UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Sistema de Sensores Inerciales para Análisis Biomecánico

.

Karen Elizabeth Rodríguez Zapata

Ingeniería Mecánica

Trabajo de integración curricular presentado como requisito
para la obtención del título de
Ingeniera Mecánica

Quito, 06 de diciembre de 2019

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE CIENCIAS E INGENIERÍA

HOJA DE CALIFICACIÓN
DE TRABAJO DE INTEGRACIÓN CURRICULAR

Sistema de Sensores Inerciales para Análisis Biomecánico

Karen Elizabeth Rodríguez Zapata

Calificación:	**(puntos logrados) / (puntos posibles)**
Nombre del profesor, Título académico	**Paul Arauz, Ph.D.**
Firma del profesor:	**_____**
Nombre del profesor, Título académico	**Carlos Andrade, Ing.**
Firma del profesor:	**_____**
Nombre del profesor, Título académico	**Alfredo Valarezo, Ph.D.**
Firma del profesor:	**_____**

Quito, 06 de diciembre del 2019

Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Karen Elizabeth Rodríguez Zapata

Código: 00118119

Cédula de identidad: 1717423295

Lugar y fecha: Quito, diciembre de 2019

RESUMEN

En el presente trabajo de titulación se realiza el diseño e implementación de un prototipo inercial de bajo costo que tiene como objetivo la captura del movimiento biomecánico de un sistema vivo. El sistema de sensores debe cumplir con los siguientes requerimientos: portátil, móvil, que entregue datos en tres ejes con 6 grados de libertad y con conexión Wireless. Los sensores individuales deben ser adaptables a diferentes personas, con buena capacidad de generalización, sin sufrir el fenómeno de oclusión de datos que se presenta en otros sistemas. Se empieza con el desarrollo del prototipo del sistema de sensores para determinar si es funcional y de bajo costo. El sistema consta una placa maestra y tres sensores esclavos individuales, en donde el sensor de inercial individual consta de una unidad de medición inercial *MPU 6050,* un microprocesador y una batería.

Mediante programación se obtiene la información de cada uno de los tres sensores, la cual pasa a la placa madre teniendo resultados de aceleración y velocidad angular. Esta información se procesa en la aplicación Processing, donde se realiza dos tipos de métodos para tener el resultado de posición relativa. El primer método es la doble integración de la aceleración la cual se realiza por métodos numéricos, dando como resultados los datos de la velocidad y la posición de los tres sensores. El segundo método, se obtiene las coordenadas X Y de los tres sensores basado en los ángulos *pitch y roll,* y la longitud de cada elemento donde se encuentra colocado cada sensor. El mismo programa realiza las animaciones de los dos métodos, teniendo como resultado que el segundo método representa de mejor manera el movimiento realizado en tiempo real.

Palabras Clave: *sensor de inercia, IMU, captura de movimiento, desarrollo e implementación, posición, MPU 6050, wifi.*

ABSTRACT

In the present work, a low-cost inertial prototype that aims to capture the biomechanical movement of a living system is designed and implemented. The sensor must be portable, mobile, providing data 3 axes with 6 degrees of freedom and with wireless connection. The individual device must be adaptable to different people, with good biological generalization capacity, without suffering the phenomenon of occlusion of marker which is present in other systems. The system includes a master receiver and three slave inertial sensors. The inertial sensor consists of an inertial measurement unit, microprocessor and battery. The prototype of the sensor system is developed to be functional, and low cost.

A simple program was written to obtain the information of each of the three sensors and pass it to the master board. Results of acceleration and angular velocity is processed in the software "Processing", where two types of methods are performed to have the relative position result. The first method is the double integration of acceleration by numerical methods, resulting in the speed and position data of the three sensors. The second method calculates the X Y coordinates of the three sensors based on the pitch and roll angles, and the length of the elements where the sensors are placed. The same program generates the animations of both methods, resulting in the second method better representing the real time movement.

Keywords: *inertia sensor, IMU, motion capture, development, position, MPU 6050, wifi*

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

La biomecánica es un área de conocimiento que emplea los principios de la mecánica, la cual estudia el efecto de la energía y de las fuerzas de la materia en sistemas vivos para modelarlos, estudiar su comportamiento y resolver problemas derivados de diversas condiciones a la que un cuerpo puede ser sometido. A pesar del avance de estas aplicaciones en países desarrollados, aún la comprensión en el Ecuador acerca de los procesos biomecánicos es limitado, existen tan solo 25 estudios sobre análisis biomecánicos a nivel nacional con respecto al estudio detallado del gesto de un deportista. (Villa, Díaz, & Urgilés, 2008). Con el tiempo y a través de la investigación e inversión en el desarrollo científico, se puede avanzar en el conocimiento de la función, movimiento, desempeño y desarrollo de los principios de la biomecánica.

El presente trabajo de titulación, se estudiará el desarrollo y la implementación de un sensor inercial de medición para análisis biomecánico; donde se ha procedido al planteamiento de tres objetivos principales, los cuales son: selección de materiales adecuados y óptimos para el dispositivo, diseñar el sensor tanto eléctrico como mecánico, programación del dispositivo para poder tomar las mediciones, y por último, realizar una observación para opción a mejora u optimización del sistema inercial.

En el año 2016, Ecuador implementó un laboratorio biomecánico para deportistas de alto rendimiento, sin embargo, el acceso se encuentra limitado y altamente restringido (Universo, 2016); de manera adicional se resalta el costo elevado de los equipos que individualmente conforman el laboratorio y el largo tiempo que implica realizar la importación de los mismos. El acceso limitado a los beneficios brindados por estos equipos causa dificultad para realizar investigación a nivel académico, consecuentemente impide un

desarrollo considerable en el ámbito de la biomecánica, ergonomía, animación 3D, diseño, al igual de las aplicaciones en el campo de la medicina en nuestro país. En la actualidad existen sistemas avanzados para realizar el análisis de movimiento, sin embargo, los laboratorios para este tipo de investigación en el país son escasos, algunos restringidos y de alto costos como antes se menciona. (Flores & Sandoval-gonzalez, 2015).

Actualmente en el mercado se dispone de varios sistemas que se encuentran disponibles con una clasificación especifica en cuanto a captura de movimiento. Este tipo de tecnología se clasifica en este tipo de sensores: ópticos, inerciales y magnéticos; siendo el sensor óptico el sistema operatorio más utilizado en cuanto a investigación y publicaciones, por ser la pionera en captura de movimiento, un ejemplo claro de este sistema, es el Mocap que se basa en la captura de imágenes 2D tomadas con varias cámaras para encontrar un modelo 3D. (Gomez et al., 2018)

Figura 1:Sistema de captura de movimiento MoCap

Sin embargo, en la actualidad se ha desarrollado sensores de medidas inerciales, los cuales son pequeños, fáciles de usar y móviles. Este sistema inercial tiene un gran auge debido a la combinación de diferentes sensores en una sola unidad de medición inercial

(IMU), en comparación con el primer sistema antes mencionado. (Roetenberg, Luinge, & Slycke, 2009a). XSENS es una empresa considerada innovadora líder en tecnología y productos de seguimiento de movimiento 3D, con su fusión de sensores que permiten una interacción perfecta entre el mundo físico y el digital en dispositivos de consumo y aplicaciones profesionales como animación en 3D en el mundo de entretenimiento digital, análisis de movimiento y control, estabilización industrial y en la medicina (Roetenberg, Luinge, & Slycke, 2009). La mayoría de tecnología utilizada para la fabricación de productos se basa en el sistema inercial, por su facilidad de uso y su baja oclusión de datos en el análisis biomecánico de un sistema vivo.

Figura 2: Sistema de captura de movimiento inercial. (Xsens, 2018

En el caso de la medicina y en el deporte, estos dispositivos se han implementado para el beneficio de las personas donde les permite tener una mejor calidad de vida como indica el estudio de (Cuervo, Olaya, & Salamanca, 2013) en el área médica, donde un sistema inercial fue aplicado en niños, donde se implementó una plataforma de realidad virtual que permitió que niños mayores de cinco años realicen terapias de recuperación en los miembros

superiores, basado en juegos con el uso de sensores, tensores, detectores y unidades de sistema inercial. En el deporte se utiliza para realizar exámenes de medición de gesto deportivo, donde los movimientos tienen que estar acorde al tipo de deporte que se practica, con el objetivo de realizar un estudio detallado del gesto deportivo en todas las fases del mismo, demostrando que en conjunto con profesionales entrenadores del deporte es necesario el desarrollo de herramientas de ingeniería para garantizar el éxito de un plan de trabajo y entrenamiento, alcanzando el éxito en cada uno de los deportistas (Villa et al., 2008).

Los sensores inerciales, también llamados unidades de medición inercial (IMU), son sensores combinados por como acelerómetros, giróscopos y magnetómetros. La aplicación de estos dispositivos inició su auge a principios de los años 20´s principalmente en aplicaciones de navegación, dirección y control en aeronaves, barcos y dispositivos guiados automáticamente como misiles, por ejemplo, en esa época se trataba con dispositivos grandes, pesados, de alto consumo de energía y sobre todo de alto costo económico. En 1979 la universidad de Stanford presentó el primer sensor inercial denominado sistema micro electromecánico (MEMS), y es el que ha permitido el desarrollo continuo de este tipo de dispositivos y que ya en la actualidad es posible conseguir sensores inerciales con sensibilidad totalmente aptos para ser implementados en aplicaciones médicas donde se requieren equipos ligeros de fácil uso y que no obstruya o altere los movimientos naturales. (Martínez-Méndez, Portillo-Rodriguez, Romero-Huertas, & Vilchis-González, 2012).

Los sensores de inercia permiten que no exista oclusión de datos al momento en el que un humano o sistema vivo realice ciertos movimientos o actividades, estos dispositivos inerciales requieren menor tiempo al momento de ser colocados y pueden ser usados directamente en el cuerpo en segmentos específicos, deduciendo la cinemática de las

articulaciones. Los sensores deben ser adaptables a diferentes personas, es decir con buena capacidad de generalización, para esto es necesario incorporar restricciones para mejorar la precisión de captura y reducir el espacio de algoritmos. (Xu, He, Zhang, Yao, & Tseng, 2018)

En Ecuador, existe escasa investigación con dispositivos inerciales y un alto porcentaje de publicaciones de captura de movimiento son realizadas con tecnología óptica Microsoft Kinect en el ámbito de diseño, animación 3D y deporte. Esta tecnología consta de un dispositivo sensorial que se conecta a la consola Xbox One en conjunto con una aplicación llamada 3D Scan que permite digitalizar un objeto, mientras el dispositivo Kinect se sostiene en las manos girando alrededor de él (Microsoft, 2019), sin embargo, los resultados con este sistema no son exactos a diferencia de los datos que nos brinda la tecnología inercial.

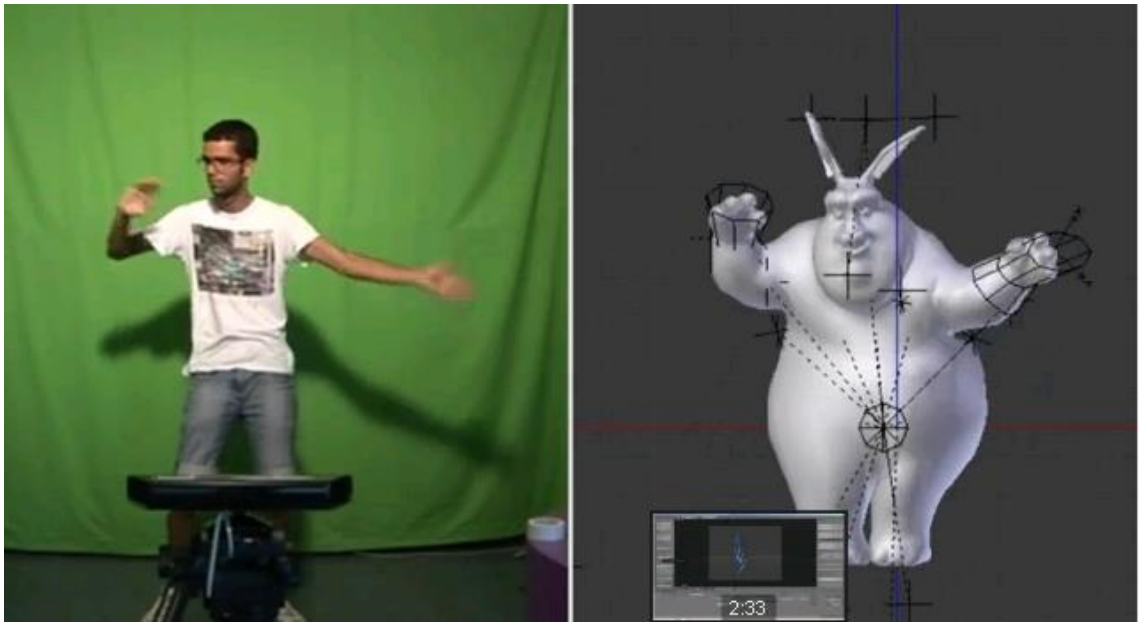

Figura 3:Sistema de captura de movimiento Kinect

Esto permite que mediante este desarrollo de sensores inerciales se dé a conocer más esta tecnología, captar un segmento de mercado y sobre todo que se expanda hacia investigación académica.

Para el desarrollo e implementación de un sensor de inercia de bajo costo, es necesaria la rama de ingeniería electrónica y mecánica, dado que, mediante la fusión de los conocimientos en estas dos áreas, permitirán que se desarrolle el dispositivo para capturar el movimiento del cuerpo humano o de un ser viviente sin la necesidad de otros sistemas que cuenta con cámaras, en este caso sistema óptico o emisores externos.

Para una estimación precisa y libre de desviación de la orientación, se sabe de varios métodos que combinan las señales de giroscopios, acelerómetros y magnetómetros 3D, por lo cual para el desarrollo e implementación de este dispositivo de bajo costo es necesario que el sistema se conforme por: una unidad de medición inercial, microprocesadores y baterías. La fusión de los sensores se utiliza para estimar el estado de un sistema a partir de los datos recopilados, utilizando dos o más sensores para poder obtener la cinemática y forma del cuerpo en movimiento (Santos et al., 2019).

Los sensores IMU, normalmente deben ser colocados directamente en el cuerpo vivo, teniendo una recopilación directa de datos cinemáticos de cada una de las partes del cuerpo para realizar el análisis del movimiento. Estos sensores inerciales tienen la ventaja de identificar el movimiento del cuerpo en una amplia variedad de entornos, sin la necesidad de tener en cuenta la distancia de la cámara, el rango de captura, resolución o la velocidad de los fotogramas como en otros sistemas utilizados para la recolección de este tipo de información. Este sistema de sensores inerciales tiene la capacidad de obtener información de diferencias anatómicas originales, magnitud y orientación de segmentos corporales (Ong, Seet, Khoo, & Noroozi, 2018).

El principal desafío en el desarrollo de este tipo de sistema es la traducción de datos de velocidad y aceleración, en datos tridimensionales como ángulos y posición en el espacio 3D. En teoría, al realizar una única integración de la velocidad angular y una doble

integración de aceleración nos permitiría obtener la orientación y la posición de un sensor inercial, siendo cada sensor montado en cada una de las partes del cuerpo humano (Tadano, Takeda, & Miyagawa, 2013).

Con estos antecedentes, a lo largo de esta investigación el objetivo consistirá en el desarrollo y cuantificar cual es la capacidad del sistema de sensores inerciales. Al momento de implementarlo se valorará que el sistema sea: rentable, de análisis de movimiento humano, basado en unidades de medida inerciales, con fácil portabilidad para evaluar cualitativa y cuantitativamente las articulaciones del cuerpo humano con la función de seguimiento gráfico 3D y sus datos cinemáticos, como la aceleración, la velocidad angular, la orientación (ángulo) y la posición (Ong et al., 2018). El presente trabajo es un estudio preliminar sobre un enfoque de extracción de datos de orientación tridimensional utilizando la unidad inercial IMU, de donde se obtiene los datos de los sensores integrados en esta unidad, los cuales son: acelerómetros y giroscopios. Existen varios métodos para poder traducir los datos de aceleración y velocidad en datos en tres dimensiones, para esto se escogerá el método óptimo que permitirá que los datos se acerquen más a una medición más precisa.

2. METODOLOGÍA

Se implementará tres sensores, que se encuentren conectados inalámbricamente vía wifi para obtener información de posición relativa de un mecanismo vivo. Una vez recolectada la información con los sensores, se envía a una computadora donde se procesa los resultados transformándolos en animación de la captura de movimiento, se procederá a realizar dos métodos para la obtención de la posición relativa.

En el primer método, se procederá a realizar la doble integral de la aceleración dada por el MPU 6050 que es un valor absoluto. El segundo método es obtener las coordenadas X Y, mediante ángulos pitch y row en dos dimensiones. Los materiales a utilizar en los dos métodos no tienen ninguna variación, solo se procederá a realizar variaciones en la programación. Por lo cual los principales materiales para la construcción el sistema de red inalámbrica de sensores corporales (WBSN), serán: un microprocesador que utiliza un mecanismo de comunicación unificado y simplificado, y una unidad de medida inercial MPU6050 (Fei, Song, Xu, & Sun, 2014).

2.1.Materiales

En la tabla 1, se procede a enlistar los materiales utilizados para el desarrollo del sistema de sensores corporales, se describe cuantas unidades de cada uno de los componentes se utiliza, el costo unitario y total.

Tabla 1: Materiales y Presupuesto

Materiales	Unidad	Costo Unitario	Costo total
ESP-32 WROOM 32	3	$ 10,00	$ 30,00
ESP-32 WROVER Kit	1	$ 55,00	$ 55,00

MPU 6050	3	$ 6,50	$ 19,50
Batería	4	$ 5,00	$ 20,00
Cables	1	$ 5,00	$ 5,00
Total	12	81,5	129,5

Aparte de los elementos enlistados se procederá a utilizar dos softwares, el primero es ARDUINO IDE y PROCESSING. El primero permitirá programar la conexión wifi y obtener los datos de posición y aceleración, el segundo, permitirá realizar la animación en tiempo real del movimiento que capturan los sensores.

En la tabla 2, se proporciona la función y la especificación de los componentes principales y esenciales como: ESP32- WROOM-32 y MPU6050.

Tabla 2: Especificaciones técnicas de los componentes principales.

Componentes	Función	Especificaciones
ESP32-WROOM-32	Microprocesador integrado con comunicación wifi y bluetooth.	802.11 b / g / n (802.11n hasta 150 Mbps) Agregación A-MPDU y A-MSDU y soporte de intervalo de protección de 0.4 µs Rango de frecuencia 2.4 GHz ~ 2.5 GHz Protocolos Bluetooth v4.2 BR / EDR y especificación BLE Radio: receptor NZIF con sensibilidad de –97 dBm Transmisor de clase 1, clase 2 y clase 3 AFH (Systems, 2019)
MPU 6050	Chip único integrado con acelerómetro de tres ejes y giroscopio de tres ejes	Salida digital de 6 ejes. Giroscopio con sensibilidad de ±250, ±500, ±1000, y ±2000dps Acelerómetro con sensibilidad de ±2g, ±4g, ±8g y ±16g Algoritmos embebidos para calibración Sensor de temperatura digital Entrada digital de video FSYNC Interrupciones programables Voltaje de alimentación: 2.37 a 3.46V Voltaje lógico: 1.8V±5% o VDD 10000g tolerancia de aceleración máxima. Dimensiones: 4x4x0.9 mm (Ave, Number, & Date, 2013)

2.2.Descripción de los elementos principales

ESP32 es un chip combinado de Bluetooth y Wi-Fi de 2.4 GHz de ultra baja potencia, posee versatilidad y confiabilidad en una amplia variedad de aplicaciones y diferentes perfiles de potencia. Este microprocesador, integra el interruptor de antena, amplificador de potencia, amplificador de recepción de bajo ruido, filtros y módulos de administración de energía. También utiliza CMOS para radio y banda base totalmente integrados de un solo chip, y también integra circuitos de calibración avanzados que permiten que la solución se ajuste dinámicamente eliminando imperfecciones de circuitos externos o adaptarse a cambios en condiciones externas. No requiere equipos de prueba de Wi-Fi caros y especializados. (Systems, 2019)

Por otro lado, los dispositivos MPU proporcionan la primera solución integrada de procesador de movimiento de 6 ejes, que elimina la desalineación del eje transversal del giroscopio y acelerómetro a nivel de paquete asociado con soluciones discretas. Los dispositivos combinan un giroscopio de 3 ejes y un acelerómetro de 3 ejes en la misma matriz de silicio junto con un Digital Motion Processor ™ (DMP ™) integrado capaz de procesar algoritmos complejos de fusión de sensores de 9 ejes utilizando el MotionFusion ™ patentado y probado en el campo. Los algoritmos integrados MotionFusion de 9 ejes de MPU-6050 acceden a magnetómetros externos u otros sensores a través de una interfaz I2C maestro auxiliar, lo que permite que los dispositivos recopilen un conjunto completo de datos de sensores sin intervención del procesador del sistema. Para un seguimiento preciso de movimientos rápidos y lentos, el MPU-6050 presenta un rango de escala completa de giroscopio programable por el usuario de ± 250, ± 500, ± 1000 y ± 2000 ° / seg (dps). Las

piezas también tienen un rango de escala completa de acelerómetro programable por el usuario de ± 2g, ± 4g, ± 8g y ± 16g.(Ave et al., 2013)

2.3.Método de integración de componentes electrónicos

Se tiene que implementar la comunicación de esta interfaz, simulando la secuencia con sus puertos GPIO (Entrada de Salida de Propósito General). La unidad de medida inercial MPU6050 tiene dos interfaces I2C. Este sensor actúa como esclavo cuando se comunica con el microprocesador a través de su I2C primaria. (Fei et al., 2014)

Para la programación es necesario verificar si se encuentran todas las librerías instaladas las cuales son:

- "MPU6050.h"
- "Wire.h"
- “WiFi.h”
- “WiFiClient.h”
- “WiFiAP.h”

Después se integran las características y funciones del microprocesador y la unidad de medición inercial mediante programación. Existirá una conexión inalámbrica entre esclavo - maestro, para después transferir la información del maestro hacia la interfaz gráfica, permitiendo poder procesar esta información transformándola en una animación del cuerpo estudiado.

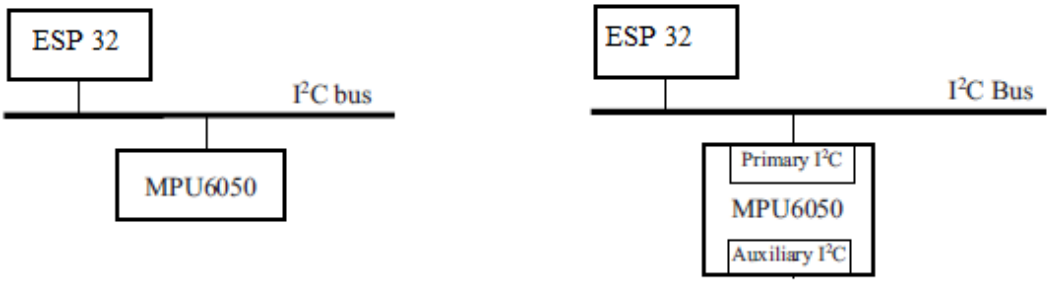

Figura 4: Arquitectura del sensor con los componentes IMU 5060 y el microprocesador ESP 32

La Figura 4 muestra la arquitectura del sistema del hardware, que consiste en un microprocesador *ESP32 WROOM – 32 y* un inercial sensor *MPU6050*.

2.4.Esquema de funcionamiento

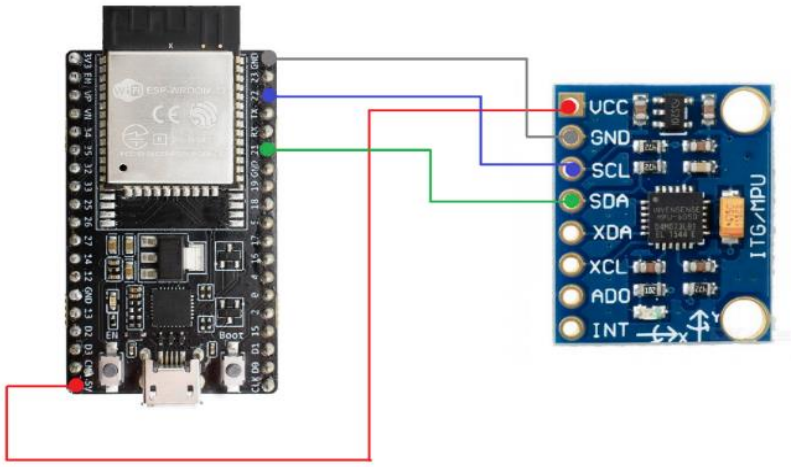

Figura 5: Conexión ESP32 con unidad de medida inercial MPU 6050

En la figura 5 se puede ver la conexión en entre el microprocesador (Esclavo) y la unidad de medida inercial IMU, los cuales procederán a realizar las mediciones de posición

aceleración y ángulos respectivos, al momento de ser colocados en el sistema vivo. Las conexiones entre ellos son:

- Pin SDA del MPU 6050, se conecta al pin GPIO 21 de ESP32 WROOM-32
- Pin SCL del MPU 6050 se conecta al pin GPIO 22 de ESP32 WROOM-32.
- Pin VCC del MPU 6050 se conecta al pin de 5V del ESP32 WROOM-32.
- Pin GND del MPU 6050 se conecta al pin de GND del ESP32 WROOM-32.

Figura 6: Sensor individual, compactado con todos sus componentes

En la figura 6 se observa el sensor individual integrado por la unidad inercial *MPU 6050*, la batería y el microprocesador *ESP32 Wroom 32*. El sensor a simple vista es compacto, pequeño y de fácil movilidad.

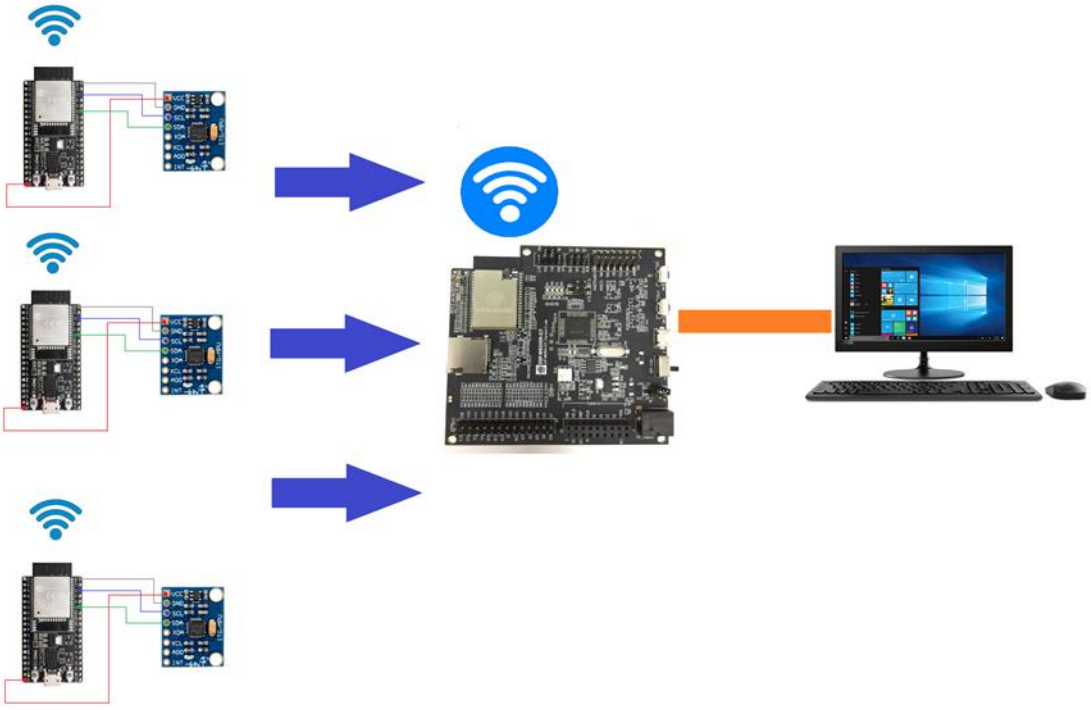

Figura 7: Sistema de sensores con conexión wifi.

En la figura 7 se puede observar el funcionamiento del sistema de red inalámbrica de sensores corporales, donde tres esclavos van a ser colocados en el cuerpo, tomando las respectivas mediciones que serán enviadas a una placa maestra que después transmitirá la información recolectada a una interfaz gráfica, donde se procederá a realizar la animación de la captura de movimiento por los sensores.

2.5.Programación

El sistema de sensores corporales cuenta con cuatro programaciones, las cuales, dos de ellas serán para la transmisión y recolección de datos de los sensores individuales. La tercera programación se encargará de realizar el cálculo de la posición mediante la doble integración de la aceleración dada por la unidad IMU 6050, dando como resultados las coordenas X Y, y la animación del movimiento de cada uno de los sensores que se encuentran

conectados al sistema de sensores. La cuarta y última programación procede a realizar la animación de las coordenadas en función de los ángulos de inclinación.

A continuación, se procederá a detallar cada una de las programaciones.

2.5.1. Programación Placa Maestro:

Se realizará la configuración por software de la placa ESP32 a través de Arduino IDE, para lo cual, como primer paso, se incluye todas las librerías antes listadas. Se le asigna un SSID, el cual será el nombre de la red creada para acceso de los demás dispositivos, se abre el puerto serial de la misma a una velocidad de 115200 baudios (bits por minuto), después de esto se activa el puerto 80 como un servidor para acceso por wifi. De aquí en adelante en el bucle infinito la placa verifica que exista o que no exista datos en el búfer de comunicación; de existir datos en el mismo, se procede a decodificar los datos enviados desde ESP32 esclavo. La decodificación consiste en sacar el residuo del dato recibido dividido para 10, el cual nos indicará de qué sensor y de qué eje llega cada una de la información. La programación fue basada en los códigos de (Leantec, 2016) y (Nylamp, 2016). El código se detallará en los anexos

2.5.2. Programación de sensores esclavos individuales

Se realizará la configuración por software de la placa ESP32 a través de Arduino IDE, para lo cual, como primer paso, se incluye todas las librerías antes listadas. Se le asigna un SSID, el cual será el nombre de la red creada para acceso de los demás dispositivos, se abre el puerto serial de la misma a una velocidad de 115200 baudios (bits por minuto), después de esto se activa el puerto 80 como un servidor para acceso por wifi. De aquí en adelante en el bucle infinito la placa verifica que exista o que no exista datos en el búfer de comunicación; de existir datos en el mismo, se procede a decodificar los datos enviados desde ESP32 esclavo. La decodificación consiste en sacar el residuo del dato recibido dividido para 10, el

cual nos indicará de qué sensor y de qué eje llega cada una de la información. La programación fue basada en los códigos de (Leantec, 2016) y (Nylamp, 2016). El código se detallará en los anexos

2.5.3. Programación de animación en el software “Processing”

El software Processing ha promovido la alfabetización de software dentro de las artes y la alfabetización visuales, para la programación en este software es lenguaje C, por lo tanto, tiene compatibilidad con ARDUINO IDE, permitiendo la transferencia directa de los datos a este programa. Se procede a ingresar la programación y se corre el programa permitiendo la observación de la animación y las coordenas respectivas para cada sensor. Fuente del código (García, 2018)

3. RESULTADOS Y DISCUSIONES

3.1 Obtención de datos de ángulo inclinado y aceleración

3.1.1. Envío y codificación de información desde el sensor esclavo a la tarjeta maestra

En la programación del esclavo, se transforma el dato recibido por el sensor a una variable de tipo “string” (cadena de caracteres), a esta variable, se le suma una letra Mayúscula, la cual será el índice, que identifica el número de sensor y el dato con el cual se trabaja. Seguido, se le suma un nuevo dato con un nuevo índice a esta misma variable, de esta manera se puede enviar una sola cadena de caracteres con los índices necesarios para ser decodificados en la tarjeta maestra. Este proceso hace que la comunicación esclavo-maestro sea más eficiente y se puede reconocer de qué sensor proviene la información; en este caso los datos serán de aceleración y ángulo de inclinación.

3.1.2. Recepción y decodificación de información en la tarjeta maestra

Para la recepción de información se utiliza el comando “indexof”, el cual permite que se devuelva la posición donde se encuentra el carácter que se requiere; en el caso que no exista el valor en la cadena de caracteres, el programa devuelve el valor de -1.

Como se menciona antes, los datos enviados del esclavo a la placa maestra se encuentran en una cadena de caracteres de índices y números, para que después, la placa maestra se encargue en decodificar esta información transformándola con el comando “toint” en una variable entera; este último dato arrojado sería el resultado real tomado en la medición.

Índices a utilizar en cada sensor son:

Sensor 1:

Ángulos de inclinación: J y K

Aceleración angula: A, B y C

Sensor 2:

Ángulos de inclinación: L y M

Aceleración angula: D, E y F

Sensor 3:

Ángulos de inclinación: P y Q

Aceleración angula: G, H y I

Una vez decodificada toda la información se procede agrupar por sensor los datos de angulación y aceleración con las siguientes variables:

Sensor 1:

Ángulos de inclinación: x1 y y1

Aceleración angular: ax1, ay1 y az1

Tabla 3:Variables de inclinación y aceleración del Sensor 1

K1=	y1=	ax1=	ay1=	az1=
29	44	0	0	0

Sensor 2:

Ángulos de inclinación: X2 y Y2

Aceleración angular: ax2, ay2 y az2

Tabla 4: Variables de inclinación y aceleración del Sensor 2

X2=	y2=	ax2=	ay2=	az2=
110	34	8660	-12380	-8204

Sensor 3:

Ángulos de inclinación: X3 y Y3

Aceleración angular: ax3, ay3 y az3

Tabla 5: Variables de inclinación y aceleración del Sensor 3

X3=	y3=	ax3=	ay3=	az3=
2147483520		-2147483520	0	0

Para realizar una toma de datos más exacta, es necesario calibrar los sensores, donde todos sus datos deben estar en cero. Una vez calibrados los sensores, se procede a la toma de mediciones, posicionándole en un plano x y moviendo el sensor en diferentes direcciones, para lo cual los resultados fueron:

Tabla 6: Resultados de ángulos de inclinación y aceleración del sensor 1

Ángulos de Inclinación		Aceleración		
X	Y	AX	AY	AZ
X1=	y1=	ax1=	ay1=	az1=
41	63	119	12600	-10748
X1=	y1=	ax1=	ay1=	az1=
41	63	119	12548	-10748
X1=	y1=	ax1=	ay1=	az1=
41	63	134	11812	-10748
X1=	y1=	ax1=	ay1=	az1=
41	63	103	10832	-10748
X1=	y1=	ax1=	ay1=	az1=
41	63	58	-13952	-10748
X1=	y1=	ax1=	ay1=	az1=
41	63	110	32216	-10748

*Tabla **7:**Resultados de ángulos de inclinación y aceleración del sensor 2*

Ángulos de inclinación		Aceleración		
X	**Y**	**AX**	**AY**	**AZ**
X2=	y2=	ax2=	ay2=	az2=
75	79	-1592	-468	-16476
X2=	y2=	ax2=	ay2=	az2=
75	79	-1592	-468	-16476
X2=	y2=	ax2=	ay2=	az2=
75	79	-1592	-468	-16476
X2=	y2=	ax2=	ay2=	az2=
75	79	-1592	-468	-16476
X2=	y2=	ax2=	ay2=	az2=
75	79	-1592	-468	-16476
X2=	y2=	ax2=	ay2=	az2=
75	79	-1592	-468	-16476

Tabla 8: Resultados de ángulos de inclinación y aceleración del sensor 3

Ángulos de inclinación		Aceleración		
X	**Y**	**AX**	**AY**	**AZ**
X3=	y3=	ax3=	ay3=	az3=
111	27	111	-13044	-4872
X3=	y3=	ax3=	ay3=	az3=
110	27	110	-13104	-4932
X3=	y3=	ax3=	ay3=	az3=
111	28	111	-12888	-4944
X3=	y3=	ax3=	ay3=	az3=
111	28	111	-13068	-5032
X3=	y3=	ax3=	ay3=	az3=
110	27	110	-12992	-5008
X3=	y3=	ax3=	ay3=	az3=
111	27	111	-13096	-4884
X3=	y3=	ax3=	ay3=	az3=

Cada uno de los datos estará identificado con el número de sensor correspondiente, como se indica en las tablas presentadas.

A continuación, se procederá a la obtención de posición mediante dos métodos.

3.2. Obtención De Posición

3.2.1. Método 1: Obtención de posición mediante la doble integral de la aceleración

La información obtenida de aceleración se procesa se envía al programa Processing, donde se procede a calcular la doble integral de la aceleración usando métodos numéricos dando como resultados la posición. Seguido de esta información, el programa procede a realizar la animación del movimiento de los tres sensores en tiempo real como se muestra en las siguientes figuras.

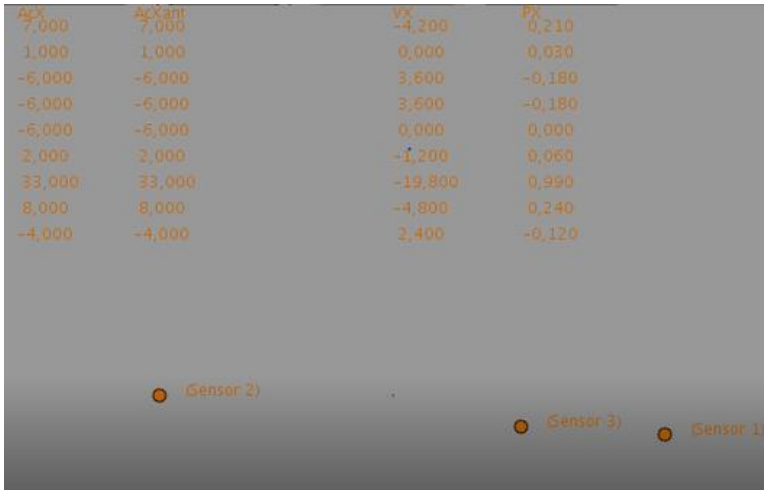

Figura 8:Observación de animación y cuando los sensores se encuentran en cadena

En la Figura 8 se puede observar como el programa da datos de la aceleración para después realizar el método numérico de integración, teniendo como resultado la velocidad y la posición, los sensores se encuentran en una posición cero, sin embargo, la animación no coincide con la posición de los sensores que se muestra en la figura 9.

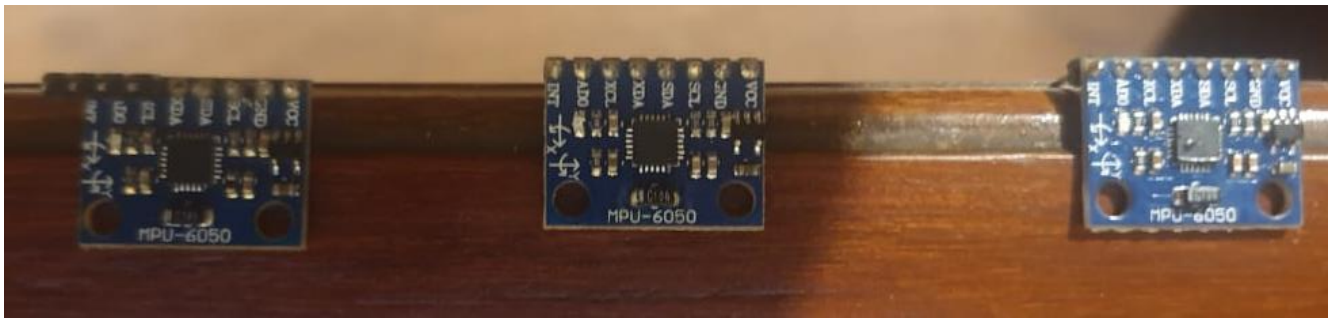

Figura 9: Ubicación de los sensores de prueba MPU 6050

En la figura 8, se puede observar los tres sensores representados en forma de círculos anaranjados. Se realizó una prueba de movimiento, donde se procede a colocar los sensores en una posición determinada horizontalmente, sin embargo, no coinciden en la posición real de los sensores y al momento de moverlos presentan un rango de error considerable ya que su punto inicial cambia y esto no permite que sea un resultado real al del movimiento. Se recalca que todo el procesamiento de los datos es en tiempo real, por lo tanto, debe tener coordinación con el movimiento que se realiza en vivo con los resultados que nos da el procesador, por lo que desde el principio no se pudo obtener.

3.2.2. Método 2: Obtención de posición mediante ángulos de inclinación.

Con los primeros datos de ángulos de inclinación, se realiza una nueva programación en el software Processing, donde en función de los ángulos pitch y roll se obtiene las coordenadas X Y de cada uno de los sensores como se muestra a continuación en las figuras.

Figura 10: Primera posición con los sensores colocados.

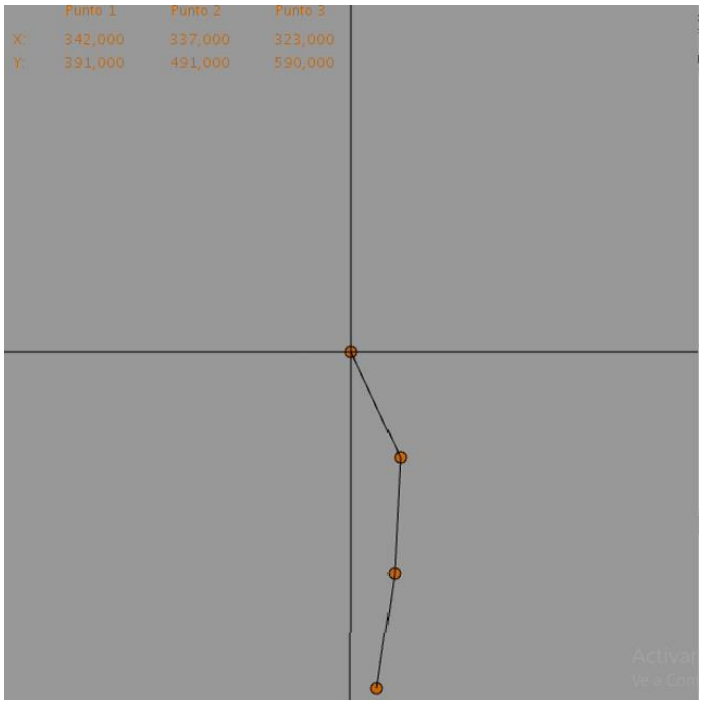

Figura 11: Animación de la primera posición de los tres sensores colocados

Figura 12: Segunda posición con los sensores colocados

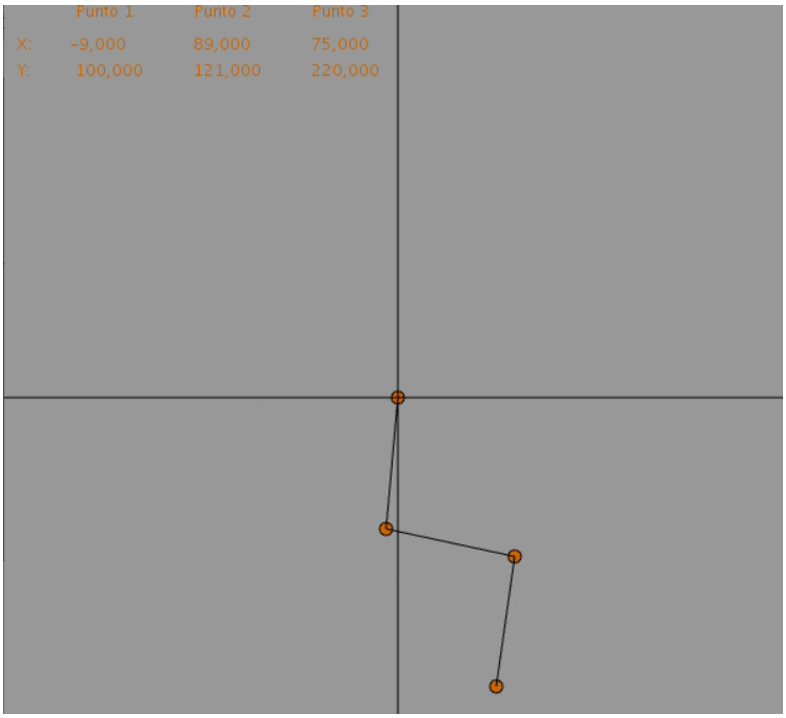

Figura 13: Animación de la segunda posición de los sensores colocados.

Figura 14: Tercera posición con los sensores colocados.

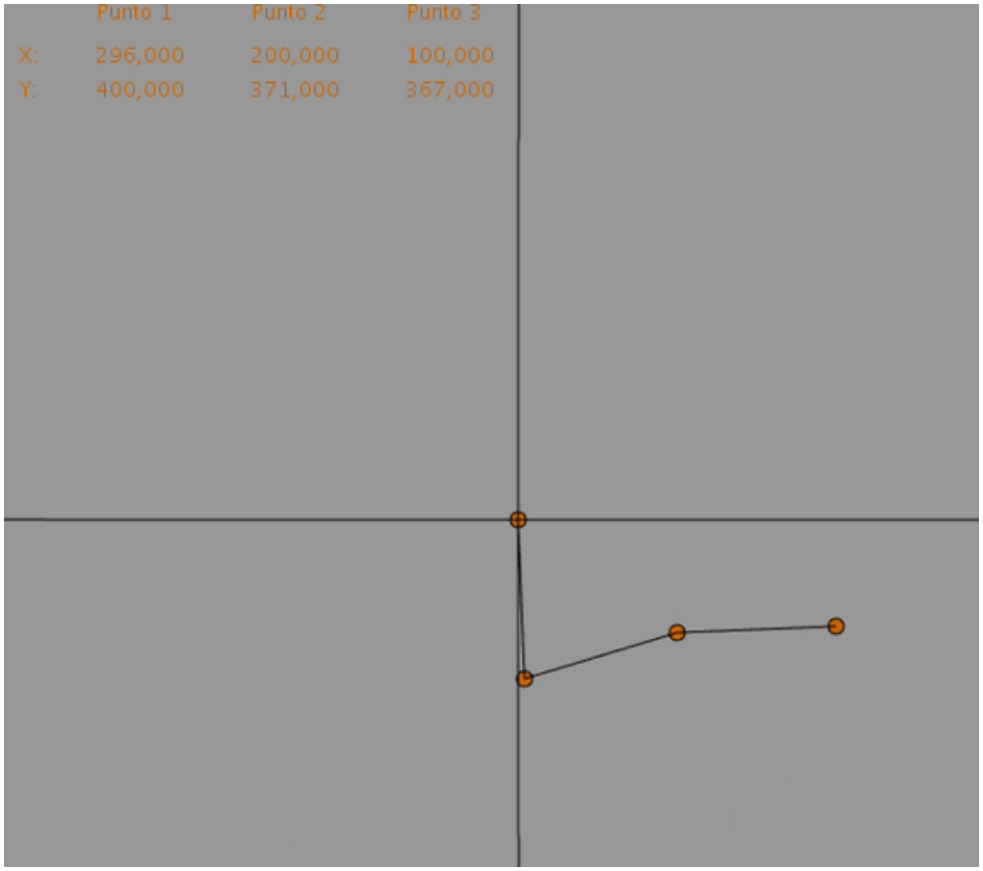

Figura 15: Animación de la tercera posición de los tres sensores colocados

Como se puede observar desde la figura 10 hasta la 15, se encuentra un circuito de movimientos donde se refleja la animación en tiempo real de la captura de movimiento de

los tres sensores que se ponen a prueba. En las figuras de las animaciones se pueden notar cuatro puntos anaranjados, sin embargo, los tres sensores están identificados a partir del segundo punto superior.

En las figuras de posición se pueden observar la ubicación de cada uno de los sensores, el primer sensor está ubicado en el brazo, el segundo en el ante brazo y el último cerca de la muñeca.

En cada par de figuras de animación y posición, se puede notar la similitud de entre la captura de movimiento y la animación, reflejando una mayor exactitud visual a comparación del primer método.

3.3. Análisis de eficacia del sistema y opción a mejora.

Por los resultados obtenidos, se realizó una búsqueda de opción a mejora del sistema, donde sea más eficaz y satisfaga las necesidades para poder realizar un análisis biomecánico, por lo que se determinó como primer objetivo cambiar la unidad inercial MPU 6050 a un MPU 9250, ya que el primero trabaja en solo 6 grados de libertad y el segundo trabaja con 9 grados de libertad. Se encontró un módulo que ya cuenta con los componentes necesarios ya integrados, los cuales son: batería, unidad de medida inercial MPU 9250 y el microprocesador integrado ESP 32. Con este módulo sería solo necesario incorporar la programación ya realizada. Como resultado se obtiene un sistema que sea más compacto al ya implementado, con mayor exactitud al momento de la recolección de datos y resultados mas eficaces y en tres dimensiones. En cuestión de precio, incluso llega a ser mucho más barato que el de ahora, esto se podrá observas en las siguientes tablas.

Tabla 9: Costos del sistema implementado

Materiales	Unidad	Costo Unitario	Costo total
ESP-32 WROOM 32	3	$ 10,00	$ 30,00
ESP-32 Wrover Kit	1	$ 55,00	$ 55,00
MPU 6050	3	$ 6,50	$ 19,50
Batería	4	$ 5,00	$ 20,00
Cables	1	$ 5,00	$ 5,00
Total	12	81,5	129,5

Tabla 10: Costos del sistema mejorado para trabajo futuro

Material	Unidad	Costo Unitario	Costo Total
Kit de desarrollo ESP32	20	$ 14,95	$ 299,00
Kit de desarrollo ESP32 Grey	1	$ 33,95	$ 33,95
Total	21	$ 48,90	$ 332,95

Como se puede observar en las tablas 9 y 10, la variación de precios si se reduce la una con la otra. Teniendo como más económica y eficaz la opción a mejora. El sistema ya implementado se procedió a tener un gasto aproximado de $130 dólares en tan solo la implementación de 3 sensores, mientras que, en la segunda opción con un sistema conformado por 20 sensores se tiene un aproximado de $333 dólares. Teniendo como conclusión que el sistema es de gran utilidad en el campo académico, y que con la inversión económica y de tiempo sería un gran proyecto.

3.4 Discusión

Los primeros datos que nos da cada uno de los sensores individuales, son resultados absolutos que son directamente dados por los giroscopios y acelerómetros que integran la unidad de medida inercial *MPU 6050*. La unidad de medida inercial, en sí, ya tiende a tener ruido al momento de la recolección de datos, pero para evitar este ruido y obtener un resultado absoluto, se procede a utilizar un low pass el cual es un filtro que elimina o filtra un rango específico de componentes de frecuencia, acercándose al valor más real. En la programación, este filtro permite que el ruido disminuya considerablemente, permitiendo que, al momento de imprimirse el dato, este sea confiable. Sin embargo, al procesar los datos en el computador y realizar la doble integración por el método numérico, el low pass perderá su función y al contrario de dar un valor más exacto, este tiende a ser cada vez más erróneo a medida que el sensor capte el movimiento. Por otra parte, utilizando el segundo método, el resultado final de posición no pierde la veracidad de los datos de ángulos de inclinación. La programación en este método sigue contando con el filtro low pass, y se procede a realizar la programación en el software Processing en el cual solo se procede a obtener las coordenadas directas de los ángulos. Sin embargo, los datos que se obtendrán serán solo en dos dimensiones, por lo que se recomienda utilizar sensores inerciales industriales, los cuales ya cuentan con su propio microcontrolador y permiten que el sensor, sin realizar ningún método numérico brinde datos exactos de desplazamiento y posición.

CONCLUSIONES

- Se escogió los materiales que se creían más convenientes en el momento, sin embargo, se determinó que para que el sistema sea más eficaz y brinde datos en tres dimensiones se tendría que cambiar la unidad de medida inercial de un MPU 6050 a un MPU 9250, que cuenta con magnetómetro aparte, del acelerómetro y giroscopio, permitiendo obtener las mediciones más precisas y en tres dimensiones no solo en dos.
- Se realizó el análisis con observaciones a mejoras para optimización del sistema de sensores corporales.
- En el análisis se determinó un ahorro considerable al cambiar de materiales para el sistema.
- Se recomienda el cambio de materiales para un sistema más eficaz.
- Se recomienda realizar más pruebas de precisión del sistema.
- Se recomienda encontrar el margen de error de medición del sistema actual y el de trabajos futuros.
- Una vez probado el sistema, aplicar para estudios de análisis de captura de movimiento de un cuerpo vivo.

REFERENCIAS BIBLIOGRÁFICAS

Altinel, Y., Öztürk, E., Akyıldız, E. Ü., Ulcay, Y., & Özgüc, H. (2012). The effect of a chitosan coating on the adhesive potential and tensile strength of polypropylene meshes. *Sociedad Hispanoamericana de Hernia*, 15.

Ave, B., Number, D., & Date, R. (2013). MPU-6500 Register Map and Descriptions, *1*(408), 1–47.

Cuervo, M. C., Olaya, A. F. R., & Salamanca, R. M. G. (2013). Métodos de captura de movimiento biomecánico enfocados en telefisioterapia. *Pan American Health Care Exchanges, PHCE*, 0–5. https://doi.org/10.1109/PAHCE.2013.6568264

Fei, Y., Song, Y., Xu, L., & Sun, G. (2014). Micro-IMU based Wireless Body Sensor Network. *Proceedings of the 33rd Chinese Control Conference, CCC 2014*, *16255*, 428–432. https://doi.org/10.1109/ChiCC.2014.6896661

Flores, J., & Sandoval-gonzalez, O. O. (2015). Diseño y desarrollo de un sistema de captura de movimiento para análisis biomecánico. *Coloquio de Investigación Multidisciplinaria*, *3*(October 2015), 246–253. Retrieved from https://www.researchgate.net/publication/321886197

Leantec. (18 de Enero de 2016). *Leantec*. Obtenido de https://leantec.es/tutorial-arduino-acelerometro-giroscopo-mp/

Lisbeth, L., Echeverry, G., Melissa, A., Henao, J., Angélica, M., Molina, R., … Bolívar, S. (2018). Sistemas de captura y análisis de movimiento cinemático humano : una revisión sistemática Human motion capture and analysis systems : a systematic review, *16*(2).

Martinez-Méndez, R., Portillo-Rodriguez, O., Romero-Huertas, M., & Vilchis-González, A. (2012). Uso de sensores inerciales en la medicion y evaluación de movimiento humano para aplicaciones en la salud. *Ideas En Ciencia*, *37*(July), 61–75.

Microsoft. (2019). *Microsoft*. Obtenido de https://developer.microsoft.com/es-es/windows/hardware/3d-print/scanning-with-kinect

Nylamp. (2016). *https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html*. Obtenido de https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html

Ong, Z. C., Seet, Y. C., Khoo, S. Y., & Noroozi, S. (2018). Development of an economic wireless human motion analysis device for quantitative assessment of human body joint. *Measurement: Journal of the International Measurement Confederation*,

115(November 2017), 306–315. https://doi.org/10.1016/j.measurement.2017.10.056

Roetenberg, D., Luinge, H., & Slycke, P. (2009a). *Xsens MVN : Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors*. *Hand, The*. https://doi.org/10.1.1.569.9604

Roetenberg, D., Luinge, H., & Slycke, P. (2009b). Xsens MVN : Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors. *Hand, The*, (January 2009), 1–7. https://doi.org/10.1.1.569.9604

Rui L, R., Nuno M, N., Joao F, M., Manuela E, G., Alexandra P, M., & Helena S, A. (2008). *Natural-based polymers for biomedical applications* . North America : CRC Press LLC.

Santos, T. M. O., Barroso, M. F. S., Ricco, R. A., Nepomuceno, E. G., Alvarenga, É. L. F. C., Penoni, Á. C. O., & Santos, A. F. (2019). A low-cost wireless system of inertial sensors to postural analysis during human movement. *Measurement*, *148*, 106933. https://doi.org/10.1016/j.measurement.2019.106933

Systems, E. (2019). ESP32 Series Datasheet. *Espressif Systems*, 1–61. Retrieved from https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Tadano, S., Takeda, R., & Miyagawa, H. (2013). Three dimensional gait analysis using wearable acceleration and gyro sensors based on quaternion calculations. *Sensors (Switzerland)*, *13*(7), 9321–9343. https://doi.org/10.3390/s130709321

Universo, E. (2016). *https://www.eluniverso.com/deportes/2016/02/25/nota/5425626/biomecanica-presente-deporte-ecuatoriano*.

Villa, A. C., Díaz, M., & Urgilés, F. (2008). Investigación en el área de la biomecánica retos y perspectivas en el Ecuador, 969–974.

Xu, C., He, J., Zhang, X., Yao, C., & Tseng, P. H. (2018). Geometrical kinematic modeling on human motion using method of multi-sensor fusion. *Information Fusion*, *41*, 243–254. https://doi.org/10.1016/j.inffus.2017.09.014

ANEXO A: PRIMERA PROGRAMACIÓN PLACA MADRE

```
#include <Arduino.h>

#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>


// Set these to your desired credentials.
const char *ssid = "ESP32ABAD";
const char *password = "12345678";
int varX_1,varY_1,varZ_1,varX_2,varY_2,varZ_2,varX_3,varY_3,varZ_3,varAx_1,v
arAx_2,varAx_3,varAy_3,varAy_2,varAy_1,varAz_3,varAz_2,varAz_1;
WiFiServer server(80);


void setup() {
  //pinMode(LED_BUILTIN, OUTPUT);
  pinMode(32,OUTPUT);
  Serial.begin(115200);
  Serial.println();
  Serial.println("Configuring access point...");

  // You can remove the password parameter if you want the AP to be open.
  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  server.begin();
  Serial.println("Server started");
}

void loop() {
  WiFiClient client = server.available();   // listen for incoming clients
String s1="";
  if (client) {                             // if you get a client,
    //Serial.println("New Client.");           // print a message out the se
rial port
    String currentLine = "";                // make a String to hold incomin
g data from the client
    s1="";
    while (client.connected()) {            // loop while the client's conne
cted
      if (client.available()) {             // if there's bytes to read from
 the client,
```

```
        char c = client.read();             // read a byte, then
        s1+=c;
      }//Serial.flush();

    }
 int poss1=s1.indexOf('J');
 int poss2=s1.indexOf('K');
 int poss29=s1.indexOf('k');
 int poss11=s1.indexOf('A');
 int poss12=s1.indexOf('B');
 int poss13=s1.indexOf('C');


 int poss3=s1.indexOf('L');
 int poss4=s1.indexOf('M');
 int poss49=s1.indexOf('m');
 int poss21=s1.indexOf('D');
 int poss22=s1.indexOf('E');
 int poss23=s1.indexOf('F');

 int poss5=s1.indexOf('P');
 int poss6=s1.indexOf('Q');
 int poss69=s1.indexOf('q');
  int poss31=s1.indexOf('G');
 int poss32=s1.indexOf('H');
 int poss33=s1.indexOf('I');

 String s2=" ";
 /* Serial.print("X1=");
 Serial.print("\tY1=");
  Serial.print("\tZ1=");
 Serial.print("\tax1=");
Serial.print("\tay1=");
Serial.print("\taz1=");

 Serial.print("\tX2=");
 Serial.print("\tY2=");
  Serial.print("\tZ2=");
 Serial.print("\tax2=");
Serial.print("\tay2=");
Serial.print("\taz2=");

 Serial.print("\tX3=");
 Serial.print("\tY3=");
 Serial.print("\tZ3=");
```

```
 Serial.print("\tax3=");
Serial.print("\tay3=");
Serial.println("\taz3="); */
 //Serial.println("\tY sensor 1=");
 if(s1.indexOf('J')>0){for(int i=0;i<poss1;i++){s2+=s1[i];
                                                varX_1=s2.toInt();}
                                                s2=' ';
                     //Serial.print("X1=");
                     //Serial.print(varX_1);
                     //moverServo(32,varX_1);
                     }
                     if(s1.indexOf('K')>0){for(int i=poss1+1;i<poss2;i++){s2+
=s1[i];
                                                varY_1=s2.toInt();}
                                                s2=' ';
                                                 //Serial.print("\tY1=");
                                                 //Serial.println(varY_1);
                                               }
                                               if(s1.indexOf('k')>0){for(int i=po
ss2+1;i<poss29;i++){s2+=s1[i];
                                                varZ_1=s2.toInt();}
                                                s2=' ';
                                                 //Serial.print("\tY1=");
                                                 //Serial.println(varY_1);
                                               }
                      if(s1.indexOf('A')>0){for(int i=poss29+1;i<poss11;i++){
s2+=s1[i];
                                                varAx_1=s2.toInt();}
                                                s2=' ';
                     //Serial.print("\tY1=");
                     //Serial.println(varY_1);
                     }
                      if(s1.indexOf('B')>0){for(int i=poss11+1;i<poss12;i++){
s2+=s1[i];
                                                varAy_1=s2.toInt();}
                                                s2=' ';
                     //Serial.print("\tY1=");
                     //Serial.println(varY_1);
                     }
                      if(s1.indexOf('C')>0){for(int i=poss12+1;i<poss13;i++){
s2+=s1[i];
                                                varAz_1=s2.toInt();}
                                                s2=' ';
                     //Serial.print("\tY1=");
                     //Serial.println(varY_1);
```

```
                    }



if(s1.indexOf('L')>0){for(int i=0;i<poss3;i++){s2+=s1[i];
                                            varX_2=s2.toInt();}
                                            s2=' ';
                    //Serial.print("\tX2=");
                    //Serial.println(varX_2);
                    }
                    if(s1.indexOf('M')>0){for(int i=poss3+1;i<poss4;i++){s2+
=s1[i];
                                                                        var
Y_2=s2.toInt();}
                                                                        s2=
' ';
                                                                        //S
erial.print("\tY2=");
                                                                        //S
erial.println(varY_2);
                                                                         }
                                                                         if
(s1.indexOf('m')>0){for(int i=poss4+1;i<poss49;i++){s2+=s1[i];
                                                                        var
Z_2=s2.toInt();}
                                                                        s2=
' ';
                                                                        //S
erial.print("\tY2=");
                                                                        //S
erial.println(varY_2);
                                                                         }
                     if(s1.indexOf('D')>0){for(int i=poss49+1;i<poss21;i++){
s2+=s1[i];
                                            varAx_2=s2.toInt();}
                                            s2=' ';
                    //Serial.print("\tY1=");
                    //Serial.println(varY_1);
                    }
                     if(s1.indexOf('E')>0){for(int i=poss21+1;i<poss22;i++){
s2+=s1[i];
                                            varAy_2=s2.toInt();}
                                            s2=' ';
                    //Serial.print("\tY1=");
```

```
                    //Serial.println(varY_1);
                    }
                     if(s1.indexOf('F')>0){for(int i=poss22+1;i<poss23;i++){
s2+=s1[i];
                                             varAz_2=s2.toInt();}
                                             s2=' ';
                    //Serial.print("\tY1=");
                    //Serial.println(varY_1);
                    }
if(s1.indexOf('P')>0){for(int i=0;i<poss5;i++){s2+=s1[i];
                                             varX_3=s2.toInt();}
                                             s2=' ';
                    //Serial.print("\tY2=");
                    //Serial.println(varY_2);
                    }
                    if(s1.indexOf('Q')>0){for(int i=poss5+1;i<poss6;i++){s2+
=s1[i];
                                                                        var
Y_3=s2.toInt();}
                                                                        s2=
' ';
                                                                         //
Serial.print("\tY2=");
                                                                         //
Serial.println(varY_2);
                                                                         }
                                                                          i
f(s1.indexOf('q')>0){for(int i=poss6+1;i<poss69;i++){s2+=s1[i];
                                                                        var
Z_3=s2.toInt();}
                                                                        s2=
' ';
                                                                         //
Serial.print("\tY2=");
                                                                         //
Serial.println(varY_2);
                                                                         }
                     if(s1.indexOf('G')>0){for(int i=poss69+1;i<poss31;i++){
s2+=s1[i];
                                             varAx_3=s2.toInt();}
                                             s2=' ';
                    //Serial.print("\tY1=");
                    //Serial.println(varY_1);
                    }
```

```
                     if(s1.indexOf('H')>0){for(int i=poss31+1;i<poss32;i++){
s2+=s1[i];
                                           varAy_3=s2.toInt();}
                                           s2=' ';
                    //Serial.print("\tY1=");
                    //Serial.println(varY_1);
                    }
                     if(s1.indexOf('I')>0){for(int i=poss32+1;i<poss33;i++){
s2+=s1[i];
                                           varAz_3=s2.toInt();}
                                           s2=' ';
                    //Serial.print("\tY1=");
                    //Serial.println(varY_1);
                    }
//angulo  1
Serial.print("aAX1 = ");
Serial.println(varX_1);
Serial.print("aAY1 = ");
Serial.println(varY_1);
Serial.print("aAZ1 = ");
Serial.println(varZ_1);
//aceleración
/* Serial.print(varAx_1);
Serial.print("\t");
Serial.print(varAy_1);
Serial.print("\t");
Serial.print(varAz_1);
Serial.print("\t"); */
//angulo 2
Serial.print("aAX2 = ");
Serial.println(varX_2);
Serial.print("aAY2 = ");
Serial.println(varY_2);
Serial.print("aAZ2 = ");
Serial.println(varZ_2);

/* Serial.print(varAx_2);
Serial.print("\t");
Serial.print(varAy_2);
Serial.print("\t");
Serial.print(varAz_2);
Serial.print("\t"); */
//angulo
Serial.print("aAX3 = ");
Serial.println(varX_3);
```

```
Serial.print("aAY3 = ");
Serial.println(varY_3);
Serial.print("aAZ3 = ");
Serial.println(varZ_3);

/* Serial.print(varAx_3);
Serial.print("\t");
Serial.print(varAy_3);
Serial.print("\t");
Serial.println(varAz_3); */
    client.stop();
  }
}
```

ANEXO B: PRIMERA PROGRAMACIÓN SENSORES ESCLAVOS INDIVIDUALES

```
#include "MPU6050.h"
#include "Wire.h"
#include <WiFi.h>
#include <WiFiMulti.h>
WiFiMulti WiFiMulti;
MPU6050 sensor;
float Xacc;
float Yacc;
float Zacc;
int16_t ax, ay, az;
int16_t gx, gy, gz;
float devT;
float aGx, aGy, aGz;
float xtr, ytr, ztr;
long tA0;
float r4,r5, r6;
int dt=250;
int minVal=265; int maxVal=402;
double x; double y; double z;
const int MPU=0x68;  // I2C address of the MPU-6050
int16_t AcX,AcY,AcZ;
void setup() {
  Serial.begin(115200);
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
    // We start by connecting to a WiFi network
```

```
    WiFiMulti.addAP("ESP32ABAD", "12345678");
    Serial.println();
    Serial.println();
    Serial.print("Buscando Servicio de Red ");

    while(WiFiMulti.run() != WL_CONNECTED) {

        delay(500);
    }

    Serial.println("");
    Serial.println("WiFi conectado");
    Serial.println("direccion IP: ");
    Serial.println(WiFi.localIP());

    delay(500);
}

void loop() {
   if((millis()-tA0)>dt) {
    const uint16_t port = 80;
    const char * host = "192.168.4.1";
    WiFiClient client;



  delay(10);
    if (!client.connect(host, port)) {
        Serial.println("Connection failed.");
        return;
    }

  Wire.beginTransmission(MPU);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,14,true);  // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read();
  AcY=Wire.read()<<8|Wire.read();
  AcZ=Wire.read()<<8|Wire.read();
  int xAng = map(AcX,minVal,maxVal,-90,90);
  int yAng = map(AcY,minVal,maxVal,-90,90);
  int zAng = map(AcZ,minVal,maxVal,-90,90);

x= RAD_TO_DEG * (atan2(-yAng, -zAng)+PI); y= RAD_TO_DEG * (atan2(-xAng, -
zAng)+PI); z= RAD_TO_DEG * (atan2(-yAng, -xAng)+PI);
```

```
//Sensor
client.print(String(x)+'P'+String(y)+'Q'+String(z)+'q' +Xacc+'G'+Yacc+'H'+Za
cc+'I'+'\n');
Serial.print(String(x)+'P'+String(y)+'Q'+String(z)+'q'+Xacc+'G'+Yacc+'H'+Zac
c+'I'+'\n');

tA0=millis();}
```

ANEXO C: PROGRAMACIÓN PARA OBTENCIÓN DE POSICIÓN MEDIANTE EL PRIMER MÉTODO

```
import processing.serial.*;
Serial myPort;  // Create object from Serial class
String val;     // Data received from the serial port
///////////////// sensor1
float AcX; float AcY; float AcZ;
float AcXant; float AcYant; float AcZant;
float VX; float VY;float VZ;
float VXant; float VYant;float VZant;
float PX; float PY;float PZ;
float PXant; float PYant;float PZant;
////////////// sensor2
float AcX2; float AcY2; float AcZ2;
float AcXant2; float AcYant2; float AcZant2;
float VX2; float VY2;float VZ2;
float VXant2; float VYant2;float VZant2;
float PX2; float PY2;float PZ2;
float PXant2; float PYant2;float PZant2;
////////////// sensor 3
float AcX3; float AcY3; float AcZ3;
float AcXant3; float AcYant3; float AcZant3;
float VX3; float VY3;float VZ3;
float VXant3; float VYant3;float VZant3;
float PX3; float PY3;float PZ3;
float PXant3; float PYant3;float PZant3;
float dt = 0.1;
float Tmp;
float GyX;
float GyY;
float GyZ;
/*String RAcX = "AcX = ";
String RAcY = "AcY = ";
String RAcZ = "AcZ = ";
*/
```

```
String RAcX = "AcX1 = ";
String RAcY = "AcY1 = ";
String RAcZ = "AcZ1 = ";
String RAcX2 = "AcX2 = ";
String RAcY2 = "AcY2 = ";
String RAcZ2 = "AcZ2 = ";
String RAcX3 = "AcX3 = ";
String RAcY3 = "AcY3 = ";
String RAcZ3 = "AcZ3 = ";
void setup()
{
  // I know that the first port in the serial list on my mac
  // is Serial.list()[0].
  // On Windows machines, this generally opens COM1.
  // Open whatever port is the one you're using.
  printArray(Serial.list());
  String portName = Serial.list()[1]; //change the 0 to a 1 or 2 etc. to match your port
  myPort = new Serial(this, portName, 115200);
  size(600,600,P3D);
  frameRate(5);
}
void draw()
{
   background(153);
 if ( myPort.available() > 0)
  {  // If data is available,
     String test = "AcX = 555";
     val = myPort.readStringUntil('\n');     // read it and store it in val
    if(val == null){
      val = "";
     }
     if(val.contains(RAcX)){
         String strNew = val.replaceFirst(RAcX, "");
         AcXant = AcX;
         AcX = Float.parseFloat(strNew);
      }else if(val.contains(RAcY)){
         String strNew = val.replaceFirst(RAcY, "");
         AcYant = AcY;
         AcY = Float.parseFloat(strNew);
      }else if(val.contains(RAcZ)){
          String strNew = val.replaceFirst(RAcZ, "");
          AcZant = AcZ;
          AcZ = Float.parseFloat(strNew);
      }else if(val.contains(RAcX2)){
          String strNew = val.replaceFirst(RAcX2, "");
          AcXant2 = AcX2;
          AcX2 = Float.parseFloat(strNew);
```

```
      }else if(val.contains(RAcY2)){
          String strNew = val.replaceFirst(RAcY2, "");
          AcYant2 = AcY2;
          AcY2 = Float.parseFloat(strNew);
      }else if(val.contains(RAcZ2)){
          String strNew = val.replaceFirst(RAcZ2, "");
          AcZant2 = AcZ2;
          AcZ2 = Float.parseFloat(strNew);
      }else if(val.contains(RAcX3)){
          String strNew = val.replaceFirst(RAcX3, "");
          AcXant3 = AcX3;
          AcX3 = Float.parseFloat(strNew);
      }else if(val.contains(RAcY3)){
          String strNew = val.replaceFirst(RAcY3, "");
          AcYant3 = AcY3;
          AcY3 = Float.parseFloat(strNew);
      }else if(val.contains(RAcZ3)){
        String strNew = val.replaceFirst(RAcZ3, "");
          AcZant3 = AcZ3;
          AcZ3 = Float.parseFloat(strNew);
      }
 }
 //////////////////sensor 1
  VXant = VX;
  VX = intVelocidad(AcXant,AcX,VXant,dt);
  VYant = VY;
  VY = intVelocidad(AcYant,AcY,VYant,dt);
  VZant = VZ;
  VZ = intVelocidad(AcZant,AcZ,VZant,dt);
 PXant = PX;
PX = intPosicion(VXant,VX,PXant,dt);
PYant = PY;
PY = intPosicion(VYant,VY,PYant,dt);
PZant = PZ;
  PZ = intPosicion(VZant,VZ,PZant,dt);
////////////////// sensor 2
  VXant2 = VX2;
  VX2 = intVelocidad(AcXant2,AcX2,VXant2,dt);
   VYant2 = VY2;
  VY = intVelocidad(AcYant2,AcY2,VYant2,dt);
  VZant2 = VZ2;
  VZ2 = intVelocidad(AcZant2,AcZ2,VZant2,dt);
  PXant2 = PX2;
  PX2 = intPosicion(VXant2,VX2,PXant2,dt);
  PYant2 = PY2;
  PY2 = intPosicion(VYant2,VY2,PYant2,dt);
  PZant2 = PZ2;
```

```
  PZ2 = intPosicion(VZant2,VZ2,PZant2,dt);
   ////////////////// sensor 3
  VXant3 = VX3;
  VX3 = intVelocidad(AcXant3,AcX3,VXant3,dt);
  VYant3 = VY3;
  VY3 = intVelocidad(AcYant3,AcY3,VYant3,dt);
  VZant3 = VZ3;
  VZ3 = intVelocidad(AcZant3,AcZ3,VZant3,dt);
  PXant3 = PX3;
  PX3 = intPosicion(VXant3,VX3,PXant3,dt);
  PYant3 = PY3;
  PY3 = intPosicion(VYant3,VY3,PYant3,dt);
  PZant3 = PZ3;
  PZ3 = intPosicion(VZant3,VZ3,PZant3,dt);
  println(val); //print it out in the console
  text("AcX",10,10); text("AcXant",100,10); text("VX",300,10); text("PX",400,10);
  text(AcX,10,20); text(AcXant,100,20); text(VX,300,20); text(PX,400,20);
  text(AcY,10,40); text(AcYant,100,40); text(VY,300,40); text(PY,400,40);
  text(AcZ,10,60);text(AcZant,100,60); text(VZ,300,60); text(PZ,400,60);
  text(AcX2,10,80); text(AcXant2,100,80); text(VX2,300,80); text(PX2,400,80);
  text(AcY2,10,100); text(AcYant2,100,100); text(VY2,300,100); text(PY2,400,100);
  text(AcZ2,10,120);text(AcZant2,100,120); text(VZ2,300,120); text(PZ2,400,120);
  text(AcX3,10,140); text(AcXant3,100,140); text(VX3,300,140); text(PX3,400,140);
  text(AcY3,10,160); text(AcYant3,100,160); text(VY3,300,160); text(PY3,400,160);
  text(AcZ3,10,180);text(AcZant3,100,180); text(VZ3,300,180); text(PZ3,400,180);
  String sData =
"("+PX+","+PY+","+PZ+")"+"("+VX+","+VY+","+VZ+")"+"("+AcX+","+AcY+","+AcZ
+")";
  String sData2 =
"("+PX2+","+PY2+","+PZ2+")"+"("+VX2+","+VY2+","+VZ2+")"+"("+AcX2+","+AcY2
+","+AcZ2+")";
  String sData3 =
"("+PX3+","+PY3+","+PZ3+")"+"("+VX3+","+VY3+","+VZ3+")"+"("+AcX3+","+AcY3
+","+AcZ3+")";
  fill(204,102,0);
  ellipse((1000*PX)+300,(1000*PY)+300,10f,10f);
  text("(Sensor 1)",(1000*PX)+300+20,(1000*PY)+300);
  ellipse((1000*PX2)+300,(1000*PY2)+300,10f,10f);
  line(PX+300,PY+300,PX2+300,PY2+300);
  text("(Sensor 2)",(1000*PX2)+300+20,(1000*PY2)+300);
  ellipse((100*PX3)+300,(100*PY3)+300,10f,10f);
  line(PX2+300,PY2+300,PX3+300,PY3+300);
  text("(Sensor 3)",(100*PX3)+300+20,(100*PY3)+300);
  }
public float intVelocidad(float Aant,float Aact,float Vant,float dt){
  return  Vant + (0.5*(Aant-Aact)*dt);
}
```

```
public float intPosicion(float Vant,float Vact,float Pant,float dt){
  return  Pant + (0.5*(Vant-Vact)*dt);
}
```

ANEXO D: PROGRAMACIÓN PARA OBTENCIÓN DE POSICIÓN MEDIANTE EL PRIMER MÉTODO

```
import processing.serial.*;

Serial myPort;  // Create object from Serial class
String val;     // Data received from the serial port

////////////////// sensor1
float AcX; float AcY; float AcZ;
float AcXant; float AcYant; float AcZant;
float VX; float VY;float VZ;
float VXant; float VYant;float VZant;
float PX; float PY;float PZ;
float PXant; float PYant;float PZant;


////////////// sensor2
float AcX2; float AcY2; float AcZ2;
float AcXant2; float AcYant2; float AcZant2;
float VX2; float VY2;float VZ2;
float VXant2; float VYant2;float VZant2;
float PX2; float PY2;float PZ2;
float PXant2; float PYant2;float PZant2;

////////////// sensor 3
float AcX3; float AcY3; float AcZ3;
float AcXant3; float AcYant3; float AcZant3;
float VX3; float VY3;float VZ3;
float VXant3; float VYant3;float VZant3;
float PX3; float PY3;float PZ3;
float PXant3; float PYant3;float PZant3;

float dt = 0.1;

float Tmp;
float GyX;
float GyY;
float GyZ;

float aAX1;
float aAY1;
```

```
float aAZ1;
float aAX2;
float aAY2;
float aAZ2;
float aAX3;
float aAY3;
float aAZ3;

String aAcX = "aAX1 = ";
String aAcY = "aAY1 = ";
String aAcZ = "aAZ1 = ";


String aAcX2 = "aAX2 = ";
String aAcY2 = "aAY2 = ";
String aAcZ2 = "aAZ2 = ";


String aAcX3 = "aAX3 = ";
String aAcY3 = "aAY3 = ";
String aAcZ3 = "aAZ3 = ";

String RAcX = "AcX = ";
String RAcY = "AcY = ";
String RAcZ = "AcZ = ";


/*
String RAcX = "AcX1 = ";
String RAcY = "AcY1 = ";
String RAcZ = "AcZ1 = ";*/

String RAcX2 = "AcX2 = ";
String RAcY2 = "AcY2 = ";
String RAcZ2 = "AcZ2 = ";

String RAcX3 = "AcX3 = ";
String RAcY3 = "AcY3 = ";
String RAcZ3 = "AcZ3 = ";

float pos1=300;
float pos2=300;
float pos3=300;

float largo1 = 100;
float largo2 = 100;
```

```
void setup()
{
  // I know that the first port in the serial list on my mac
  // is Serial.list()[0].
  // On Windows machines, this generally opens COM1.
  // Open whatever port is the one you're using.
  printArray(Serial.list());
  String portName = Serial.list()[1]; //change the 0 to a 1 or 2 etc. to match your port
  myPort = new Serial(this, portName, 115200);
  size(600,600,P3D);

  frameRate(180);

}

void draw()
{



   background(153);

  if ( myPort.available() > 0)
  {  // If data is available,

     val = myPort.readStringUntil('\n');     // read it and store it in val

     if(val == null){
       val = "";
     }
     if(val.contains(aAcX)){
          String strNew = val.replaceFirst(aAcX, "");
          AcXant = AcX;
          aAX1 = Float.parseFloat(strNew);
      }else if(val.contains(aAcY)){
          String strNew = val.replaceFirst(aAcY, "");
          AcYant = AcY;
          aAY1 = Float.parseFloat(strNew);
      }else if(val.contains(aAcZ)){
           String strNew = val.replaceFirst(aAcZ, "");
           AcZant = AcZ;
           aAZ1 = Float.parseFloat(strNew);
      }else if(val.contains(aAcX2)){
          String strNew = val.replaceFirst(aAcX2, "");
          aAX2 = Float.parseFloat(strNew);
      }else if(val.contains(aAcY2)){
```

```
        String strNew = val.replaceFirst(aAcY2, "");
        aAY2 = Float.parseFloat(strNew);
    }else if(val.contains(aAcZ2)){
         String strNew = val.replaceFirst(aAcZ2, "");
         aAZ2 = Float.parseFloat(strNew);
    }else if(val.contains(aAcX3)){
        String strNew = val.replaceFirst(aAcX3, "");
        aAX3 = Float.parseFloat(strNew);
    }else if(val.contains(aAcY3)){
        String strNew = val.replaceFirst(aAcY3, "");
        aAY3 = Float.parseFloat(strNew);
    }else if(val.contains(aAcZ3)){
         String strNew = val.replaceFirst(aAcZ3, "");
         aAZ3 = Float.parseFloat(strNew);
    }



  }

  ////////////////////sensor 1
/*  VXant = VX;
  VX = intVelocidad(AcXant,AcX,VXant,dt);

  VYant = VY;
  VY = intVelocidad(AcYant,AcY,VYant,dt);

  VZant = VZ;
  VZ = intVelocidad(AcZant,AcZ,VZant,dt);

  PXant = PX;
  PX = intPosicion(VXant,VX,PXant,dt);

  PYant = PY;
  PY = intPosicion(VYant,VY,PYant,dt);

  PZant = PZ;
  PZ = intPosicion(VZant,VZ,PZant,dt);
  */

  ////////////////////// sensor 2
  ////////////////// sensor 3

  float spX1 = pos1 + round(largo1*sin(aAX1/57.3));
  float spY1 = pos1 + round(largo1*cos(aAX1/57.3));

  float spX2 = spX1 + round(largo1*sin(aAX2/57.3));
```

```
  float spY2 = spY1 + round(largo1*cos(aAX2/57.3));

  float spX3 = spX2 + round(largo1*sin(aAX3/57.3));
  float spY3 = spY2 + round(largo1*cos(aAX3/57.3));

println(val); //print it out in the console
  //text(aAX1,10,10);
  //text(aAY1,10,30); //text(AcYant,100,30); text(VY,300,30); text(PY,400,30);
  //text(aAZ1,10,50);//text(AcZant,100,50); text(VZ,300,50); text(PZ,400,50);
  line(300,0,300,600);
  line(0,300,600,300);
  //String sData =
"("+PX+","+PY+","+PZ+")"+"("+VX+","+VY+","+VZ+")"+"("+AcX+","+AcY+","+AcZ
+")";
  ellipse(pos1,pos1,10f,10f);
  fill(204,102,0);
  ellipse(spX1,spY1,10f,10f);
  ellipse(spX2,spY2,10f,10f);
  ellipse(spX3,spY3,10f,10f);
  //text(sData,PX+300+20,PY+300);
  //ellipse(300,300,10f,10f);
  line(spX1,spY1,spX2,spY2);
  line(spX2,spY2,spX3,spY3);
  line(pos1,pos1,spX1,spY1);

  text(spX1-pos1,50,35);
  text(spX2-pos1,140,35);
  text(spX3-pos1,230,35);
  text(spY1-pos1,50,55);
  text(spY2-pos1,140,55);
  text(spY3-pos1,230,55);
  text("Punto 1",55,10);
  text("Punto 2",145,10);
  text("Punto 3",235,10);
  text("X:",10,35);
  text("Y:",10,55);
 }


public float intVelocidad(float Aant,float Aact,float Vant,float dt){
  return  Vant + (0.5*(Aant-Aact)*dt);
}

public float intPosicion(float Vant,float Vact,float Pant,float dt){
  return  Pant + (0.5*(Vant-Vact)*dt);
}
```