sistema VIPPE también desarrollado en la Universidad de Cantabria [23] etc.

Por lo tanto, y para facilitar la integración completa en el futuro del sistema desarrollado con todas las capacidades del plugin, se han realizado las principales configuraciones y diagramas del sistema.

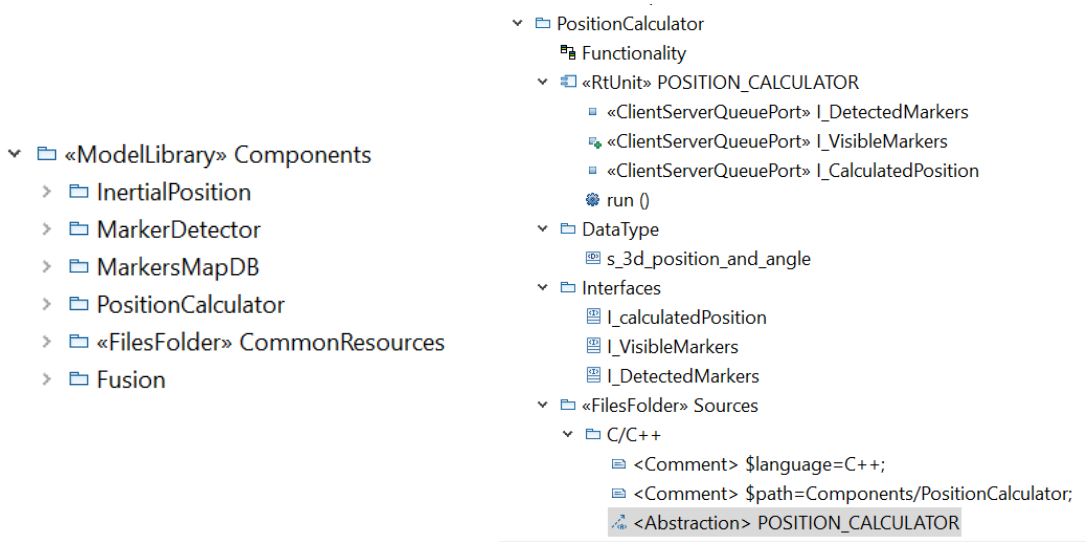

Figura 18. Librería de componentes UML.

Inicialmente, se ha desarrollado la librería de componentes para el modelado como vemos en Figura 18, en la parte izquierda se puede apreciar la estructura de la librería de componentes creada, donde cada componente es un módulo funcional del sistema. En la parte derecha podemos ver la estructura de cada uno de los componentes. Se establece una “RealTimeUnit” (RtUnit) como estereotipo para el componente principal del módulo el cual contiene su función principal, el run en este caso, y sus puertos de entrada y salida de información nombrados con el nombre de las interfaces que se utilizan para ello. Por otro lado, en “DataType” se establecen los tipos de datos creados en el módulo que se salen del estándar del lenguaje (int, double…). En “Interfaces” se colocan las interfaces que intervienen en el módulo, bien sean requeridas o provistas. Además, en el paquete “Sources” con estereotipo “FilesFolder” se establecen en

forma de comentarios, el lenguaje en el que está escrito el módulo y la ruta hacia el código. Por último, se establece la relación entre el componente de tiempo real y el código, como podemos ver en Figura 19.

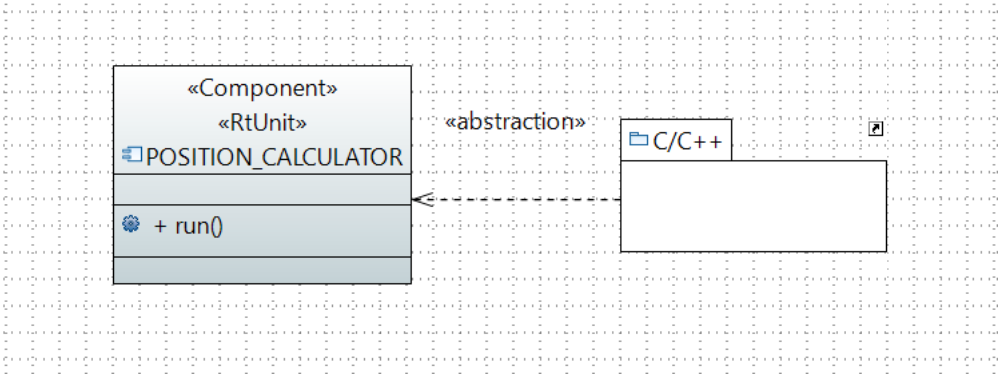

Figura 19. Relación componente-código.

Una vez esta creada la librería de componentes, se abre la posibilidad de realizar los diagramas que ayudan a la concepción y desarrollo del sistema, probablemente el de mayor relevancia es el diagrama de aplicación, que muestra los distintos componentes y como se relacionan entre sí.

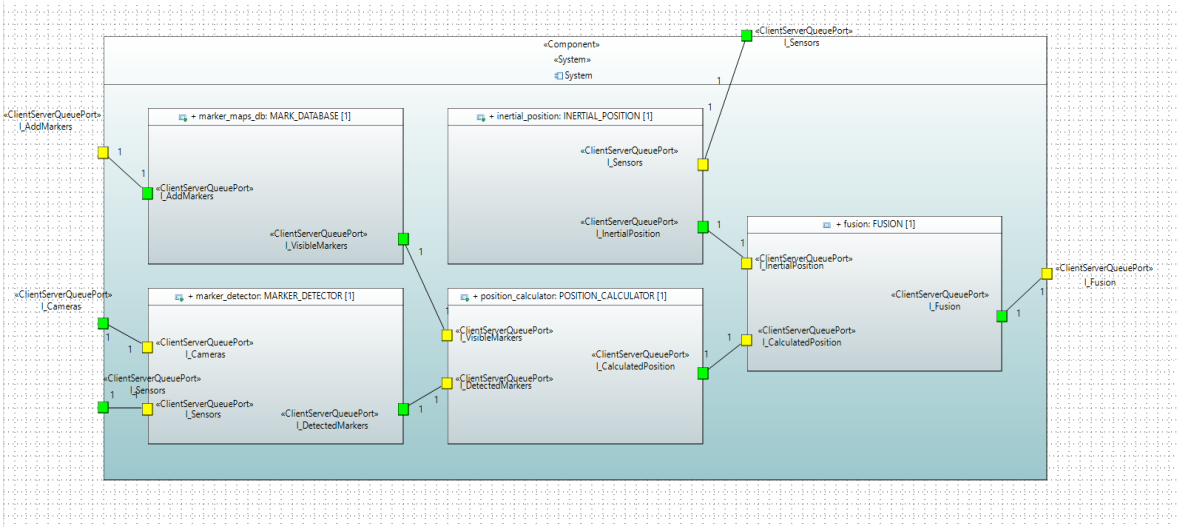

Figura 20. Diagrama de aplicación.

Como se percibe, con respecto al sistema óptico del cual parte la implementación, se ha añadido el módulo “inertial_position” que recibe los datos de la IMU mediante la interfaz “I_Sensors” y calcula la posición inercial de la manera descrita anteriormente. El otro módulo nuevo es “fusion”, el cual recoge la posición inercial y óptica a través de las interfaces “I_InertialPosition” y

“I_CalculatedPosition” respectivamente. Estas posiciones son serializadas y enviadas al programa Unity mediante sockets TCP (“I_Fusion”).

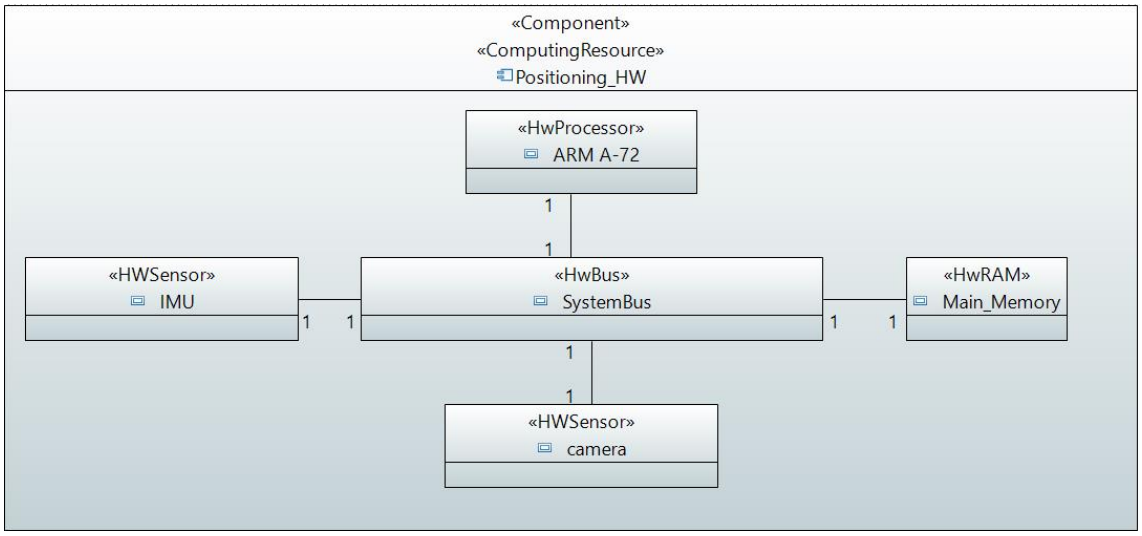

Figura 21. Vista hardware.

En la Figura 21 se encuentra el diagrama hardware del sistema de posicionamiento, que se ejecuta en el ARM A-72 de la Raspberry Pi (se puede cambiar) y se comunica con la IMU, la cámara y la memoria principal a través del bus del sistema.

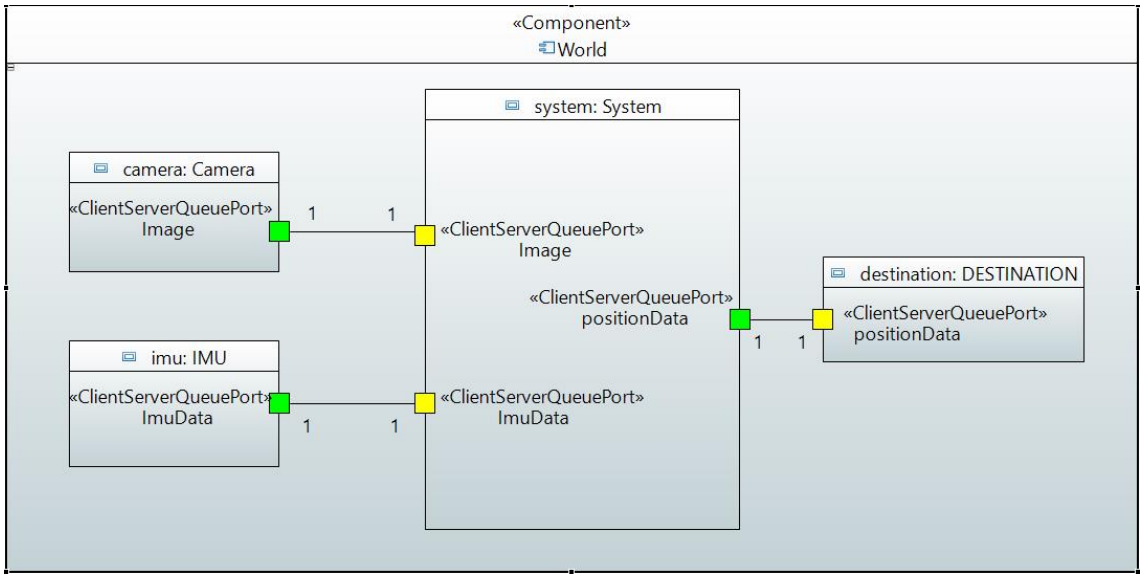

Figura 22. Vista del mundo.

En la vista de la Figura 22 se encuentra la perspectiva del mundo en la cual se concretan las entradas y salidas principales del sistema que se veían en la vista de la aplicación, se recogen los datos de las imágenes de la cámara y de la IMU y se envían los resultados finales de posición a otro sistema destino.

5.Resultados

Uno de los logros es el hardware en el que se ejecuta el sistema de posicionamiento, que es muy modesto en comparación con los sistemas del estado del arte, ejecutándose directamente en el microprocesador de propósito general de la Raspberry Pi 4, un ARM Cortex A72 que carece de optimizaciones adicionales para XR como es el caso de Snapdragon XR2 o la tarjeta gráfica Nvidia requerida en el caso de Azure Kinect. Además, según el comando top, en ejecución solo se utiliza un 0.7% de los 4GB de memoria RAM de los que se dispone, de tal manera, que podría reducirse significativamente el hardware en el que corre.

Por otro lado, problemas derivados del sistema óptico utilizados especialmente en el mundo de la robótica, son inexistentes, ya que la mayor parte de la ejecución se realiza con la IMU y cuando se utilizan las cámaras para la corrección, los puntos se encuentran fijos y dados, eliminando la carga computacional requerida para calcular, marcar y rastrear los puntos en métodos SLAM (“Simultaneous Localization and Mapping”) del estado del arte similares al explicado anteriormente utilizado por las Quest 2. Precisamente en las Quest 2 se detectó el problema de entornos no estáticos que causaban errores debido a los movimientos de los puntos de referencia, esto no ocurre con el sistema desarrollado ya que tiene los puntos fijos e incluso es capaz de trabajar sin cámaras.

En cuanto a los resultados de la ejecución, como se ha dicho habrá videos demostrativos en la defensa que demuestren el movimiento en el mundo real y su representación en el entorno Unity, sin embargo, como es obvio, este no es un formato adecuado para este documento, por lo que se utilizarán capturas numéricas y gráficas que representen el movimiento causado.

Se comienza con la situación en reposo en la cual la posición en x, y, z se mantiene en 0.0 para todos los ejes (Figura 23), esto puede resultar evidente,

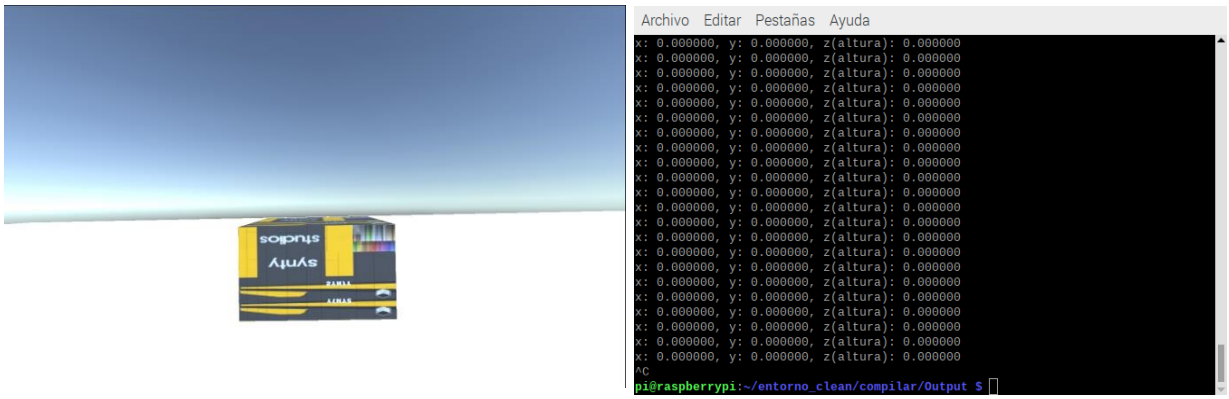

Figura 23. Estado reposo con mecanismos.

pero tras el resto del documento se puede ver que no lo es tanto, ya que para llegar a conseguir que la situación con la IMU en reposo de ese resultado deseado han entrado en acción todos los mecanismos que se han ido desarrollando en el documento, especialmente los de eliminación de ruidos, filtros bajos y la matriz de rotación para anular que la gravedad interfiera nocivamente en el cálculo de la posición.

A continuación, se desactivan dichos mecanismos para dejar patente lo que pasaría en una situación nuevamente de reposo, pero sin aplicar las técnicas utilizadas.

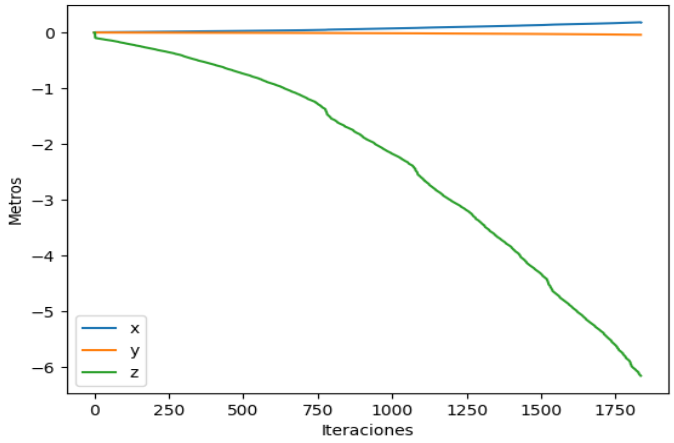

Figura 24. Estado reposo sin mecanismos.

Al cabo de pocos segundos (unas 1750 iteraciones), vemos en la Figura 24 que el eje z estaría fuera de precisión ya por 6 metros, y esto solo seguirá aumentando a lo largo del tiempo debido al deriva o error acumulado en la doble integración, las lecturas del eje x y eje y, por suerte, ha coincidido que en esta

ejecución tienen un error significativamente menor y se desviarían menos. Sin embargo, este desvío seguiría siendo muy elevado ya que el eje x estaría alrededor de 0.3 metros desviado y el eje y 0.1 metros en solo unos pocos segundos, el sistema en este estado no sería viable ya que se requiere precisión de milímetros para conseguir una buena experiencia de usuario, lo cual se consigue con los mecanismos activados. En cuanto a la representación en Unity, al ser el error en el eje z y no haber hecho colisionadores en el suelo, simplemente nos saldríamos del entorno creado traspasando el suelo.

Referente a la estimación de la rotación, como podemos ver en las siguientes figuras se ha conseguido una precisión muy elevada, que, al ojo, es prácticamente idéntica al caso real y consigue buenas sensaciones en el uso.

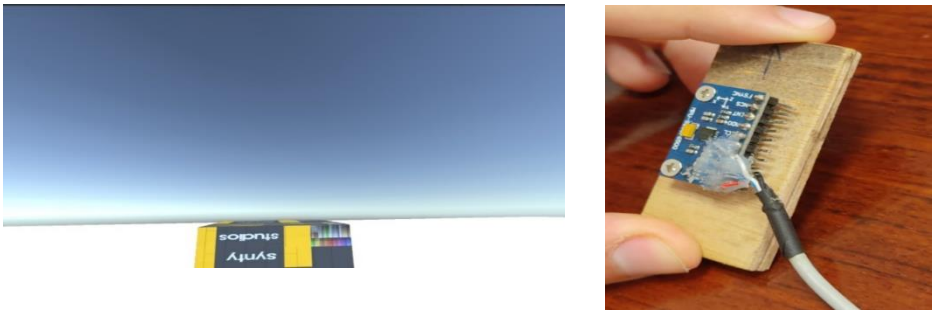

Figura 25. Cabeceo (pitch).

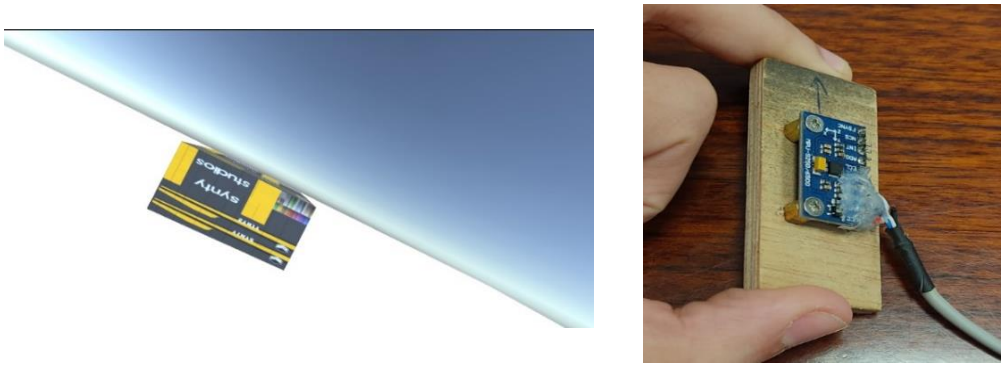

Figura 26. Alabeo (roll).

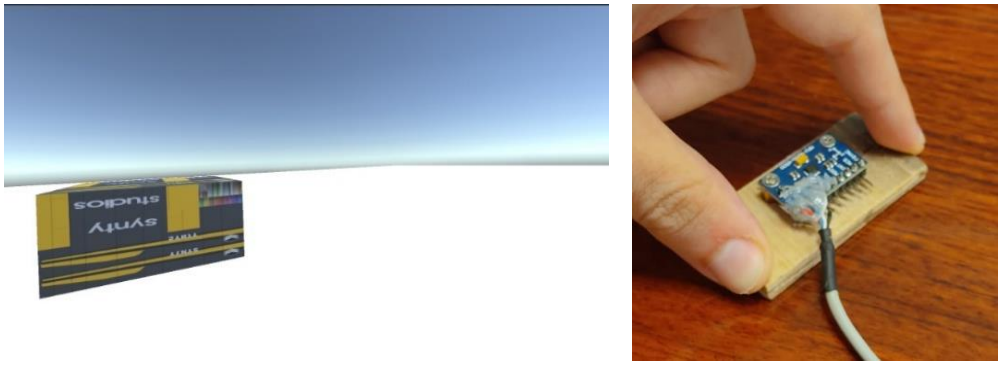

Figura 27. Guiñada (yaw).

Para demostrar el posicionamiento, se han ido realizando movimientos para cada eje.

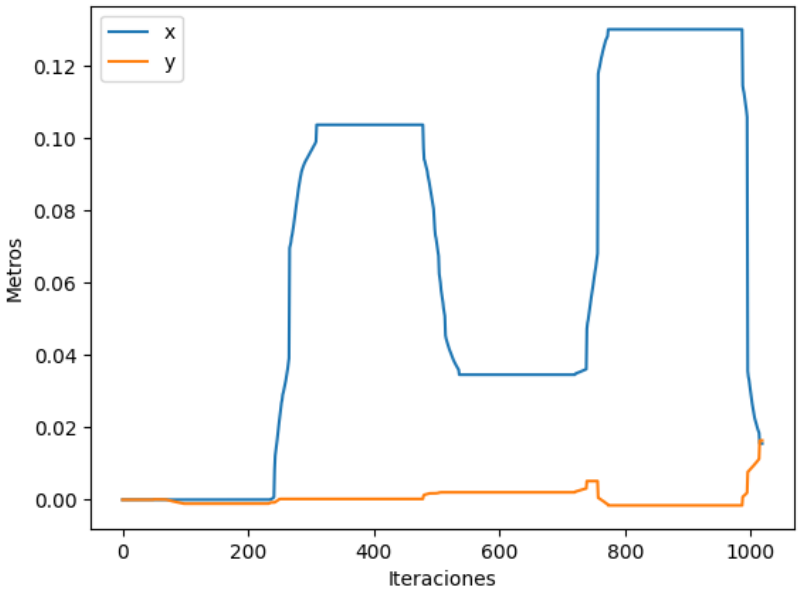

Figura 28. Movimiento en el eje x.

En esta primera Figura 28 se pueden observar dos movimientos en la vida real consistentes en un desplazamiento de la IMU de 10 centímetros hacia la derecha, volviéndose posteriormente hacia la izquierda, y nuevamente repitiendo un movimiento similar. Como podemos ver, ni en cada movimiento se movió la misma distancia ni ambos movimientos son iguales, además, la IMU se movió también en el eje y. El eje z no aparece ya que se mantuvo en 0 (el movimiento se realizó encima de una mesa). El motivo de estas circunstancias es que el

movimiento se produjo desplazando la IMU sin asistencia robótica o procedimientos de alta precisión, por lo cual es difícil hacer movimientos exactos. En cuanto al eje y, Figura 29, como se puede ver, se realizó un movimiento hacia atrás de 4 centímetros, hacia delante de aproximadamente 8 centímetros y se volvió a la posición de origen.

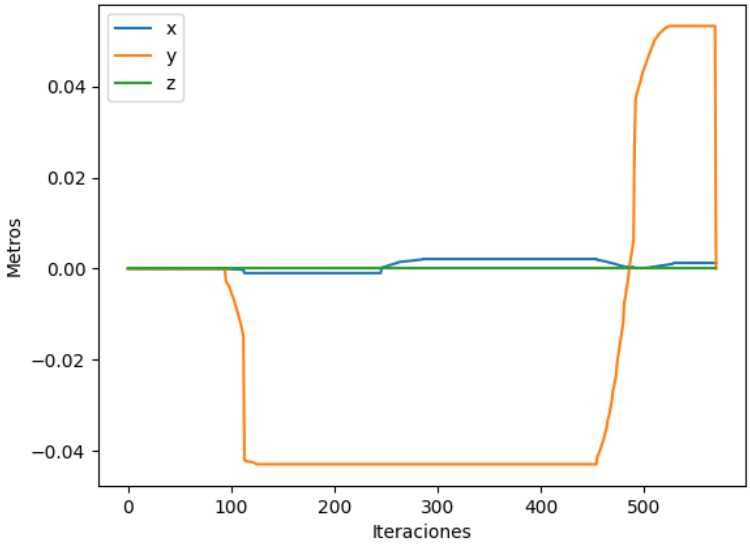

Figura 29. Movimiento en el eje y.

Para el eje z, Figura 30, se levantó la IMU de la mesa aproximadamente 20 centímetros, se volvió a posar sobre la misma y se repitió el movimiento. No se pudo evitar mover ligeramente la IMU hacia el frente al hacer el movimiento como se puede apreciar en las lecturas del eje y.

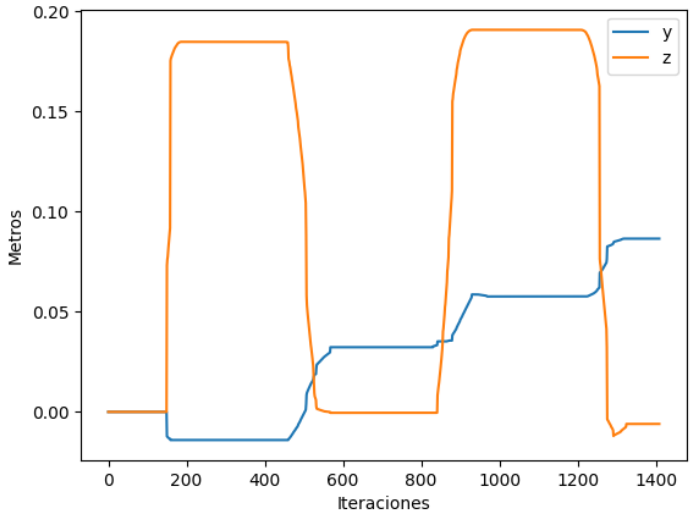

Figura 30. Movimiento en el eje z.

En cuanto a la precisión obtenida, como se puede apreciar en el caso de la Figura 30, al volver a posar la IMU en la mesa, el resultado de la posición está prácticamente perfecto en el 0.00, por lo que podemos ver una precisión que consigue resultados con un error significativamente inferior al centímetro. Además, para el resto de los ejes se realizaron más pruebas basando el movimiento en el recorrido de una regla con topes a ambos lados, conociendo de esta manera el punto de inicio y final en el mundo real de manera precisa. Se obtuvo un error medio de aproximadamente 0.20 centímetros en 8cm de recorrido y de 0.40 centímetros en 15 centímetros de recorrido, que es el máximo recorrido que se probó debido a la limitación con los cables de la IMU a la Raspberry PI y a que el sistema inercial, como se explicó anteriormente, está pensado para movimientos cortos (ya que para más largos sería necesario corregir con el sistema óptico a causa del deriva debido a que el mecanismo de reinicio que se usa cuando se detecta la IMU parada tardaría demasiado en actuar, y el error acumulado sería excesivamente grande).

A continuación, se muestra un ejemplo de corrección realizado por el sistema óptico con respecto al sistema inercial, para ello, se simulará por software que el sistema inercial está fallando y está sacando como resultado una posición muy grande en un tiempo muy pequeño y diferente de la dada por el sistema óptico, por lo que este último entrará en acción, detectará que el resultado inercial debe estar errado y corrige el resultado.

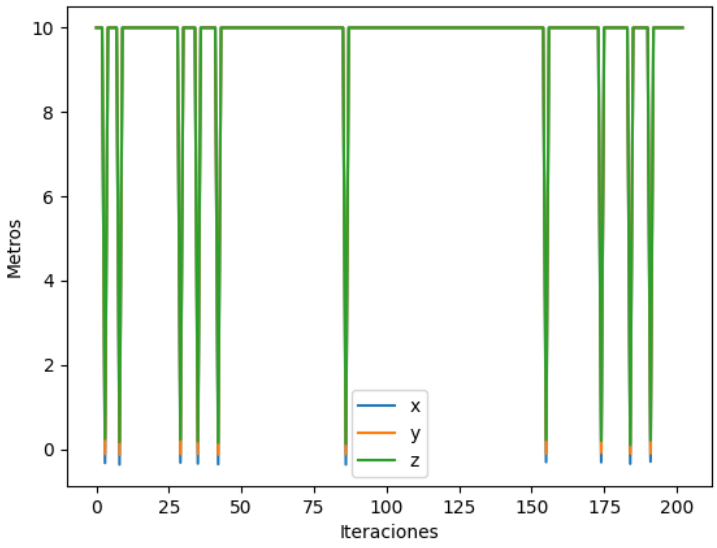

Figura 31. Corrección movimientos grandes.

Se ha hecho que la IMU dé en cada iteración un resultado de 10, 10, 10 metros en sus ejes como podemos observar en la Figura 31 y el sistema óptico corrige estas posiciones con resultados cercanos al cero como es esperado. En este caso en concreto se ha optado por no corregir el fallo de la IMU y hacer que siga dando 10 todo el tiempo para mejorar la visualización, pero en un caso real se tomaría el valor del sistema óptico y se reiniciaría la IMU a partir del punto dado por el sistema óptico, corrigiendo el problema.

6. Conclusiones

Se ha desarrollado un sistema de posicionamiento para entornos XR empleando métodos significativamente más sencillos que los utilizados por la competencia en el mercado actual. Esto lo hace una alternativa potencialmente más barata con respecto a los mismos y con un gran número de posibilidades para traer cualquier objeto del mundo real a entornos XR además de a usuarios. También funciona en condiciones muy variadas de luz, tanto con mucha como con poca. Adicionalmente, es capaz de funcionar sin necesidad del sistema óptico de manera puramente inercial durante cantidades muy grandes de tiempo obteniendo resultados precisos mientras se produzcan movimientos cortos y eliminando los problemas de los sistemas ópticos en el proceso.

Además, dicho sistema de posicionamiento ha sido implementado y utilizado en una aplicación Unity, demostrando que todo lo desarrollado funciona como se ha descrito de manera práctica y en casos de uso realistas.

7. Trabajo futuro

El proyecto tiene varias vertientes posibles de trabajo futuro.

La primera de ellas es integrar el proyecto, tanto hardware como software, en un único sistema portátil de manera cómoda, el proyecto ya se encuentra pensado en su realización para ello, lo que haría falta sería añadir una fuente externa de alimentación para la Raspberry Pi, por ejemplo un banco de energía con la suficiente capacidad, de manera que el sistema no esté limitado al alcance del cable a la toma de corriente y, tras sujetar de alguna manera los componentes al visor o al propio usuario, ya estaría el sistema listo. Otra solución más a largo

plazo sería desarrollar el propio visor con todos los componentes y software embebidos.

Además, como se ha desarrollado, desde el departamento TEISA de la universidad de Cantabria se trabaja con una metodología de modelado para sistemas embebidos basada en UML y código desarrollado siguiendo unas determinadas pautas y con filosofía modular llamado S3D desarrollado en la propia universidad [21]. El código ha sido programado siguiendo estas directrices (módulos, interfaces…) y está listo para ser utilizado en la herramienta S3D que permite entre otras cosas, ejecutar múltiples instancias del programa o los módulos de manera independiente de manera distribuida cómodamente, ejecutar diferentes implementaciones para las mismas interfaces etc. Sería interesante hacer una comparativa de rendimiento con la versión distribuida o incluso ejecutar varias instancias y comparar/fusionar los resultados.

Por otro lado, se podría explorar la idea de hacer posicionamiento de distintas partes del cuerpo de manera independiente consiguiendo de esta manera posicionamiento de cuerpo completo, añadiendo más IMUs a brazos, muñecas, piernas etc.

También se podría desarrollar algún tipo de algoritmo de fusión para combinar el sistema óptico e inercial de manera más sofisticada con respecto a la actual.

Finalmente, se podrían obtener resultados más exactos de precisión realizando pruebas con métodos más precisos como la utilización de un brazo robótico que mueva el sistema de un punto inicial a un punto final conocidos de manera correcta y analizar el resultado obtenido por la IMU en comparación con el punto conocido.

8. Bibliografía

[1]: Consumer virtual reality (VR) market size worldwide 2023 | Statista. (2020). Retrieved 18 April 2022, from https://www.statista.com/statistics/528779/virtual-reality-market-size-worldwide/

[2]: Virtual Reality Market Size & Share Report, 2022-2030. (2022). Retrieved 20 April 2022, from https://www.grandviewresearch.com/industry-analysis/virtual-reality-vr-market

[3]: Meta spent $10 billion on the metaverse in 2021, dragging down profit. Retrieved 29 April 2022, from https://www.nytimes.com/2022/02/02/technology/meta-facebook-earnings-metaverse.html
[4]: VRChat - Steam Charts. (2022). Retrieved 29 April 2022, from https://steamcharts.com/app/438100#All
[5]: Jean-Michel Jarre estrenará su nuevo espectáculo en VRchat. (2022). Retrieved 29 April 2022, from https://www.realovirtual.com/noticias/10836/jean-michel-jarre-estrenara-su-nuevo-espectaculo-vrchat
[6]: Events — VRChat. (2022). Retrieved 29 April 2022, from https://hello.vrchat.com/events
[7]: Especificaciones de hardware de Azure Kinect DK. (2022). Retrieved 27 April 2022, from https://docs.microsoft.com/es-es/azure/Kinect-dk/hardware-specification
[8]: Cámara de profundidad de Azure Kinect DK. (2022). Retrieved 29 April 2022, from https://docs.microsoft.com/es-es/azure/kinect-dk/depth-camera
[9]: Heaney, D. (2019). How virtual reality positional tracking works. Retrieved 29 April 2022, from https://venturebeat.com/2019/05/05/how-virtual-reality-positional-tracking-works/2/
[10]: Oculus Rift CV1 sensor and stand. [Image]. Retrieved from https://en.wikipedia.org/wiki/File:Oculus-Rift-CV1-Sensor-wStand.jpg
[11]: Oculus. (2015). The Oculus Rift, Oculus Touch, and VR Games at E3. Retrieved 23 April 2022, from https://www.oculus.com/blog/the-oculus-rift-oculus-touch-and-vr-games-at-e3/
[12]: Hesch, J., Linde, O., & Kozminski, A. (2019). Powered by AI: Oculus Insight. Retrieved 29 April 2022, from https://ai.facebook.com/blog/powered-by-ai-oculus-insight/
[13]: Kalman, R. E.; *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the ASME - Journal of Basic Engineering Vol. 82: pag. 35-45 (1960).
[14]: Beavers, I. (2017). *Rarely Asked Questions—139 The Case of the Misguided Gyro* [Ebook] (1st ed.). AnalogDialogue. Retrieved from https://www.analog.com/media/en/analog-dialogue/raqs/raq-issue-139.pdf

[15]: Universidad de Cantabria. (2014). Método y sistema de localización espacial mediante marcadores luminosos para cualquier ambiente. Spanish.
[16]: Gómez, R. (2020). *Sistema de posicionamiento 3D visual multipróposito, mediante marcas led*. Santander: UCrea. Retrieved from https://repositorio.unican.es/xmlui/handle/10902/21417
[17]: Currawong, N. (2022). *RTIMULib2* [Windows/Linux/Mac] from https://github.com/Nick-Currawong/RTIMULib2
[18]: Yaw, pitch, and roll rotations. (2006). Cambridge University Press. Retrieved 29 April 2022, from http://planning.cs.uiuc.edu/node102.html
[19]: González,H. (2022). *https://replit.com/@HugoGonzalez14/mediaplot#main.py* [Windows/Linux/Mac].
[20]: Unity. (2022). Unity - Manual: Prefabs. Retrieved 29 April 2022, from https://docs.unity3d.com/es/530/Manual/Prefabs.html
[21]: Universidad de Cantabria. *S3D* [Eclipse]. https://s3d.unican.es/documentation/
[22]: Oculus. (2020). Screenshot (2:12) [Video]. Retrieved from https://www.youtube.com/watch?v=d73PfTXZDvo
[23]: VIPPE Project – Virtual Platform Parallel performance Evaluation. (2014). Retrieved 26 May 2022, from https://vippe.unican.es/
[24]: Ultra-wideband (UWB o Banda ultra ancha). Retrieved 6 June 2022, from https://www.etsist.upm.es/estaticos/ingeniatic/index.php/tecnologias/item/639-ultra-wideband-uwb-o-banda-ultra-ancha.html
[25]: Two way ranging diagram. [Image]. Retrieved from https://www.inpixon.com/hubfs/website_graphics/TWR.jpg
[26]: Indoor Positioning Comparative table. [Image]. Retrieved from https://www.inpixon.com/technology/standards/ultra-wideband