UNIVERSIDAD ECCI
COORDINACIÓN DE INGENIERÍA ELECTRÓNICA

TÍTULO

SIMULADOR CLÍNICO BASADO EN REALIDAD VIRTUAL PARA LA TOMA DE MUESTRAS DE GASES ARTERIALES RADIALES

PARA OBTENER EL GRADO DE:

TECNÓLOGO EN ELECTRÓNICA INDUSTRIAL

AUTORES:

SANTIAGO MARTINEZ GARCIA

MICHEL JULIANA ROJAS GONZALEZ

DIRIGIDO POR:
ING. JUAN EMILIO SANABRIA SANABRIA - Director
ING. VICTOR HUGO BERNAL - Codirector

BOGOTÁ DC - COLOMBIA

2022-2023

TABLA DE CONTENIDO

1. ÍNDICE DE FIGURAS

2. ÍNDICE DE TABLAS

3. RESUMEN

A partir del proyecto de investigación de la convocatoria 006 del 2020 de la Universidad ECCI denominado SIMULADOR CLÍNICO BASADO EN REALIDAD VIRTUAL PARA LA ENSEÑANZA DE CIENCIAS DE LA SALUD, el presente proyecto hace parte de una de las líneas principales del desarrollo de dicha convocatoria, este proyecto busca desarrollar una aplicación de software y hardware la cual pueda interactuar con el usuario generando un entorno simulado de realidad virtual del procedimiento de toma de muestra de gases arteriales (radial), dicha aplicación se basa en el trabajo de grado de la dirección de enfermería denominado procedimiento clínico a desarrollar por medio del modelo AHP(método cuantitativo para la toma de decisiones multicriterio).

4. TÍTULO DE LA INVESTIGACIÓN

SIMULADOR CLÍNICO BASADO EN REALIDAD VIRTUAL PARA LA TOMA DE MUESTRAS DE GASES ARTERIALES RADIALES

5. PROBLEMA DE INVESTIGACIÓN

5.1. DESCRIPCIÓN DEL PROBLEMA

Se está desarrollando un simulador clínico de toma de gases arteriales, el cual permitirá diferentes tipos de interacción en un espacio clínico, para que el usuario pueda comprender y tener un primer acercamiento al procedimiento.

5.2. FORMULACIÓN DEL PROBLEMA

¿Cómo puede la realidad virtual en un entorno de simulación proporcionar una experiencia realista y efectiva para el aprendizaje de la técnica de toma de muestras de gases arteriales radiales, mejorando los conocimientos y la seguridad de los profesionales en el área de la salud?

6. OBJETIVOS DE LA INVESTIGACIÓN

6.1. OBJETIVO GENERAL

Desarrollar un simulador basado en realidad virtual para el proceso de enseñanza aprendizaje del procedimiento clínico de toma de gases arteriales para estudiantes de ciencias de la salud en la Universidad ECCI.

6.2. OBJETIVOS ESPECÍFICOS

- Determinar los requisitos para la realización del simulador basado en realidad virtual para el procedimiento clínico de toma de gases arteriales.
- Diseñar el simulador basado en realidad virtual a partir de las herramientas de hardware digitales y software especializado.
- Realizar la adquisición de datos para hacer una interpretación gráfica de las señales por medio de Matlab.

7. JUSTIFICACION Y DELIMITACION DE LA INVESTIGACIÓN

7.1. JUSTIFICACIÓN

Las herramientas virtuales son cada vez más utilizadas hoy en día para el proceso de la enseñanza, dadas las circunstancias de los últimos años por la crisis sanitaria por el SRAS-COV[1] como una consecuencia de lo anterior fue el aislamiento obligatorio lo que generó las necesidad de innovar en métodos que permitieran realizar prácticas de laboratorio de forma segura. La simulación es una herramienta de aprendizaje para el desarrollo educativo lúdico y didáctico que permite al alumno actuar en un escenario real recreando diferentes entornos, la realidad mixta permite replicar entornos reales reconstruidos de forma digital, la apropiación de estas herramientas permiten el acercamiento de la práctica con la teoría de forma remota, para el presente caso vista desde el proceso de la toma de muestra de gases arteriales radiales la cual es un examen médico complejo de realizar, antes de realizarlo en una persona el poder entrenar con un simulador a enfermeros y médicos en el procedimiento minimiza los posibles errores que pueden ocurrir en dicho proceso, aumentado la confianza y perfeccionando la técnica con que lo realizan con un aprendizaje significativo[2], esta es una gran ventaja ya que pueden equivocarse y repetir cuantas veces sea necesario desarrollando así un aprendizaje emocional junto a la memoria muscular[3] para una capacitación completa. Este proyecto hace parte de la convocatoria interna de investigación de la universidad ECCI número 006 del 2020.

7.2. DELIMITACIÓN

El presente proyecto contempla tanto el diseño de software como integración de hardware para interactuar realidad virtual en el proceso didáctico del simulador para el acercamiento a herramientas propias de la enfermería minimizando tanto el riesgo que existe como los errores en un escenario real, optimizando el proceso de la toma de muestra. Se pretende que el entorno virtual sea interesante, agradable y sencillo de manejar, para que el estudiante se mantenga cautivado frente a la realización de la guía y le aporte los conocimientos necesarios para realizar el procedimiento.

8. MARCO DE REFERENCIA

8.1. MARCO TEÓRICO

8.1.1. MODELO AHP

El modelo AHP es una forma lógica y estructurada de trabajar que agiliza la toma de decisiones complejas cuando hay múltiples criterios o atributos al descomponer un problema en una estructura jerárquica. Puede descomponer un atributo complejo en un conjunto de atributos más simples y decidir cómo cada uno de ellos estos atributos individuales afecta el objetivo de la decisión. Esta influencia está representada por las asignaciones de valor asignadas a cada atributo o criterio.

La aplicación del método AHP no requiere información cuantitativa sobre el resultado de cada alternativa bajo cada estándar considerado, sólo los juicios de valor de los tomadores de decisiones.[4]

8.1.2. UNITY

Unity es un motor de juegos y herramienta de desarrollo de software que se utiliza para crear juegos y otras experiencias interactivas en 2D y 3D. Fue creado por Unity Technologies y es utilizado por una amplia variedad de desarrolladores de juegos y otras empresas para crear contenido para dispositivos móviles, consolas de videojuegos, PC y realidad virtual.[5]

Uno de los principales componentes de Unity es su editor de escena, que permite a los desarrolladores crear y modificar la apariencia y la funcionalidad de sus juegos. El editor incluye una amplia variedad de herramientas para modelar, texturizar y animar objetos, así como para añadir iluminación y efectos visuales. También hay una amplia variedad de componentes y scripts disponibles que se pueden utilizar para añadir funcionalidades específicas a los juegos, como la física y el sonido.[6]

8.1.3. OCULUS

La realidad virtual es una tecnología que permite a los usuarios sumergirse en entornos virtuales y experimentar sensaciones de estar presentes en un mundo completamente diferente al que nos rodea. Esto se logra mediante el uso de dispositivos de VR, que incluyen gafas especiales y controles que permiten a los usuarios interactuar con el mundo virtual.

Oculus ha sido pionera en el desarrollo de dispositivos de VR de alta calidad y ha trabajado en la creación de una amplia variedad de aplicaciones y juegos para sus dispositivos. Además, ha desarrollado tecnologías de realidad aumentada y mixta,

que combinan elementos de la realidad virtual y la realidad real para crear experiencias de realidad híbrida.[7]

8.1.3.1 QUEST 2

El Oculus Quest 2 es un visor de realidad virtual autónomo fabricado por Oculus, permite a los usuarios experimentar la realidad virtual sin necesidad de un ordenador o una consola de juegos, ya que cuenta con un procesador y batería integrados.

El dispositivo presenta pantallas OLED con una resolución de 1832x1920 por ojo y una frecuencia de actualización de 90 Hz, lo que proporciona una experiencia de realidad virtual muy inmersiva y detallada. Además, tiene controladores de movimiento que permiten a los usuarios interactuar con los objetos virtuales y una amplia variedad de juegos y aplicaciones disponibles.

El Quest 2 también cuenta con una opción para conectarse a un ordenador mediante un cable, lo que permite a los usuarios acceder a juegos y aplicaciones más potentes que no están disponibles en la tienda Oculus.[8]

8.1.4. MATLAB

Es una plataforma de programación y cálculo numérico utilizada por millones de ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos, está presente en sistemas de seguridad activa de automóviles, naves espaciales interplanetarias, dispositivos de monitorización de la salud, redes eléctricas inteligentes y redes móviles LTE. Se utiliza para aprendizaje automático, procesamiento de señales, procesamiento de imágenes, visión artificial, comunicaciones, finanzas computacionales, diseño de control, robótica y muchos otros campos.

El lenguaje de Matlab está basado en matrices, es la forma más natural del mundo para expresar las matemáticas computacionales. Las gráficas integradas facilitan la visualización de los datos y la obtención de información a partir de ellos.

8.1.5. ARDUINO

Es una plataforma electrónica de código abierto, esta permite a la comunidad de desarrollo crear una amplia variedad de microcontroladores de placa única que se pueden usar para una variedad de propósitos. Una placa con todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador, fue creada por estudiantes para estudiantes.

Un microcontrolador es un circuito integrado que puede registrar instrucciones escritas utilizando el lenguaje de programación disponible en el entorno IDE de Arduino. Puede utilizar estas instrucciones para crear un programa que interactúe con los circuitos de su placa.Arduino no tiene un modelo de placa específico, eso

significa que puede encontrar muchos tipos diferentes de placas que comparten el diseño básico, disponible en una variedad de formas, tamaños y colores para adaptarse a las necesidades del proyecto en el que está trabajando, básico o extendido, IoT o impresión 3D.

8.1.6. ACELERÓMETRO Y GIROSCOPIO MPU6050

Es una unidad de medición inercial o IMU (Inertial Measurement Units) de 6 grados de libertad pues combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Este sensor es muy utilizado en navegación, goniometría, estabilización, etc. Este sensor cuenta una precisión bastante buena, posee además un hardware de conversión análogo-digital de 16 bits por cada canal, por lo tanto puede leer el valor de X, Y y Z al mismo tiempo. Funciona de 3 a 5V.

9. ESTADO DEL ARTE

El uso de la tecnología en la educación ha tomado fuerza en los últimos años, permitiendo que todo tipo de estudios que requieren prácticas o simulaciones se apoyen de los distintos tipos de realidades ofrecidas en la actualidad, debido a capacidad de sumergir al estudiante en una exploración del campo requerido por medio de imágenes [1]. Como lo es en este caso, la práctica de un proceso clínico a través del modelo AHP, permitiendo capacitar y evaluar jerárquicamente las habilidades de los estudiantes de enfermería de la institución [2].

Para tal motivo, se requiere el uso de la realidad virtual (RV) como una tecnología emergente que permite el uso de imágenes y entornos virtuales basados en el mundo real, por medio de algún dispositivo como el computador, tableta o teléfono inteligentes [1], permitiendo el aumento o cambio de la percepción que el estudiante tiene, convirtiéndolo en una herramienta que permite completar la realidad por medio de elementos virtuales, preparando a los estudiantes en posibles casos que pueden enfrentarse en su vida profesional. Para lograr complementar el modelo AHP planteado, se puede contemplar el uso de la RV por medio de la cuarta dirección de la implementación de la RV en la educación de Yuen, Yaoyuneyong y Johnson [3], permitiendo el entrenamiento de habilidades, apoyando el entrenamiento de los estudiantes a partir del desarrollo de tareas específicas, generalmente desarrolladas por medio de cascos de realidad virtual, aprovechando los beneficios demostrados de la RV en la enseñanza.

Las categorías de los beneficios de la RV en entornos educativos de Diegmann et al [4], evalúan cómo la mente, la enseñanza, el aprendizaje y los costos hacen de esta tecnología la mejor opción a la hora de realizar simulaciones. Motivando al estudiante a prestar más atención, logrando una mejor concentración para satisfacer sus metas propias. Aumentando no solamente el aprendizaje del estudiante, sino también, el aprendizaje colectivo. Por medio de altos detalles en la información, permitiendo que los estudiantes interactúen buscando una curva de aprendizaje acelerada a partir de la creatividad y la memoria del estudiante en el entorno desarrollado reduciendo costos a largo plazo sobre materiales y transporte en laboratorios físicos.

La realidad virtual permite sumergir al usuario en entornos completamente simulados, donde se crea una experiencia inmersiva que reemplaza la percepción de la realidad física. Mediante el uso de dispositivos de visualización, como cascos de realidad virtual, los usuarios pueden interactuar con entornos virtuales generados por computadora y percibirlos como si fueran reales. Esta tecnología permite la creación de ambientes virtuales que pueden ser explorados y

manipulados, brindando una experiencia inmersiva y facilitando la integración de elementos virtuales en el contexto de aprendizaje.

La arquitectura de un sistema de realidad virtual cumple las tareas descritas a continuación:

1. Generación del entorno virtual
   Se desarrolla un entorno digital utilizando técnicas de modelado y animación 3D, que puede incluir objetos, escenarios y personajes virtuales.

2. Interacción del usuario
   Los usuarios pueden interactuar con el entorno virtual a través de dispositivos de entrada, como controladores o sensores de movimiento, lo que les permite explorar, manipular y participar en el entorno simulado.

3. Renderización y visualización
   El sistema de realidad virtual procesa gráficos en tiempo real y los muestra en dispositivos de visualización, como cascos de realidad virtual. Estos dispositivos presentan imágenes estereoscópicas a cada ojo, lo que crea la ilusión de profundidad y perspectiva en el entorno virtual.

4. Sonido y retroalimentación táctil
   Para una experiencia más inmersiva, se pueden incorporar efectos de sonido y retroalimentación táctil, como vibraciones o respuestas físicas, que complementan la experiencia visual y mejoran la sensación de presencia en el entorno virtual.

10. DISEÑO METODOLÓGICO

10.1 PLANEACIÓN DISEÑO UNITY

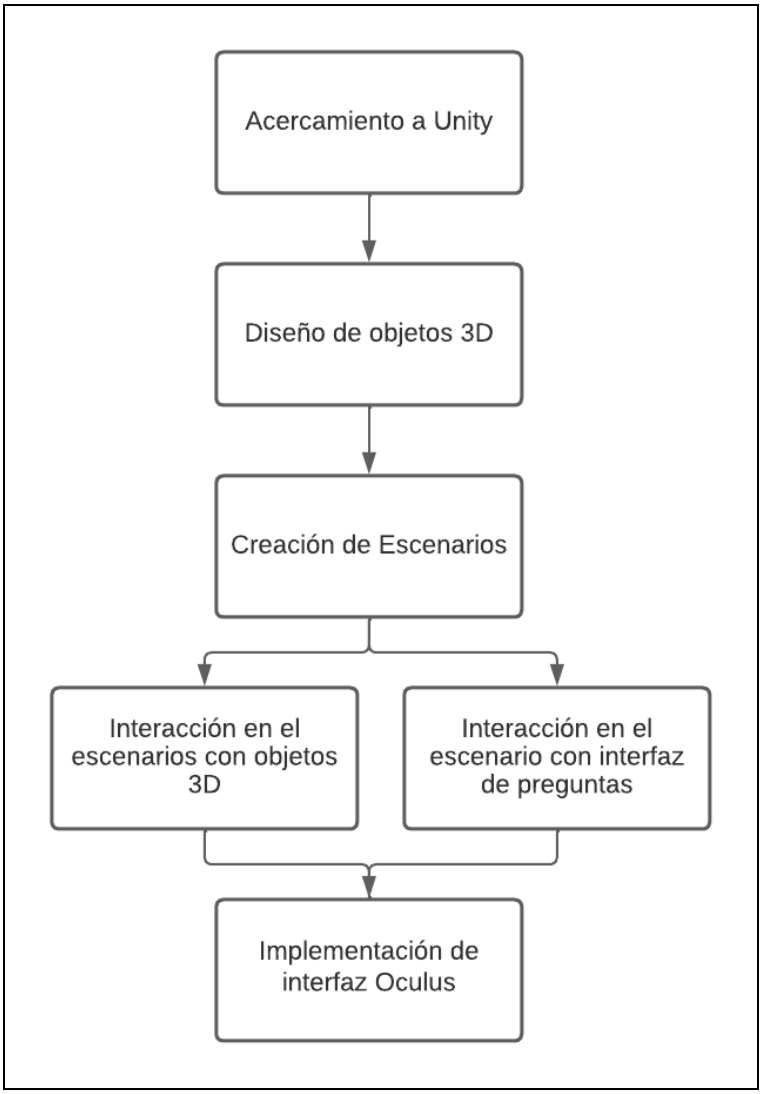

Figura 1. Diagrama de flujo proyecto en Unity

Para el diseño en el programa de unity el plan de trabajo en primer lugar es el acercamiento a este nuevo programa mediante tutoriales que el mismo programa nos ofrece, a su vez que cursos de internet, foros, videos, entre otros, para que de ese modo adquirir las habilidades necesarias para el desarrollo del proyecto.

Seguido a esto se enfocará en el diseño y creación de los diferentes implementos clínicos que son necesarios para este procedimiento (jeringa, gasas, solución de heparina, botella de alcohol, etc.), para este paso se consideran dos opciones, realizar los diferentes objetos en el mismo programa de Unity, ya que este permite el modelado 3D y se pueden agregan diferentes extensiones que permitan un

mejor desarrollo de los implementos; o podemos apoyarnos en un programa externo como Blender que se especializa en el diseño de objetos 3D y permite un mejor detalle.

Una vez finalizado los objetos, se cambia el enfoque a la creación de diferentes escenarios donde el usuario tendrá que interactuar con los objetos creados o con el entorno, debido a que existirán escenarios donde se realizarán preguntas para poner a prueba el conocimiento del usuario sobre el procedimiento clínico que se va a realizar, haciendo que se fortalezcan los conocimientos y de igual forma el acercamiento a cómo se vería un entorno clínico donde se realiza este procedimiento.

Por último se realizará la implementación del software de Oculus, el cual mediante diferentes códigos y propiedades en el entorno de Unity, permitirá realizar las interacciones en realidad virtual, como una visión más cerca de diferentes entornos y objetos, además de poder interactuar con ellos. En este caso un visor Quest 2, del cual también gracias a Unity se comprende su funcionamiento, gracias a extensiones que brinda el mismo programa para facilitar su programación.

10.2 PLANEACIÓN DISEÑO MATLAB

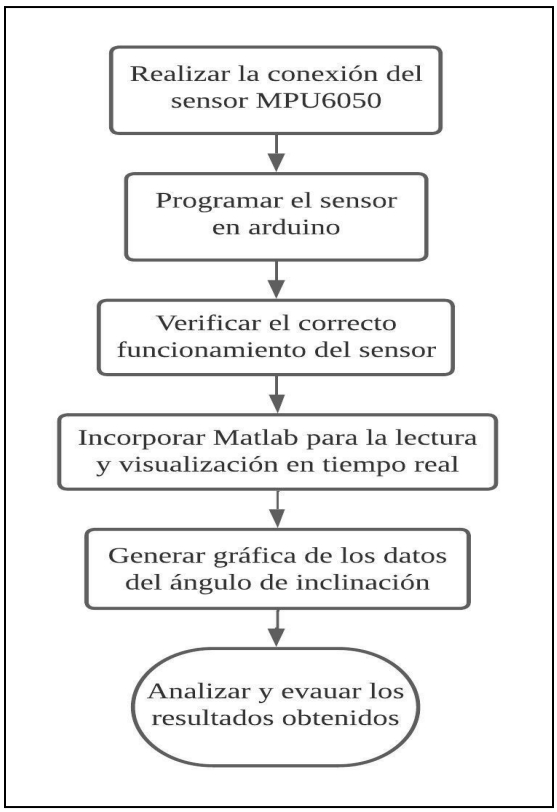

Figura 2. Diagrama de flujo Proyecto en Matlab

Para el modelo que se quiere lograr se plantea el uso de un sensor nos permita saber el ángulo de inclinación ya que este procedimiento requiere llevarse a cabo con alta precisión para que esté correcto, debe mantener un ángulo de 45 a 60 grados de la aguja en relación al brazo para así lograr llegar a la arteria radial (en caso de no realizarlo correctamente puede formar un coágulo dentro del vaso sanguíneo, aparecer trombosis o un espasmo arterial en otros casos hay presencia de hematoma en la región de la punción o infección local en la zona de extracción). Para lograr cumplir con este objetivo, se considera la implementación del sensor MPU6050 ya que cuenta internamente con acelerómetro y giroscopio.

Se tiene previsto programar en Arduino para verificar su correcto funcionamiento, esto es fundamental para detectar posibles errores o inconvenientes relacionados con el sensor o las conexiones utilizadas. Una vez que se haya asegurado el adecuado funcionamiento del sensor, se procederá a incorporar MATLAB. Esta herramienta permitirá leer la conexión en el ordenador a través de la comunicación serial y generar gráficas en tiempo real para visualizar los datos. Esta visualización en tiempo real resulta de gran utilidad, ya que brinda una representación visual de los datos del ángulo de inclinación, lo que facilita su análisis y seguimiento durante el procedimiento.

Con este enfoque, se espera desarrollar un modelo efectivo que proporcione información precisa sobre el ángulo de inclinación y que garantice la seguridad junto a la eficiencia en el procedimiento. Se espera que esta combinación de tecnologías cumpla con los objetivos establecidos y brinde resultados satisfactorios.

11. IMPLEMENTACIÓN

11.1 DISEÑO EN UNITY

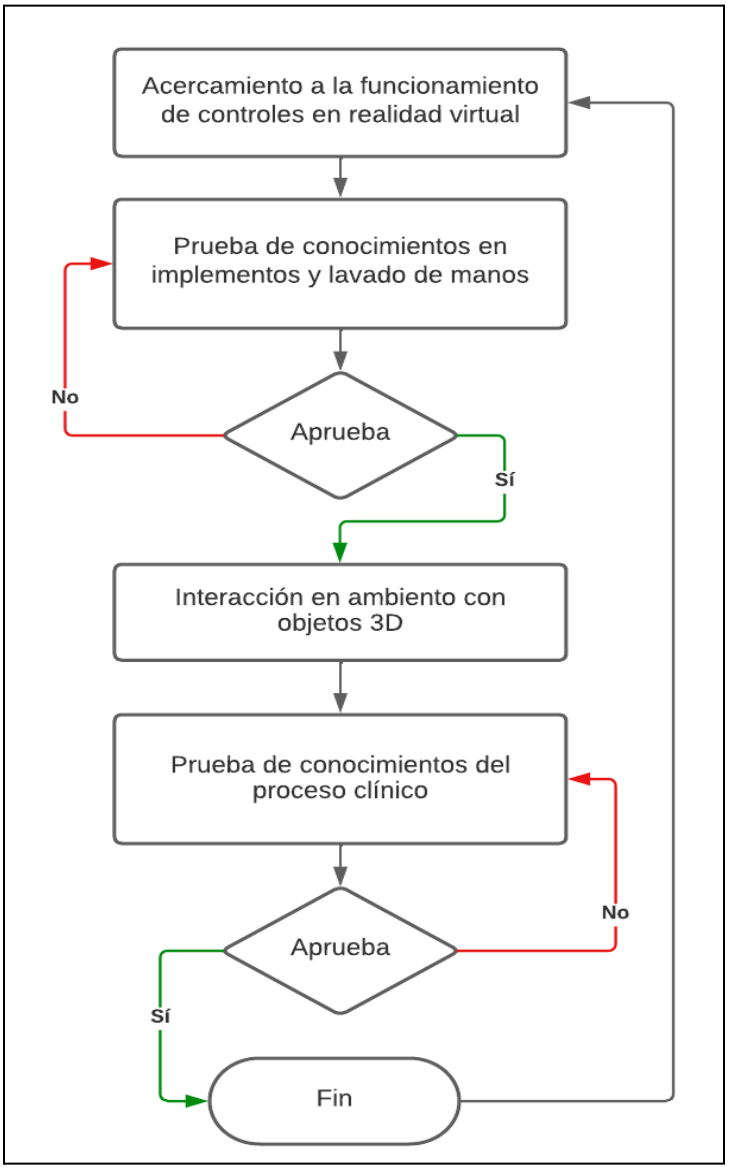

Figura 3. Diagrama de flujo Proceso Simulador

Inicialmente nuestro objetivo es enseñar cómo funciona nuestra interfaz en la realidad virtual, por lo cual diseñamos un escenario en el que brindemos un pequeño tutorial de como funciona el visor, los controles y las diferente interacciones que pueden estar presente durante todo el simulador para que su usuario se acomode y no tenga problemas en el desarrollo del proceso.

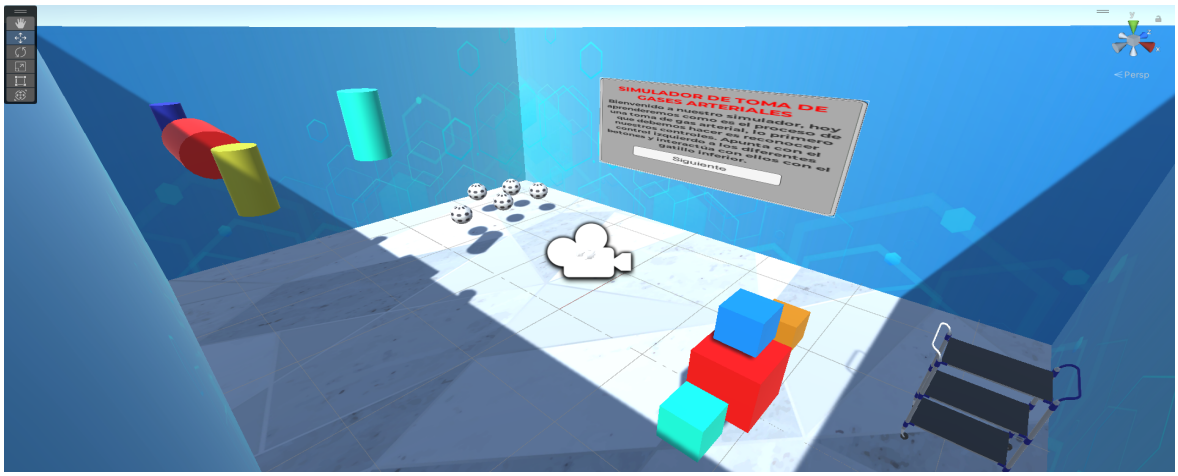

Figura 4. Escenario Principal en Unity

En el escenario principal el usuario podrá encontrar objetos básicos como, cubos o esferas con las cuales podrá interactuar, para que de este modo entienda el uso de los controles, también encontrará objetos más detallados que de igual forma podrá interactuar con estos además de un panel flotante con las instrucciones a seguir.

Por la parte de unity se empezó a desarrollar los diferentes objetos 3D con los que interactúa el usuario, algunos de estos fueron importados desde páginas web debido al nivel de detalle que requerían, los demás fueron diseñados en el mismo programa usando extensiones que permiten un mejor modelado de objetos 3D como ProBuilder.

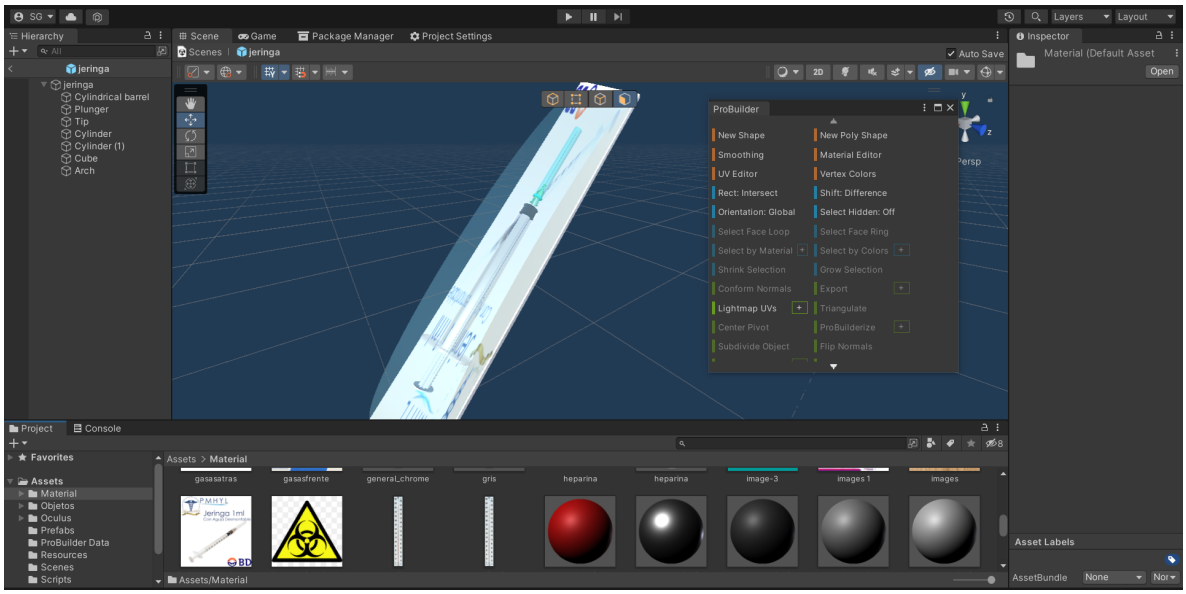

Figura 5. Creación de Objetos 3D

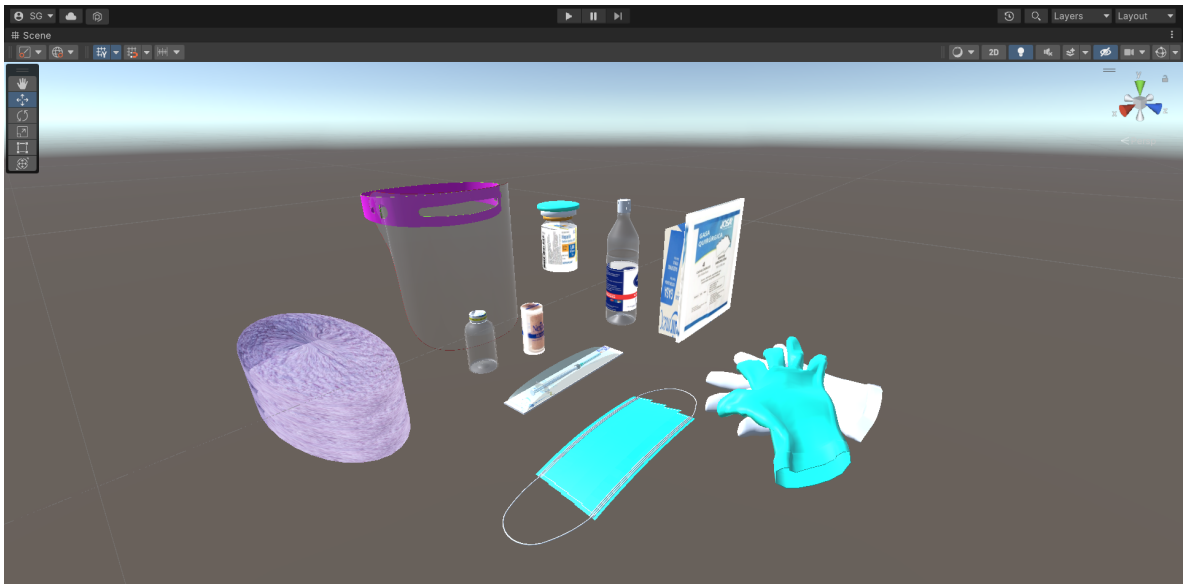

Figura 6. Objetos 3D Finales

Una vez finalizado los modelamientos 3D nos enfocamos en los diferentes escenarios donde el usuario podrá interactuar con el entorno y con los objetos anteriormente diseñados, los escenarios fueron diseñados desde cero con los diferentes implementos del programa, se utilizaron implementos ya diseñados de internet, para que el escenario sea lo más realista posible y recrea un ambiente clínico donde se realice la práctica de toma de gas arterial.

Figura 7. Escenario Consultorio Medico

En la figura anterior podemos ver una parte del escenario donde el usuario deberá recolectar los diferentes elementos para proceder a la toma de gas arterial,

podemos analizar las camillas donde el paciente se recostará, al igual que un armario donde se encontrarán los elementos que son requeridos.

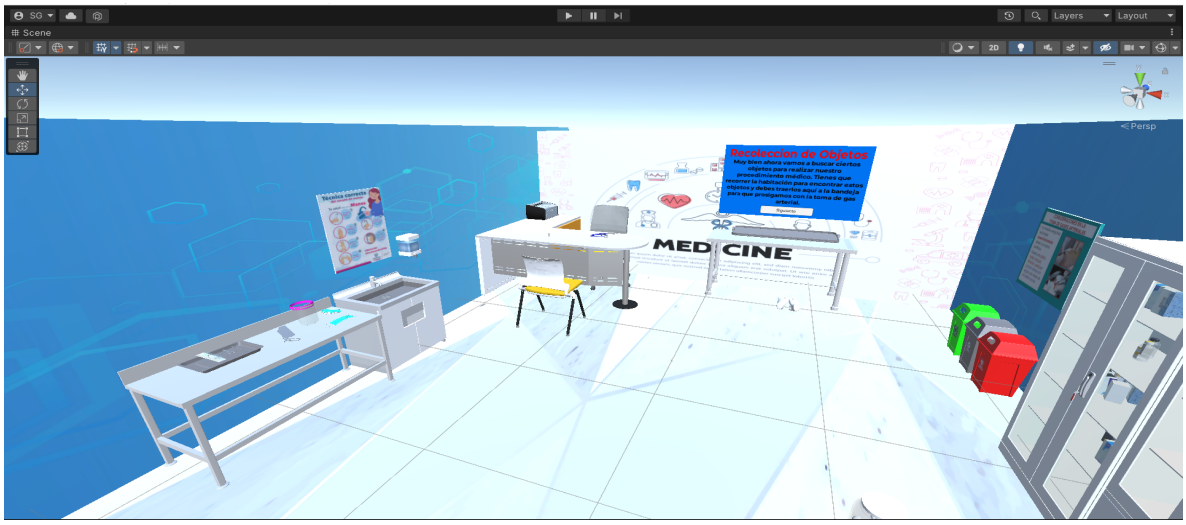

Figura 8. Escenario Consultorio Medico

Aquí podemos ver el resto del escenario clínico, podemos ver que cuenta con una zona de preparación de implementos, sus respectivas basuras, indicaciones de lavado de manos y entre otros detalles más, que hacen que nuestro escenario recree a un mayor detalle un escenario real.

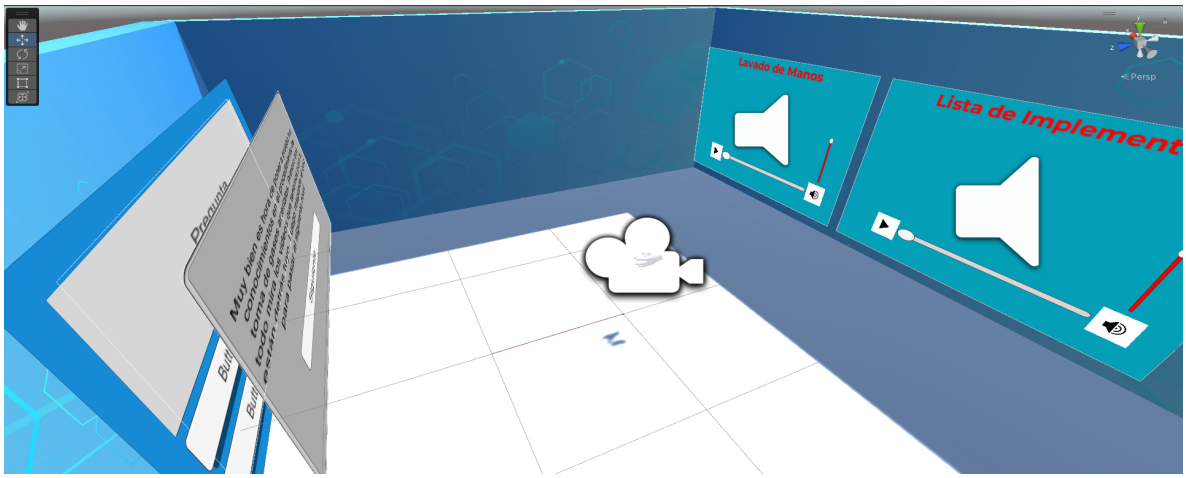

Figura 9. Escenario de Preguntas

Como podemos ver en las diferentes imágenes, Unity nos permite crear varios escenarios los cuales estarán el orden que se muestra en el diagrama de flujo, cada escenario fue creado aparte, es decir, que cada escenario es independiente, por lo cual el usuario únicamente estará con una interacción a la vez, una vez complete el objetivo del escenario podrá pasar al siguiente.

Lo último que se implementó en los diferentes escenarios fueron los Scripts, que permiten las diferentes interacciones del usuario con los implementos y el entorno, mediante el uso de la interfaz de Oculus, como la interfaz de preguntas, la visualización de los videos, el contador de objetos, entre otros. Estos códigos se realizaron mediante apoyo de paginas web y videos en el lenguaje de C# en la plataforma de Visual Studio.

11.2 DESCARGA Y EXPORTACIÓN DE LA APLICACIÓN

Una vez finalizado nuestro programa, para que lo podamos exportar a unas gafas Oculus, debemos asegurarnos que todo esté en formato Android, ya que las gafas VR Oculus manejan este tipo de plataformas, esto lo debemos hacer mediante el panel de “Build Settings” de Unity.

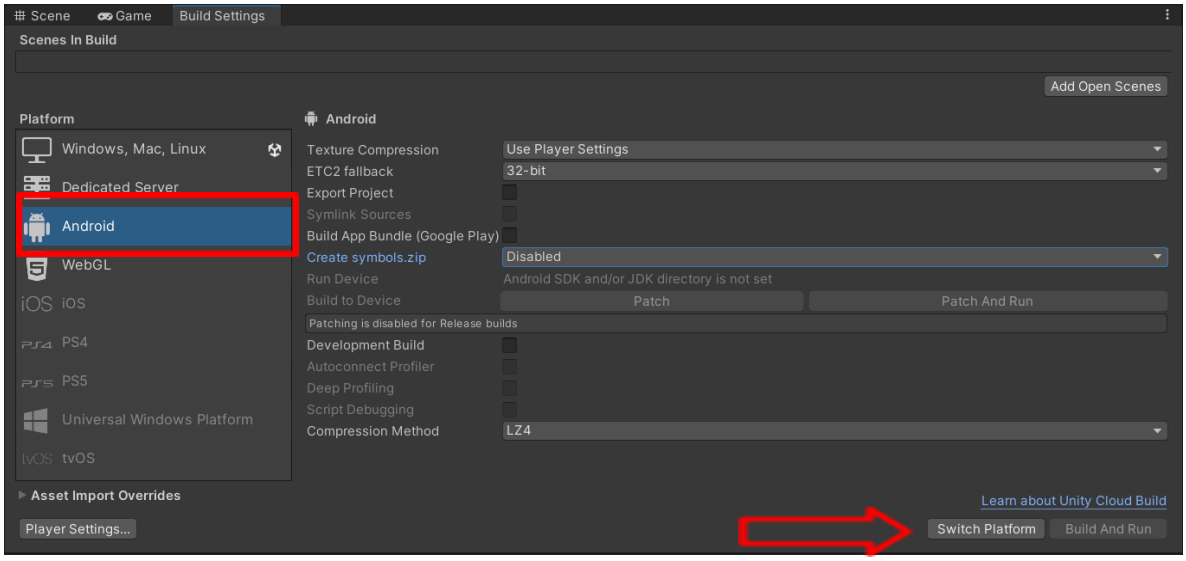

Figura 10. Panel de Build Settings de Unity

Una vez tengamos todo nuestro proyecto en la plataforma de Android vamos a asegurarnos de que todos los escenarios que hemos creado estén adjuntos, es importante que los añadamos en orden, debido a que si vamos a tener varios escenarios y estos están interconectados, al momento de cambiar de escenario seguirán el orden en que los seleccionamos. Por último le daremos al botón de “Build” y guardaremos nuestro Apk. Esto permitirá una fácil instalación de nuestra aplicación en dispositivos Oculus, ya que solamente debemos pasar el archivo Apk para que otros usuarios lo puedan porbar.

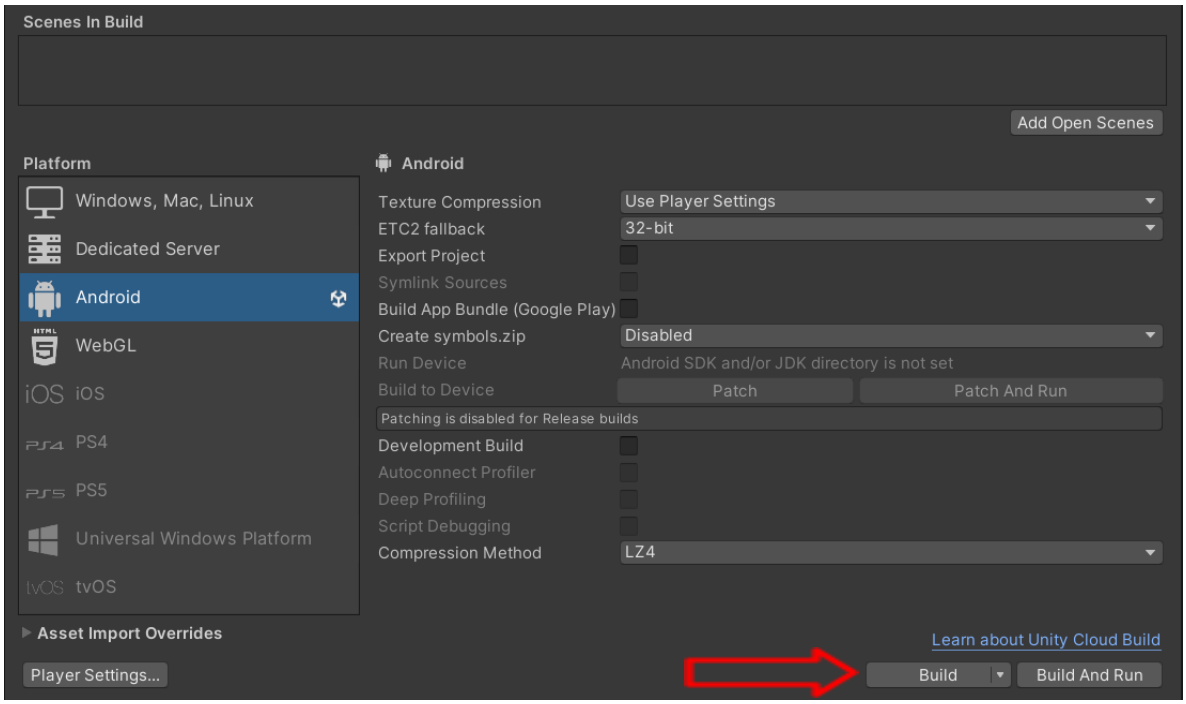

Figura 11. Panel de Build Settings de Unity

Para la instalación de nuestro Apk en nuestras gafas VR Oculus, existen varios métodos ya que se trata de un aplicación en la plataforma Android como cualquier otra, por lo cual se puede instalar con las mismas aplicaciones que nos brinda Meta, como lo son “Oculus developer hub”, aunque también puede ser por aplicaciones externas que permitan el manejo de dispositivos Android como lo puede ser el software adbLink, que es una herramienta utilizada en el desarrollo de aplicaciones para dispositivos Android, permite a los desarrolladores y usuarios realizar diversas acciones en sus dispositivos Android como instalación y desinstalación de aplicaciones y transferencia de archivos[9]. Lo más importante que debemos tener en cuenta para esto, es que los permisos de instalación de orígenes desconocidos estén activos en nuestros Oculus Quest.

Una vez tengamos el apk instalado en las gafas VR, lo único que debemos hacer es ir al apartado de Aplicaciones-Orígenes desconocidos, y ahí encontraremos nuestro apk listo para ser ejecutado.

11.3 MODELAMIENTO EN MATLAB

En la parte física utilizamos Arduino Uno el cuál se conecta al sensor MPU6050 de tal manera que Vcc corresponda a 5V en el arduino, GND se conectan entre sí, además para la comunicación utilizamos los pines A5 y A4 para el caso de SCL (Serial Clock) y SDA (Serial Data) respectivamente. Una vez conectados se comprueba en el IDE de Arduino que la comunicación entre ellos esté funcionando

correctamente, esto implica que los datos se están recibiendo en el monitor serial para su visualización que permite verificar que los valores de aceleración y giro obtenidos son coherentes y corresponden a los movimientos del sensor en tiempo real. Para esto se utiliza la librería ***Wire.h*** ya que proporciona funciones para enviar y recibir datos mediante la comunicación I2C (Inter-Integrated Circuit), el Arduino envía comandos al sensor MPU-6050 para solicitar datos de los registros específicos. Luego, lee los datos recibidos y los almacena en variables para su posterior procesamiento o visualización.

Una vez finalizado el propósito de Arduino procedemos a Matlab para la representación gráfica en la toma de muestra de gases arteriales radiales, durante este proceso, se emplea una jeringa que debe mantenerse a un ángulo de 45 a 60 grados para garantizar su correcta realización. Los componentes de acelerómetro y giroscopio del sensor permiten la lectura del ángulo de inclinación al aprovechar los grados de libertad del objeto, es importante identificar el eje de movimiento relevante y determinar la lectura correspondiente en cada caso.

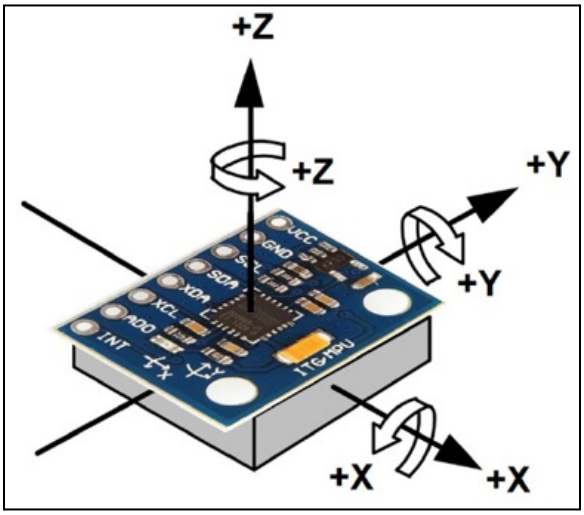

Figura 12. Sensor MPU 6050

Se buscan los datos de Pitch (cabeceo), Yaw (giro) y Roll (balanceo) como se muestra en la imagen por medio de comunicación serial. En MATLAB, cuando se establece una conexión serial para comunicarse con un dispositivo a través de un puerto específico, se crea un objeto serial. Este objeto serial representa la conexión y permite enviar y recibir datos a través de ella, para esto el primer paso es cerrar y eliminar alguna conexión existente o anterior permitiendo que el puerto esté disponible para utilizarlo, leer el puerto serial para esto usamos comandos como ***fopen*** o ***fclose*** para abrir y cerrar el puerto según se requiera, hay que tener muy en cuenta varios factores, sin embargo son dos fundamentales; el primero es en donde esta conectado el sensor ya que el orden de las conexiones puede

cambiar en el caso que tengamos varias cosas conectadas (en nuestro caso es ***COM4***) y la segunda cosa a tener en cuenta es la velocidad de comunicación en bits por segundo ya que si no es la misma en que trabajamos anteriormente se llega a perder información(***BaudRate*** en nuestro caso utilizamos ***9600***).

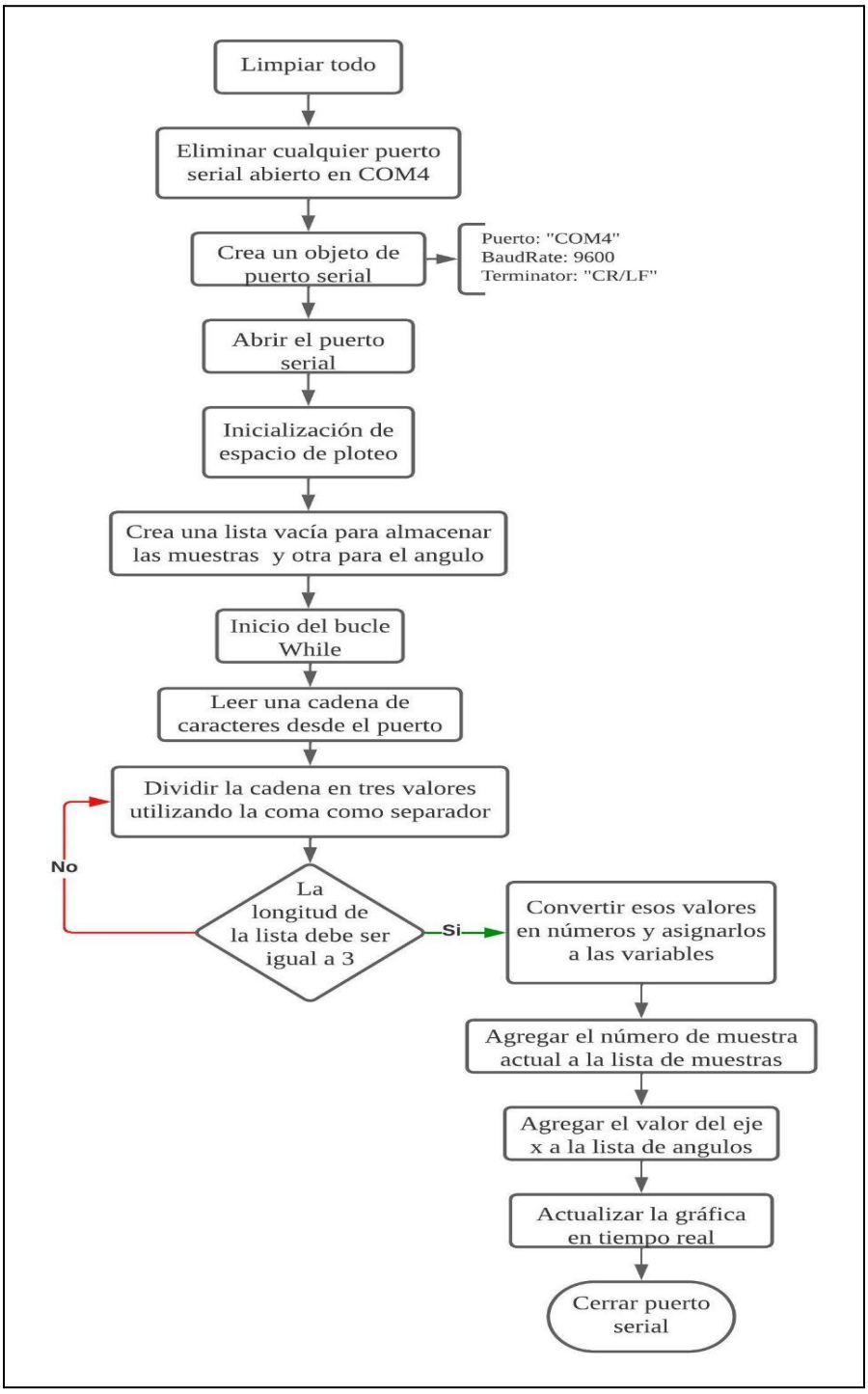

Figura 13. Diagrama de flujo código en Matlab

Dentro de un bucle ***While***, se lee continuamente la información recibida del sensor a través del puerto serial, se procesan los datos y se almacenan en listas anteriormente creadas correspondientes a las muestras y los ángulos de inclinación en el eje x. Y para finalizar, se actualiza la gráfica en tiempo real con los nuevos datos, lo que permite visualizar de forma dinámica los cambios en el ángulo de inclinación. El bucle continúa ejecutándose hasta que se detenga manualmente la simulación, una vez finalizado se cierra la conexión con el puerto serie.

En resumen, este código implementa un sistema de monitoreo en tiempo real del ángulo de inclinación utilizando el sensor MPU6050 que visualizando los datos en una gráfica proporciona una herramienta útil para controlar y analizar el ángulo de inclinación de forma precisa y más segura,lo cuál llega a ser fundamental para el proceso además de ser fácilmente integrable a la metodología docente gracias a su capacidad de adaptación.

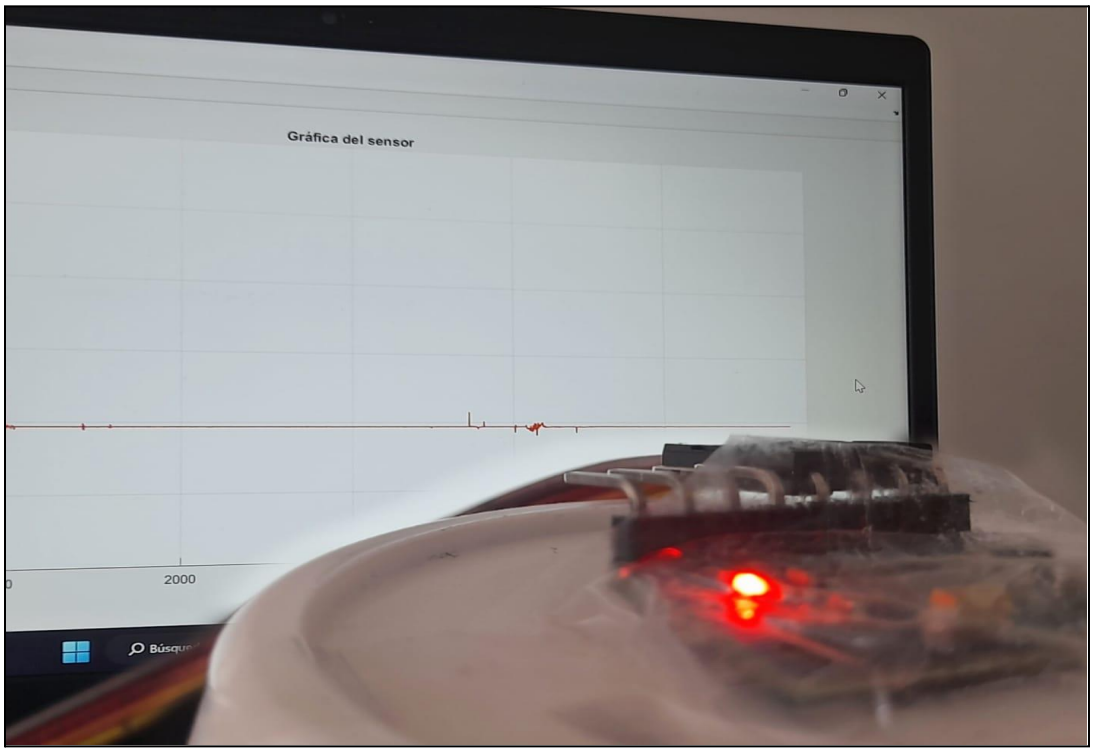

Figura 14. Sensor fijo con gráfica en tiempo real

Al mantener una superficie plana y recta como referencia, se garantiza que los datos obtenidos sean precisos y confiables. Esto es fundamental en aplicaciones donde la información del ángulo de inclinación juega un papel crucial, como la navegación, el control de movimientos o la estabilización de objetos.

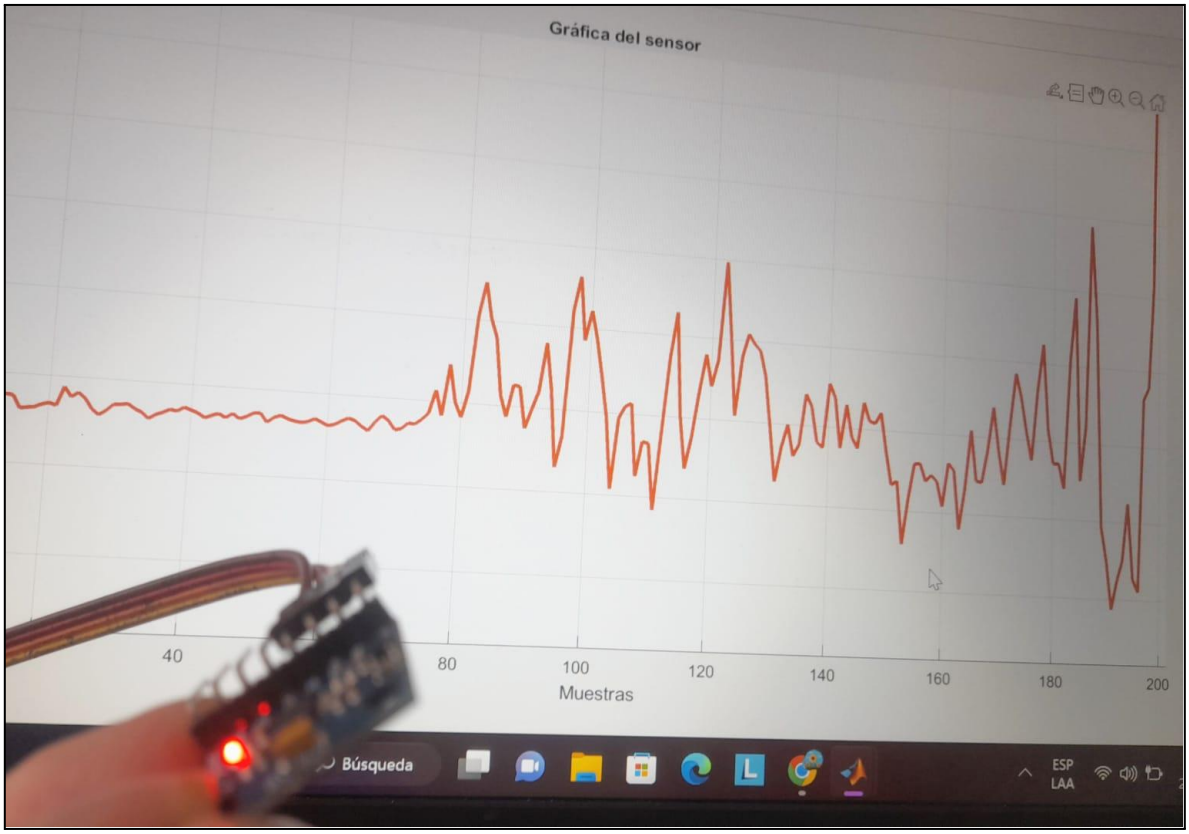

Figura 15. Gráfica en tiempo real del movimiento

Al no limitar el movimiento del sensor, se obtiene una representación visual más completa de su comportamiento. La gráfica en movimiento proporciona una representación visual más dinámica y detallada de los datos capturados por el sensor, esto permite una mejor comprensión de cómo el objeto o el sensor se mueve en el espacio, así como la detección de patrones o tendencias en el movimiento.

12. RECURSOS

INFORMACIÓN DE LA COMPRA				
Cantidad	**Concepto**	**Valor Unitario**	**fuente de financiación**	**Observaciones**
2	microcontrolador	$ 134.784,16	Convocatoria	
2	Shield Joystick Análogo Con Botonera	$18.802,00	Convocatoria	
2	CABLE PLANO RIBBON MULTICOLOR 14 HILOS 1 METRO	$6.683,84	Convocatoria	1 Metros
1	Celular	$699.900,00	Convocatoria	ANDROID
1	Gafas de Realidad Virtual	$2.500.000,00	Convocatoria	
	papelería documento final (proyecto de grado)	$50.000,00	Estudiante	
1	memoria usb 32Gb	$50.000,00	Estudiante	
1	Módulo Acelerómetro Giroscopio MPU6050	$16.500,00	Estudiante	
1	Brazo sintético	----------	Enfermería	
1	piel sintética	----------	Enfermería	
1	insumos de enfermeria	----------	Enfermería	
TOTAL		$3’341.885,84		

Tabla 1. Recursos invertidos para llevar a cabo el proyecto

13. CRONOGRAMA

DESCRIPCIÓN DE LA ACTIVIDAD	Jun	Jul				Octubre				Nov	2023			
	S.4	S.1	S.2	S.3	S.4	S.1	S.2	S.3	S.4		Feb	Mar	Abr	May
Primeras interacciones con el programa, realizar tutoriales recomendados														
Creación de objetos 3D														
Creación de escenarios														
Implementación de las interacción en los escenarios (Scripts)														
Mejora de objetos y escenarios para el producto final														

Tabla 2. Cronograma para el simulador de Unity

DESCRIPCIÓN DE LA ACTIVIDAD	Noviembre				Diciembre				Enero			
	S.1	S.2	S.3	S.4	S.1	S.2	S.3	S.4	S.1	S.2	S.3	S.4
Acercamiento a herramientas virtuales y elección de materiales a utilizar												
Creación del código en Arduino para el acelerómetro y giroscopio MPU6050												
Creación de código de Matlab que permite la visualización												
Implementación y pruebas de Matlab en conjunto con Arduino												
Mejoras y correcciones del entorno y la gráfica para el fin del proyecto												

Tabla 3. Cronograma para modelamiento en Matlab

14. CONCLUSIONES

Unity es ampliamente considerado como la mejor opción para trabajar en el desarrollo de experiencias de realidad virtual debido a su compatibilidad, su motor gráfico, facilidad de uso y disponibilidad de recursos como Probuilder que permiten un complemento en el desarrollo de estos entornos. Además de su versatilidad y capacidad para crear experiencias inmersivas y de alta calidad lo que lo convierte en una herramienta destacada en el campo de la realidad virtual.

El diseño del simulador permite a los profesionales evaluar su conocimiento e identificar áreas de mejora, poder desarrollar un razonamiento crítico multitareas (AHP) en relación con este procedimiento de la toma de muestra de gases arteriales radiales

El objetivo inicial de modelar el ángulo de inclinación de la jeringa en Matlab queda como propuesta para dar continuidad al desarrollo del proyecto, teniendo en cuenta el adquirir los datos y generar una gráfica en tiempo real los cuales proporcionarán beneficios significativos en términos de visualización, análisis y seguimiento del proceso de toma de muestra de gases arteriales radiales. Estos pasos iniciales establecen una base sólida para avanzar en el proceso de adopción y alcanzar los objetivos deseados con la nueva tecnología.

15. REFERENCIAS

[1] C. Moreira Segura y B. Delgadillo Espinoza. "La virtualidad en los procesos educativos: Reflexiones teóricas sobre su implementación". SciELO - Scientific Electronic Library Online. https://www.scielo.sa.cr/pdf/tem/v28n1/0379-3982-tem-28-01-00121.pdf.

[2] Y. Liu, C. M. Eckert y C. Earl. "Una revisión de los métodos AHP difusos para la toma de decisiones con juicios subjetivos. sistemas expertos con aplicaciones." ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S0957417420305625?via=ihub.

[3] E. K. Parra Díaz y D. E. Vargas Buitrago. "Guía de cuidado interactiva humanizada para la toma de gases arteriales basados en la teoría de Swanson para estudiantes de la facultad de enfermería de la Universidad Nacional de Colombia". Repositorio Universidad Nacional. https://repositorio.unal.edu.co/handle/unal/58928.

[4] C. A. Niño Herrera. "Fortalecimiento de la simulación clínica como herramienta pedagógica en enfermería: Experiencia de internado". SciELO - Scientific Electronic Library Online. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S2216-09732015000100013.

[5] "Qué es Unity y para qué sirve". MasterD: Oposiciones, Cursos y FP. https://www.masterd.es/blog/que-es-unity-3d-tutorial.

[6] "Plataforma de desarrollo en tiempo real de Unity | Motor de VR, AR, 3D y 2D". Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine. https://unity.com/es.

[7] "Oculus Quest 2: Características, juegos y más sobre las gafas de realidad virtual". ▷ Hiraoka | Encuentra las mejores ofertas AQUÍ | 100% garantía. https://hiraoka.com.pe/blog/post/oculus-quest-2-caracteristicas-juegos-y-mas-sobre-las-gafas-de-realidad-virtual#:~:text=Oculus%20Quest%202%20son%20unas,a%20una%20computadora%20para%20funcionar.

[8] "Meta Quest 2: Our most advanced new all-in-one VR headset | Oculus". Meta – Shop VR headsets and smart glasses. https://www.meta.com/quest/products/quest-2/tech-specs/#tech-specs.

[9] "AdbLink". Softonic. https://adblink.softonic.com/#:~:text=adbLink%20es%20un%20programa%20complementario,u%20ordenadores%20con%20Microsoft%20Windows.

ANEXOS

1) Código Unity - Contador de objetos

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class Objetos : MonoBehaviour
{
   int score = 0;
   public Text puntaje;
   int TObj = 9;
   public GameObject SIGUIENTE;
   void OnTriggerEnter(Collider other)
   {
      if(other.CompareTag("Bandeja"))
      {
         score++;
         puntaje.text = "MARCADOR: " + score + " / " + TObj;
      }
      if(score == TObj)
      {
         SIGUIENTE.SetActive(true);
      }
   }
}
```

2) Código Unity - Interfaz de Preguntas

```
[System.Serializable]
public class QuestionsAndAnswer
{
   public string Question;
   public string[] Answer;
   public int CorrectAnswer;
}
```

```
////////////////////////////////////////////////////////////////////////////////
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
```

```
public class QuizManager : MonoBehaviour
{
    public List<QuestionsAndAnswer> QnA;
    public GameObject[] options;
    public int currentQuestion;

    public GameObject Pregunta;
    public GameObject GOPanel;
    public GameObject Siguiente;

    public Text QuestionTxt;
    public Text ScoreTxt;

    int totalQuestions = 0;
    public int score;

    private void Start()
    {
        totalQuestions = QnA.Count;
        GOPanel.SetActive(false);
        Siguiente.SetActive(false);
        generateQuestion();
    }

    public void RETRY()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    void GameOver()
    {
        Pregunta.SetActive(false);
        GOPanel.SetActive(true);
        ScoreTxt.text = score + "/" + totalQuestions;

        if(score != totalQuestions)
        {
            Siguiente.SetActive(false);
        }
        else
        {
            Siguiente.SetActive(true);
        }
    }
```

```
    public void correct()
    {
        score += 1;
        QnA.RemoveAt(currentQuestion);
        generateQuestion();
    }

    public void wrong()
    {
        QnA.RemoveAt(currentQuestion);
        generateQuestion();
    }

    void SetAnswer()
    {
        for (var i = 0; i < options.Length; i++)
        {
            options[i].GetComponent<AnswerScript>().isCorrect = false;
            options[i].transform.GetChild(0).GetComponent<Text>().text =
QnA[currentQuestion].Answer[i];

            if(QnA[currentQuestion].CorrectAnswer == i+1)
            {
                options[i].GetComponent<AnswerScript>().isCorrect = true;
            }
        }
    }

    void generateQuestion()
    {
        if(QnA.Count > 0)
        {
            currentQuestion = Random.Range(0, QnA.Count);
            QuestionTxt.text = QnA[currentQuestion].Question;
            SetAnswer();
        }
        else
        {
            Debug.Log("Out ot Questions");
            GameOver();
        }
    }
}
```

```
//////////////////////////////////////////////////////////////////////////////////
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AnswerScript : MonoBehaviour
{
    public bool isCorrect = false;
    public QuizManager quizManager;

    public void Answer()
    {
        if(isCorrect)
        {
            Debug.Log("Correct Answer");
            quizManager.correct();
        }
        else
        {
            Debug.Log("Wrong Answer");
            quizManager.wrong();
        }
    }
}
```

3) Código Unity - Interfaz de Video

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Video;
using UnityEngine.EventSystems;

public class SliderVideo : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    public GameObject botonPlay;
    public GameObject botonPausa;
    public GameObject botonReiniciar;
    public GameObject botonVolumne;
    public GameObject botonMute;
    public bool play;
```

```
public bool termino;
public VideoPlayer videoPlayer;
public Slider barra;
public bool slide;
public AudioSource audioSource;
public Slider barraVolumen;

void Start()
{
   botonReiniciar.SetActive(false);
   videoPlayer.Play();
   videoPlayer.Pause();
}

public void OnPointerDown(PointerEventData a)
{
   videoPlayer.Pause();
   slide = true;
}

public void OnPointerUp(PointerEventData a)
{
   if(play)
   {
      videoPlayer.Play();
   }
   else
   {
      videoPlayer.Pause();
   }
   float frame = (float)barra.value*(float)videoPlayer.frameCount;
   videoPlayer.frame = (long)frame;
   slide = false;
}

void Update()
{
   if(slide==false)
   {
      barra.value = (float)videoPlayer.frame / (float)videoPlayer.frameCount;
   }

   if(barra.value >= 0.99 && termino == false)
   {
```

```
        botonReiniciar.SetActive(true);
        botonPlay.SetActive(false);
        botonPausa.SetActive(false);
        termino = true;
    }

    if(barra.value == 0)
    {
        termino = false;
    }
}

public void Volumen()
{
    audioSource.volume = barraVolumen.value;
}

public void Play()
{
    videoPlayer.Play();
    botonPlay.SetActive(false);
    botonPausa.SetActive(true);
    play = true;
}

public void Pausa()
{
    videoPlayer.Pause();
    botonPlay.SetActive(true);
    botonPausa.SetActive(false);
    play = false;
}

public void Reiniciar()
{
    videoPlayer.frame = 0;
    Play();
    botonReiniciar.SetActive(false);
}

public void Silenciar()
{
    if(audioSource.mute)
    {
```

```
            botonMute.SetActive(false);
            botonVolumne.SetActive(true);
            audioSource.mute = false;
        }
        else
        {
            botonMute.SetActive(true);
            botonVolumne.SetActive(false);
            audioSource.mute = true;
        }
    }
}
```

4) Código Arduino

```
#include &quot;Wire.h&quot; // This library allows you to communicate with I2C
devices.
const int MPU_ADDR = 0x68; // I2C address of the MPU-6050. If AD0 pin is set
to HIGH, the I2C address will be 0x69.
int16_t accelerometer_x, accelerometer_y, accelerometer_z; // variables for
accelerometer raw data
int16_t gyro_x, gyro_y, gyro_z; // variables for gyro raw data
int16_t temperature; // variables for temperature data
char tmp_str[7]; // temporary variable used in convert function
char* convert_int16_to_str(int16_t i) { // converts int16 to string.
Moreover, resulting strings will have the same length in the debug monitor.
  sprintf(tmp_str, &quot;%6d&quot;, i);
  return tmp_str;
}
void setup() {
  Serial.begin(9600);
  Wire.begin();
  Wire.beginTransmission(MPU_ADDR); // Begins a transmission to the I2C
slave (GY-521 board)
```

```
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
}
void loop() {
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H) [MPU-6000
and MPU-6050 Register Map and Descriptions Revision 4.2, p.40]
  Wire.endTransmission(false); // the parameter indicates that the Arduino
will send a restart. As a result, the connection is kept active.
  Wire.requestFrom(MPU_ADDR, 7*2, true); // request a total of 7*2=14
registers

  // &quot;Wire.read()&lt;&lt;8 | Wire.read();&quot; means two registers are read and
stored
in the same variable

  accelerometer_x = Wire.read()&lt;&lt;8 | Wire.read(); // reading registers: 0x3B
(ACCEL_XOUT_H) and 0x3C (ACCEL_XOUT_L)
  accelerometer_y = Wire.read()&lt;&lt;8 | Wire.read(); // reading registers: 0x3D
(ACCEL_YOUT_H) and 0x3E (ACCEL_YOUT_L)
  accelerometer_z = Wire.read()&lt;&lt;8 | Wire.read(); // reading registers: 0x3F
(ACCEL_ZOUT_H) and 0x40 (ACCEL_ZOUT_L)
  temperature = Wire.read()&lt;&lt;8 | Wire.read(); // reading registers: 0x41
(TEMP_OUT_H) and 0x42 (TEMP_OUT_L)
  gyro_x = Wire.read()&lt;&lt;8 | Wire.read(); // reading registers: 0x43
(GYRO_XOUT_H) and 0x44 (GYRO_XOUT_L)
  gyro_y = Wire.read()&lt;&lt;8 | Wire.read(); // reading registers: 0x45
(GYRO_YOUT_H) and 0x46 (GYRO_YOUT_L)
```

```
  gyro_z = Wire.read()&lt;&lt;8 | Wire.read(); // reading registers: 0x47
(GYRO_ZOUT_H) and 0x48 (GYRO_ZOUT_L)

  //print out data
  //Serial.print(convert_int16_to_str(accelerometer_x));Serial.print(&quot;,&quot;);
  //Serial.print(convert_int16_to_str(accelerometer_y));Serial.print(&quot;,&quot;);
  //Serial.print(convert_int16_to_str(accelerometer_z));
  // the following equation was taken from the documentation [MPU-6000/MPU-
6050 Register Map and Description, p.30]
  //Serial.print(temperature/340.00+36.53);
  Serial.print(convert_int16_to_str(gyro_z));Serial.print(&quot;,&quot;);
  Serial.print(convert_int16_to_str(gyro_y)); Serial.print(&quot;,&quot;);
  Serial.print(convert_int16_to_str(gyro_x));
  Serial.println();

  delay(20);
}
```

5) Código Matlab

```
clear all
close all;
clc;
delete(instrfind({'Port'},{'COM4'}));
puerto_serial=serial('COM4','BaudRate',9600,'Terminator','CR/LF');
warning('off','MATLAB:serial:fscanf:unsuccessfulRead');
fopen(puerto_serial);
fwrite(puerto_serial,'a')
i=1;
figure;
```

```
l1 = line(nan, nan, ‘Color’, ‘r’, ‘LineWidth’, 2);
xlabel(‘Muestras’);
ylabel(‘Ángulo de inclinación’);
title(‘Gráfica del sensor’);
grid on;
muestras = [];
angulos_x = [];
while (i)
data= fscanf(puerto_serial,’%s’);
data = split(data,’,’)
if length(data) == 3
z_val = str2num(data{1});
y_val = str2num(data{2});
x_val = str2num(data{3});
muestras = [muestras numel(muestras)+1];
angulos_x = [angulos_x x_val];
set(l1, ‘YData’, angulos_x, ‘XData’, muestras);
drawnow;
end
end
fclose(puerto_serial);
```