PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DESARROLLO DE UN SISTEMA DE ADQUISICIÓN PARA LA EVALUACIÓN DEL BALANCE CORPORAL EN BASE A LA MEDIDA DEL MOVIMIENTO DEL TRONCO

Tesis para obtener el título profesional de Ingeniero Electrónico

AUTOR:

JUAN MARCOS LUNA JARAMILLO

ASESORES:

MsC. Rocio Liliana Callupe Perez

Dr. Damián Eleazar Sal y Rosas Celi

Lima, Octubre del 2019

RESUMEN

Durante el control del balance corporal es el cerebro quién, por medio de impulsos nerviosos transmitidos hacia los músculos de la columna, activa el movimiento de una persona generando un cambio en su posición. Sin embargo, con el paso de los años, dicha respuesta frente a este tipo de control se ve mermada, produciendo retrasos al momento en que la persona puede reaccionar frente a cambios repentinos respecto de su posición de equilibrio. Dicho decaimiento en la velocidad de reacción frente a una pérdida repentina del equilibrio corporal puede generar la ocurrencia de caídas, las cuales podrían derivar en consecuencias negativas.

Bajo este enfoque, el Laboratorio de Investigación en Biomecánica y Robótica Aplicada (LIBRA) en conjunto con el investigador Dr. Peter Reeves (Sumaq Life LLC) propusieron la necesidad de conocer la velocidad a la que las personas pueden responder al realizar movimientos predecibles e impredecibles que los obliguen a perder su posición de equilibrio.

La presente tesis desarrolla un sistema electrónico que permite medir, en tiempo real, el ángulo del movimiento del tronco mientras la persona realiza movimientos laterales (izquierda o derecha), los cuales son propuestos por una interfaz gráfica sobre la que se llevará a cabo el test de evaluación. Asimismo, con la necesidad de replicar dicha evaluación en distintos lugares se diseñó un sistema portable, el cual incluye todo el sistema contenido dentro de un único dispositivo lo cual facilita el desarrollo de la prueba.

En el desarrollo de la presente, se utilizó un sensor inercial MPU6050 cuya señal es adquirida y procesada por un Microcontrolador Arduino Nano. Para el procesamiento de dicha señal se comparó el desempeño de un filtro Complementario y un filtro Kalman, ampliamente utilizados en este tipo de información. Dicho desempeño fue medido utilizando el sistema de referencia Mikrolar R3000 Rotopod. La interfaz gráfica fue desarrollada utilizando el microcontrolador Raspberry Pi 3B, el cual recibe el dato del movimiento realizado y muestra dicha información en la pantalla, esto con la finalidad que la persona conozca su desempeño actual. Finalmente, se asignó vital importancia al análisis del tiempo de respuesta del sistema completo, así como la búsqueda de reducir dicho tiempo de respuesta tal que no influya en el desempeño de la persona evaluada. Los tiempos obtenidos al final del desarrollo de la presente fueron de aprox. 30.2ms utilizando el filtro Complementario y de aprox. 25.2ms aplicando el filtro de Kalman, con lo cual se puede evidenciar que el sistema, en ambos casos, responde de forma óptima.

A mis padres y a mis hermanos por ser parte importante de mi vida, por apoyarme durante toda esta etapa y confiar en mí en todo momento.

AGRADECIMIENTOS

A la profesora Rocio Callupe, por brindarme su apoyo aceptando ser mi asesora, y por todas las recomendaciones que me brindó en este corto periodo. Al profesor Damián Sal y Rosas, por haberme asesorado durante todo este tiempo brindándome sus consejos y experiencias. Al Dr. Dante Elías, por confiar en mí y permitirme formar parte del Laboratorio de Investigación en Biomecánica y Robótica Aplicada (LIBRA) para llevar a cabo este enriquecedor proyecto. Asimismo, a cada uno de los miembros del Laboratorio por brindarme su apoyo a lo largo del desarrollo de la presente.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Durante la vida diaria, las personas desarrollan diversos tipos de actividades físicas, las cuales exigen que el cuerpo humano responda de forma óptima frente a los requerimientos exigidos por cada tipo de actividad. Obtener dicha respuesta requiere que la persona presente un buen control sobre su equilibrio corporal. Dicho control del equilibrio es producido mediante el intercambio de impulsos nervioso desde el cerebro hacia los músculos de la columna vertebral. Es en estos últimos donde se produce el movimiento de la columna frente a alguna orden proveniente del cerebro. Sin embargo, con el paso de los años, se ha evidenciado que dichos músculos evidencian un deterioro en sus características físicas, con lo cual se reduce la respuesta que estos puedan presentar frente a un cambio de movimiento fortuito ocurrido en cierto instante de tiempo. La respuesta de dichos músculos puede medirse por la velocidad a la que estos actúan frente a algún estímulo que los obligue a cambiar de posición.

Con el desarrollo de la tecnología existen diversos dispositivos que permiten medir la velocidad con la que una persona logra controlar su equilibrio corporal frente a un cambio de posición. Sin embargo, dichos sistemas focalizan sus mediciones considerando todo el cuerpo como un único sistema, lo cual no permite conocer de forma específica como contribuye cierto músculo, sobre dicha respuesta. Asimismo, dichos dispositivos son sistemas de muy elevado costo, lo cual dificulta la masificación de este tipo de evaluaciones.

El investigador Dr. Peter Reeves (Sumaq Life LLC), propuso una forma de medir dicha velocidad de respuesta, focalizando la evaluación, únicamente, en la columna vertebral. Su método de análisis consiste en modelar la respuesta de la persona como un sistema de control clásico, donde la persona actúa como la planta del sistema, mientras que, el controlador del sistema se define como el control que la persona presenta sobre su balance corporal. Asimismo, el método de evaluación propuesto, en este caso, consiste en medir que tan rápido la posición de la persona (salida del sistema) logra seguir la posición de un objetivo visual (entrada de referencia del sistema).

Por lo tanto, el objetivo del presente trabajo consiste en diseñar e implementar un sistema de adquisición que permita medir, en tiempo real, los ángulos de movimiento de la columna vertebral de una persona al moverse en el plano frontal; es decir, hacia la izquierda o a la derecha. Dichas posiciones laterales serán indicadas por una interfaz gráfica con la que la persona interactuará durante la prueba. Asimismo, el dispositivo debe ser capaz de ser un sistema portable, lo cual permitirá replicar dichas pruebas en más lugares, así como, ser un

sistema de bajo costo en comparación con las tecnologías actualmente existentes en el mundo. Esto con la finalidad de facilitar el acceso a este tipo de tecnologías de evaluación.

CAPITULO 1: MARCO PROBLEMÁTICO DEL ÁNALISIS DEL BALANCE CORPORAL

1.1. PROBLEMÁTICA DE LA INVESTIGACIÓN

En su vida diaria las personas necesitan realizar diversas actividades como caminar, correr, moverse de un lado a otro, saltar, etc. Dichas actividades requieren que la persona se encuentre en la condición física adecuada para poder llevarlas a cabo sin evidenciar perjuicio alguno en su salud. No obstante, llevar a cabo dichas tareas por adultos mayores, o por personas que padecen alguna discapacidad motriz, resulta ser, en ocasiones, una labor sumamente compleja. Dichas condiciones, manifiestan problemas en la columna vertebral, lo que conlleva a la reducción de la capacidad del balance corporal.

La columna vertebral opera en conjunto con el cerebro para realizar las acciones pertinentes al momento de llevar a cabo diversas tareas. Este enlace es el que permite conservar el equilibrio del cuerpo durante la realización de diversas actividades que realice la persona. Mantener un buen balance corporal es un factor crítico al momento de realizar toda clase de movimientos, ya que de no darse de forma correcta podrían ocurrir caídas que podrían afectar gravemente a aquellas personas que la padezcan.

Según estudios llevados a cabo en EEUU, las caídas representan un alto índice de atención en el departamento de emergencias, especialmente en pacientes adultos mayores [1]. Asimismo, según estadísticas tomadas por brainline.org existen alrededor de 2.8 millones de personas que padecieron lesiones cerebrales traumáticas en 2013. Siendo alrededor de 1.3 millones de estos accidentes ocasionados por la ocurrencia de caídas, lo que lo posiciona como la causa principal del desarrollo de este padecimiento [2].

Las caídas ocurren por la pérdida de la estabilidad del cuerpo, siendo esta ocasionada por un ineficiente control del equilibrio por parte de la persona. Dicha ineficiencia durante el control del balance corporal es ocasionada por problemas ocurridos durante el intercambio de información entre el cerebro y la columna. Este hecho se manifiesta por la ocurrencia de retrasos durante la transmisión de información, desde el cerebro hacia la columna, que impiden controlar la postura de la persona oportunamente. Asimismo, dichos retrasos en la activación del sistema lumbar predisponen a las personas a sufrir problemas de postura en algún futuro cercano [3]. Resulta importante evaluar los tiempos que le demanda a la

columna vertebral responder frente a un estímulo externo al que la persona se encuentra sometida durante la realización de alguna actividad. Estos tiempos son conocidos como **delay**, tiempo que demora en iniciarse el movimiento, y **lag**, tiempo que demora en ejecutarse el movimiento desde que se inició [3].

La columna vertebral posee tres tipos de movimiento, de flexión/extensión (adelante/atrás), de inclinación (izquierda/derecha) y de rotación (giro sobre el eje del tronco) como se muestra en la Figura 1.1. La mayoría de las veces que una persona evidencia una caída, ésta es producida por alguna perturbación ocurrida al realizar movimientos de inclinación [4]. Dichos movimientos son los que requieren mayor esfuerzo para restablecer el equilibrio del cuerpo para evitar la ocurrencia de caídas. Debido a esto resulta importante conocer la velocidad con la que una persona puede contrarrestar las pérdidas de equilibrio que se evidencian al llevar cabo esta clase de movimientos.

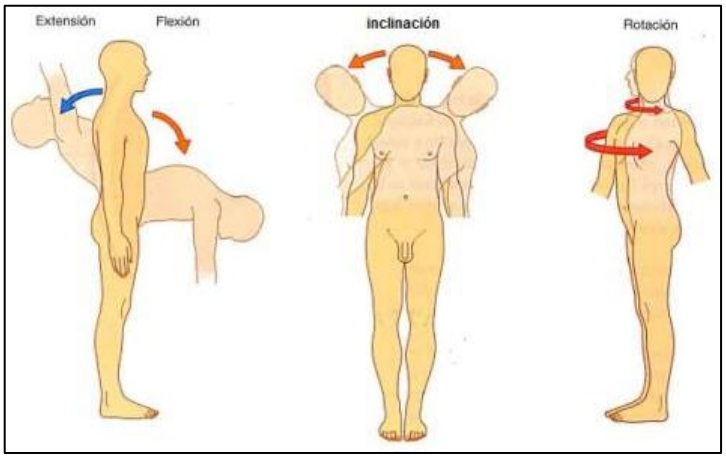


Figura 1.1: Movimientos de la columna vertebral [25]

1.2. ESTADO DEL ARTE EN EL ANÁLISIS DEL BALANCE CORPORAL

En la actualidad existen diversos sistemas que permiten evaluar la velocidad y eficiencia con la que una persona tiende a controlar su balance corporal. Con dicha información se pueden identificar patrones que permitan conocer de forma más amplia la reacción que presenta una persona frente a la ocurrencia de cambios de posición involuntarios que, en ocasiones, provocaría la ocurrencia de caídas. En ese sentido, un gran número de estos dispositivos se caracterizan porque realizan las evaluaciones correspondientes utilizando plataformas de fuerza. Estas plataformas cuentan con sensores de fuerza que permiten medir los cambios de presión que ejerce el centro de masa de la persona evaluada sobre estos, tal que le sea posible conservar el equilibrio y, con esta información, identificar la

trayectoria seguida por el cuerpo de la persona frente a los estímulos a los que se encuentra sometido durante el desarrollo de la prueba.

- Dynamic Balance Assesment & Rehab Systems (NeuroCom)

  Este sistema es muy usado en el ámbito clínico para llevar a cabo evaluaciones y tratamientos enfocados en el balance corporal en pacientes con dificultades motrices o problemas de equilibrio. Asimismo, este sistema permite llevar a cabo entrenamientos para mejorar las condiciones atléticas del individuo. Durante el desarrollo de la prueba, el paciente se ubica sobre una plataforma de fuerza en la cual intenta imitar los movimientos que se le indican en la pantalla con la que está interactuando. El sistema, basándose en la presión ejercida sobre la plataforma en cada momento, brinda información del control del equilibrio de la persona evaluada. La Figura 1.2 detalla los modos en que se aplica este sistema. La información obtenida de dichas evaluaciones es usada para diseñar tratamientos de rehabilitación o programas de entrenamiento enfocados para un paciente en específico [5].

Figura 1.2: Sistema de evaluación Neurocom Balance Master [5].

Cabe resaltar que dichas tecnologías evalúan el balance corporal del individuo modelando el cuerpo humano como una sola pieza. Sin embargo, es necesario tener un enfoque más específico de la respuesta de las partes del cuerpo que se dedican exclusivamente a controlar el balance corporal. Dicho esto, se vuelve necesario evaluar de forma exclusiva como actúa el sistema lumbar durante el control del equilibrio. En base a esta información, se pueden adquirir mayores conocimientos acerca de los factores principales que conllevan a una persona a sufrir una caída.

En la actualidad se ha evidenciado el desarrollo de diversas técnicas que, en comparación con las plataformas de fuerza, permiten evaluar de forma específica el movimiento de una parte específica del cuerpo con gran precisión. Por un lado, tal como es mostrado en [6], [7], los sistemas de captura de movimiento por medio de video, tales como los dispositivos VICON, o Kinect de Microsoft, extienden su uso en el área del análisis y validación del movimiento mientras se intenta conservar el balance corporal. Esta técnica utiliza cámaras que visualizan marcadores colocados alrededor del cuerpo de la persona. Por medio de estos, es posible obtener un avatar virtual que replica los movimientos ejecutados por la persona en las áreas del cuerpo donde se ubican los marcadores. La Figura 1.3 muestra la implementación de un sistema de video para la captura del movimiento mediante el uso de cámaras VICON.

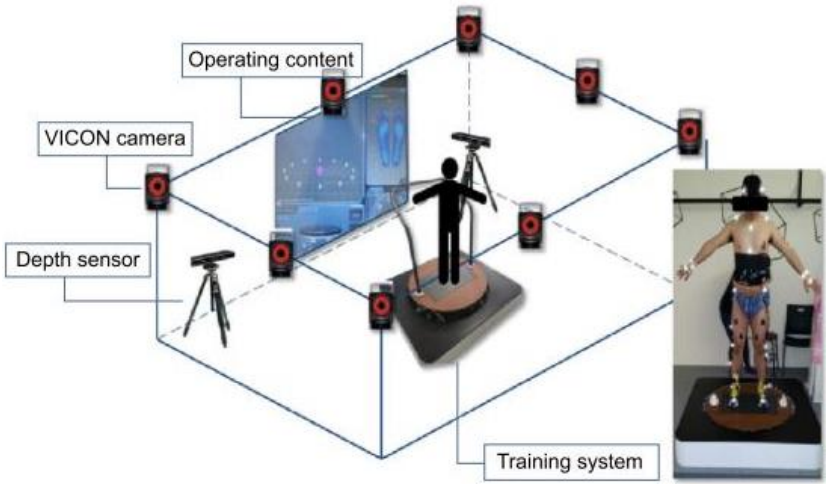


Figura 1.3: Sistema de captura de movimiento basado en video [7]

Por otro lado, los sistemas inerciales están extendiendo su aplicación en el área del análisis del movimiento debido a que son sensores pequeños en tamaño, pero de gran eficiencia y de muy bajo costo en comparación con las tecnologías previamente descritas. En base a su tamaño son muy usados en aplicaciones ‘wireless’, permitiendo el desarrollo de sistemas portables para captura de movimiento y el análisis de patrones específicos durante la realización de alguna actividad, tales como caminar, correr, etc. Estos sistemas contienen acelerómetros, giroscopios o magnetómetros, los cuales, al trabajar de manera individual son susceptibles errores producido por la naturaleza propia de cada sensor. Por dicha razón, en la actualidad se utilizan algoritmos, conocidos como ‘Sensor Fusion’, que permiten fusionar la data adquirida por cada uno de estos sensores mediante la utilización de algoritmos matemáticos. De esta forma se puede estimar, de forma precisa, la posición del cuerpo sobre el cual fue colocado. Tal es así que [8]–[11] basan su investigación del análisis

del movimiento en la utilización de este tipo de sensores, obteniendo resultados satisfactorios, tal como es mostrado en la Figura 1.4.

Figura 1.4: Sensores inerciales para el comando de un avatar virtual [9]

Actualmente existe una investigación realizada en conjunto por investigadores del Laboratorio de Investigación en Biomecánica y Robótica Aplicada (LIBRA PUCP) y el investigador Dr. Peter Reeves (CEO Sumaq Life LLC) [12]. Esta investigación diseñó una tecnología que permitía evaluar de forma específica el comportamiento realizado por la columna vertebral al reaccionar frente a diversos estímulos visuales a los cuales era sometido durante el desarrollo de la prueba. Este sistema se basaba en la interacción de la persona evaluada con una interfaz gráfica, el cual tenía como finalidad el seguimiento de la posición de un objetivo realizando movimientos de flexión(adelante) y extensión(atrás), en tiempo real. Durante la evaluación, la persona evaluada utilizaba una vestimenta metálica la cual lo obligaba a mantenerse en una posición erguida. Esta condición era necesaria ya que una variación en su posición (como el encogimiento de los hombros) alteraría las mediciones requeridas. Dicha vestimenta se conectaba a unos potenciómetros lineales, los cuales, dependiendo del movimiento realizado (flexión/extensión) por la persona, permitían conocer el ángulo correspondiente a dicho movimiento, tal como se muestra en la Figura 1.5. Asimismo, dicha información se almacenaba en tiempo real para que, una vez culminada la prueba, toda la información sea procesada y puedan estimarse los parámetros requeridos por dicha investigación.

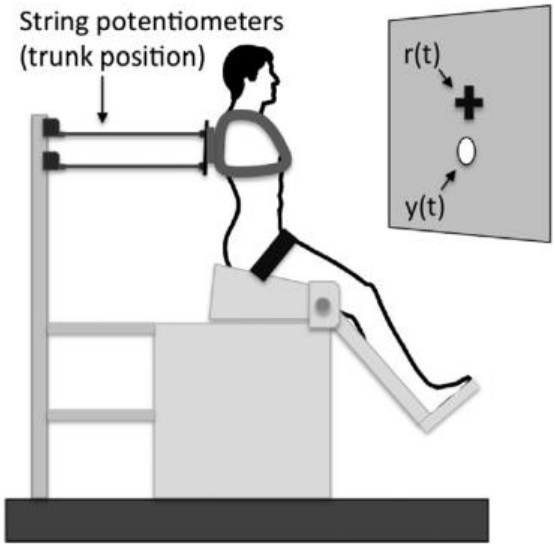


Figura 1.5: Sistema de adquisición de ángulos de inclinación basado en potenciómetros [3]

Este trabajo establece las bases para evaluar la capacidad con la que actúa la persona para mantener su balance corporal con la finalidad de evitar las pérdidas de equilibrio críticas. El desempeño mostrado durante la prueba es modelado como un sistema de entrada – salida [3], tal como es mostrado en la Figura 1.6. En dicho modelamiento se muestra la planta (P, cuerpo de la persona) y el controlador del sistema (K, control del balance corporal). Asimismo, y(t) representa el movimiento del tronco de la persona al seguir la posición objetivo, r(t). Dicho seguimiento debe ocurrir de la forma más rápida y precisa posible en busca de analizar la capacidad de la persona para responder a dicho movimiento. Siendo la finalidad del sistema obtener y(t)=r(t).

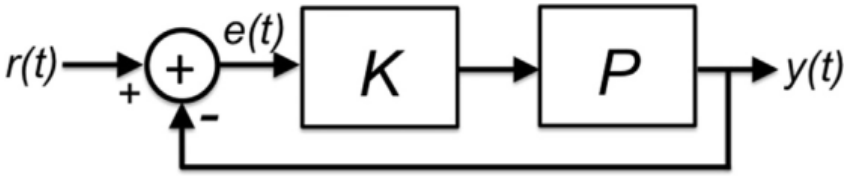


Figura 1.6: Sistema de entrada-salida [3]

Modelar la forma en que actúa la persona para conserva su equilibrio, de esta manera, permite identificar los tiempos que le demanda realizar dicha acción de forma óptima.

1.3. JUSTIFICACIÓN

Los sistemas de evaluación del balance corporal que existen en la actualidad manifiestan desventajas que complican el impacto que estos puedan tener en investigaciones posteriores. En primer lugar, la tecnología utilizada en el diseño de estos dispositivos genera que sus costos sean muy elevados. Este hecho genera que solo un pequeño número de centros terapéuticos o de investigación tengan la facilidad de adquirir alguno de estos dispositivos de evaluación. En segundo lugar, la realización de los estudios pertinentes puede convertirse en una complicación, ya que los dispositivos existentes para dicho fin son sistemas no portables. Este hecho obliga a que las personas acudan al centro correspondiente para ser sometidos a las pruebas, lo cual puede resultar contraproducente para los intereses de las personas que serán evaluadas.

En base a lo expuesto, surge la necesidad de contar con un sistema que permita realizar la evaluación pertinente de forma que contrarreste las desventajas de los dispositivos ya existentes, sin dejar de ser eficiente.

1.4. OBJETIVOS

1.4.1. Objetivo General

El objetivo general de la presente tesis consiste en desarrollar un sistema que sea portable y de bajo costo, tal que permita medir en tiempo real el arco de movimiento del tronco al realizar movimientos laterales ante un estímulo proveniente de una interfaz gráfica. Asimismo, dicha información debe ser almacenada, en el instante en que ocurre, para utilizarse en la estimación de parámetros que permitan identificar la capacidad de balance corporal de la persona evaluada.

La data adquirida por el sistema será información sumamente valiosa para futuras investigaciones que requieran analizar el comportamiento de una persona al ser sometido a diversos cambios de posición.

1.4.2. Objetivos Específicos

Los objetivos específicos que plantean alcanzarse en este estudio son:

1. Seleccionar el sensor que permita medir de forma apropiada el movimiento del tronco de una persona sin añadir latencia considerable que pueda degenerar la información requerida.
2. Diseñar e implementar la estructura electrónica para la medición y almacenamiento en tiempo real del movimiento del tronco corporal de una persona.
3. Programar la interfaz gráfica(GUI) con la que interactuará una persona durante la prueba.
4. Optimizar los tiempos de respuesta del sistema de adquisición.

CAPITULO 2: ANÁLISIS DE TECNOLOGÍAS PARA EL DISEÑO DEL SISTEMA DE MEDICIÓN PARA EVALUACIÓN DEL BALANCE CORPORAL

2.1. DETALLE Y ANÁLISIS DEL OBJETIVO GENERAL

Como se mostró en el capítulo anterior, el objetivo general consiste en implementar un sistema portable que permita medir el arco de movimiento del tronco al realizar movimientos de inclinación (laterales, hacia la izquierda y hacia la derecha). Estos movimientos serán realizados mientras el usuario interactúa con una interfaz gráfica, la cual desplegará un objetivo visual cuya trayectoria debe ser seguida. Asimismo, este sistema debe tener la capacidad de capturar, en tiempo real, dichas mediciones y almacenarlas para su posterior evaluación. La Figura 2.1 muestra la idea de diseño que se tiene para el objetivo planteado, la cual se basa en 3 módulos.

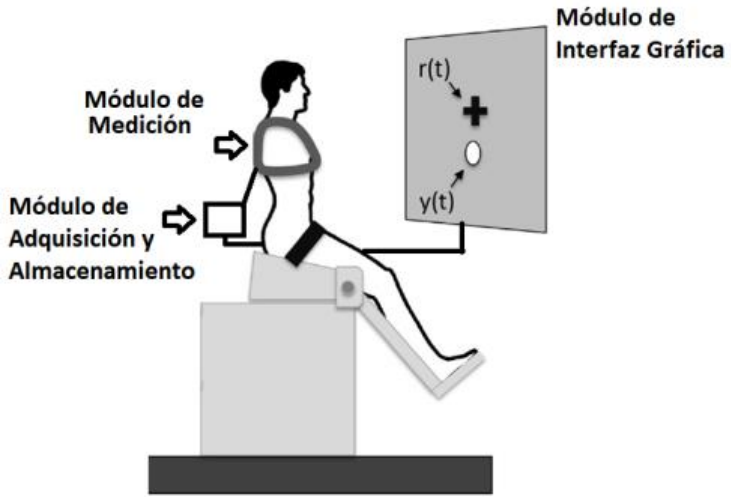


Figura 2.1: Sistema de adquisición esperado. Imagen modificada de [3]

La Figura 2.1 esquematiza el objetivo general del presente trabajo, de la cual se puede deducir que los tres módulos principales que tendrá el sistema son los siguientes.

- Módulo de Interfaz Gráfica
- Módulo de medición de movimiento del tronco corporal
- Módulo de adquisición y almacenamiento

Debido a que se busca que el sistema a implementar sea portable, se debe lograr que todas las partes se encuentren incluidas dentro del mismo dispositivo.

A continuación, se procederá a explicar más a detalle cada módulo del sistema, así como las tecnologías que permitirán llevar a cabo su implementación.

2.1.1. Módulo de Interfaz Gráfica

Desplegar la evaluación en una interfaz gráfica requiere el uso de tecnologías de transmisión de imágenes o video. Dichas tecnologías pueden ser analógicas o digitales. La elección entre una u otra se basa en factores como resolución de imagen, compatibilidad de equipos, costo, etc.

En el desarrollo del presente dispositivo se ha propuesto utilizar tecnologías ya existentes en el mercado, siendo las más resaltantes y de fácil acceso los estándares de video VGA y HDMI, cuyas características son detalladas en la Tabla 2.1.

Tabla 2. 1: Comparación de tecnologías de Audio y Video [26]–[28].

	Estándares de transmisión	
Características	VGA	HDMI
Tipo de tecnología	Analógico	Digital
Información transmitida	Video	Audio y Video
Resolución de imagen o video	320x200, 720x480, hasta 1920x1080 bits	Alcanza niveles HD, Full HD, 4K y hasta 8K en dispositivos de mayor procesamiento
Inmudidad al ruido del canal	Susceptible a factores como calidad y longitud del cable	Alta inmudidad al ser un formato digital
Practicidad	Requiere cableado adicional para transmisión audio analógico	Solo un canal digital para la transferencia de audio y video digital
Alcance	Usado a nivel mundial debido a su compatibilidad con diversos dispositivos	En actual expansión debido a presentar mejores características que sus predecesoras
Costos	Alrededor de $4	Alrededor de $10

A partir de la comparación realizada, se elige el estándar HDMI, el cual, al tener una alta tasa de resolución de imagen, permitirá brindarle al usuario una mejor experiencia visual al medir el arco de movimiento del tronco mientras se realizan movimientos laterales, y sin generar interferencias que puedan afectar su desempeño.

2.1.2. Módulo de medición de movimiento del tronco corporal

Durante la prueba, el usuario deberá mover su tronco corporal, realizando movimientos laterales, para seguir la posición del objetivo desplegado en pantalla. Resulta necesario medir la posición a la que se encuentra el tronco en un instante específico para conocer de forma precisa como éste se comporta al realizar el seguimiento. En la actualidad existen diferentes tecnologías que permiten tener acceso a la información de posición

requerida. Dichas tecnologías son comparadas en la Tabla 2.2 con la finalidad de elegir la mejor opción a utilizar en la presente tesis.

Tabla 2.2: Comparación de tecnologías para la medición de movimiento

<table>
<tr><th></th><th colspan="4">Tecnologías de medición</th></tr>
<tr><th></th><th>Sistemas mecánicos</th><th>Sistemas magnéticos</th><th>Sistemas de video</th><th>Sistemas inerciales</th></tr>
<tr><td rowspan="4">VENTAJAS</td><td>-Elementos de poca complejidad para ser usados.</td><td>-Sus componentes no se desgastan debido a que no hay fricción entre ellos.</td><td>-Usado de forma continua en la industria de captura de movimiento.</td><td>-Cuenta con sensores acelerómetro, giroscopio y magnetómetro que le permite conocer valores de aceleración, velocidad angular e intensidad del campo magnético, respectivamente.</td></tr>
<tr><td rowspan="3">-Bajo precio.</td><td>-Sensores de tamaño diminuto que brindan portabilidad al dispositivo.</td><td>-Gran precisión ya que utiliza algoritmos complejos para convertir la imagen en información de posición.</td><td>-Sensores de tamaño diminuto que brindan portabilidad al dispositivo.</td></tr>
<tr><td>-Gran precisión al no ser sometido a interferencias externas.</td><td rowspan="2">-Costo relativamente mayor al resto.</td><td rowspan="2">-Bajo costo y gran precisión.</td></tr>
<tr><td>-Bajo precio.</td></tr>
<tr><td rowspan="3">DESVENTAJAS</td><td>-Expuestos al desgaste mecánico producido por la fricción entre sus componentes.</td><td rowspan="3">- Susceptible a interferencias si existiese algún campo magnético externo entre el sensor y el imán.</td><td>-No suelen ser sistemas portables ya que requieren ubicar estratégicamente cada dispositivo.</td><td>- Requieren estar correctamente calibrados para tener la posición deseada.</td></tr>
<tr><td>-Requiere elementos altamente calibrados para lograr una gran precisión.</td><td rowspan="2">- Son sistemas relativamente costosos en comparación con las anteriores debido a que usa mejores tecnologías.</td><td>-Utiliza sensores (acelerómetro, giroscopio, magnetómetro) susceptibles a errores de medición.</td></tr>
<tr><td></td><td>-Dependen de la resolución de sus componentes para brindar una óptima medición.</td></tr>
</table>

En base a lo presentado en la Tabla 2.2 se pueden notar las ventajas y desventajas de las diversas tecnologías que surgen como alternativa para llevar a cabo la medición requerida. No obstante, debido a las necesidades de portabilidad, bajo costo, y confiabilidad en la medición que se tienen en la presente tesis, se ha optado por utilizar los sistemas inerciales. Debido a esto se expondrá, más a detalle, el porqué de la elección realizada.

Sensores inerciales: Son sensores que se basan en el principio de la inercia para realizar sus mediciones. Están compuestos por unidades denominadas MEMS (Micro Electro Mechanical Systems), los cuales permiten tener sensores de tamaños diminutos. Este tipo de unidades son muy utilizados para la navegación autónoma, así como sistemas de seguimiento en unidades robóticas.

Los sensores inerciales se conocen mayormente como sensores IMU (*Inertial Measurement Unit*) los cuales integran acelerómetros, giroscopios y magnetómetros en una misma unidad. Con dichos módulos se puede tener un conocimiento preciso, dependiendo de la calidad del sensor, de parámetros como aceleración, velocidad angular e intensidad de campo magnético, respectivamente. De esta forma, se puede tener información de posición, desplazamiento y orientación de un objeto, lo cual es útil para el seguimiento o captura de movimiento [13].

Asimismo, al trabajar con sensores inerciales se deben considerar diversos aspectos que pueden recaer en fuentes grandes de error. Estos efectos ya son conocidos y, en la actualidad, existen técnicas que permiten eliminar el efecto contraproducente que éstas representan en las mediciones de los sensores. Estas fuentes de error provienen en principio de:

- ✓ Acelerómetro: Este sensor brinda mediciones precisas de la aceleración al realizar cambios de posición lentos. Sin embargo, al moverse rápidamente, se manifiesta errores en dicha medición. De lo cual se deduce que el sistema es susceptible a las altas frecuencias [14].
- ✓ Giroscopio: Este sensor brinda medidas de la velocidad angular, por lo que para calcular el ángulo girado se requiere realizar una integral. Este hecho añade un error aleatorio respecto al ángulo inicial, el cual se acumula generando errores considerables de medición en grandes periodos temporales, lo que le imposibilita regresar al valor de offset inicial. Este error es conocido como "drift" [15].

En la actualidad existen diversas técnicas que permiten eliminar dichas fuentes de error. Estas técnicas se basan en la fusión de las señales adquiridas por cada sensor que compone una IMU (acelerómetro, giroscopio y magnetómetro) mediante algoritmos matemáticos. Dichas técnicas son conocidas como "Sensor Fusion", cuya calidad de la señal de salida depende exclusivamente de la complejidad y eficiencia del algoritmo

elaborado, así como de la eficacia de la unidad de procesamiento y de la calidad de la señal brindada por los sensores. Entre estas técnicas destacan las siguientes:

- Filtro Complementario: Es un filtro que, como su nombre lo dice, complementa las medidas del acelerómetro y giroscopio. Este aplica un filtro pasa-bajos a la señal del acelerómetro, para eliminar el error producido en los cambios rápidos de posición. Y a la vez, aplica un filtro pasa-altos a la señal proveniente del giroscopio con lo cual se elimina el error acumulativo que se genera en este en largos periodos de tiempo. La ecuación 2.1 muestra la fórmula del Filtro Complementario utilizado para calcular el ángulo requerido, θ, siendo α el factor de fidelidad entre la lectura del giroscopio versus la lectura del acelerómetro ($0 < \alpha < 1$).

  $$\theta_{estimado} = \alpha * \left(\theta_{estimado} + Veloc.angular_{gyro} * dt\right) + (1 - \alpha) * \theta_{accel} \quad (2.1)$$

  De esta forma, se tendrá una medición confiable dependiendo de los parámetros de dicho filtro [15][16][17]. La Figura 2.2 esquematiza el funcionamiento de un Filtro Complementario.

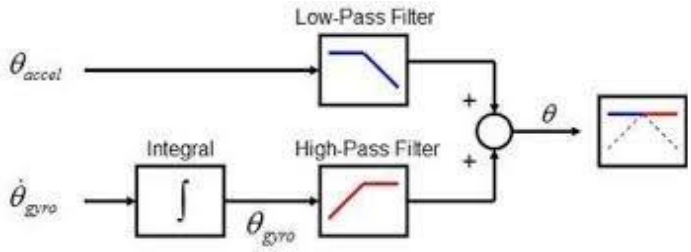


Figura 2.2: Filtro complementario [17]

- Filtro Kalman: Es un filtro que opera como un observador de estados del sistema evaluado. La entrada del sistema consiste de la señal deseada más el ruido adicional propio del sensor, todo esto incluido en un mismo dato, por lo que conocer el valor de la señal sin ruido es no medible. Sin embargo, haciendo uso de un observador de estados, es posible estimar dicho valor deseado utilizando la señal de entrada del sistema. El filtro Kalman funciona como el observador del sistema, y a su vez, añade una ganancia, conocida como Ganancia de Kalman, con la cual se busca reducir el error existente entre el valor estimado por dicho filtro y el valor real del sistema. Dicho resultado realimenta a dicho filtro para estimar el siguiente valor respectivo[18]. La Figura 2.3 muestra las ecuaciones del Filtro Kalman, y a la vez esquematiza su funcionamiento.

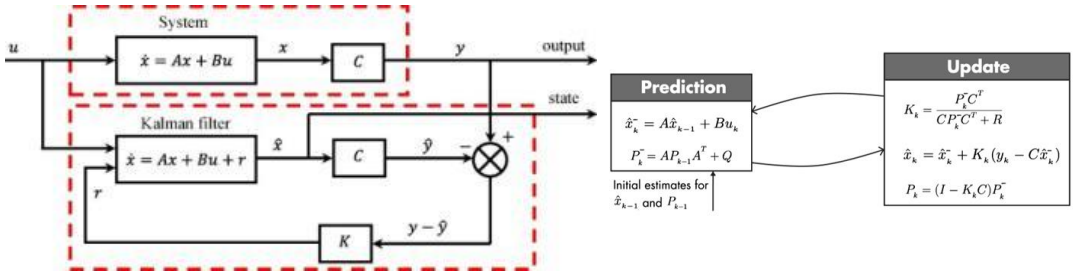
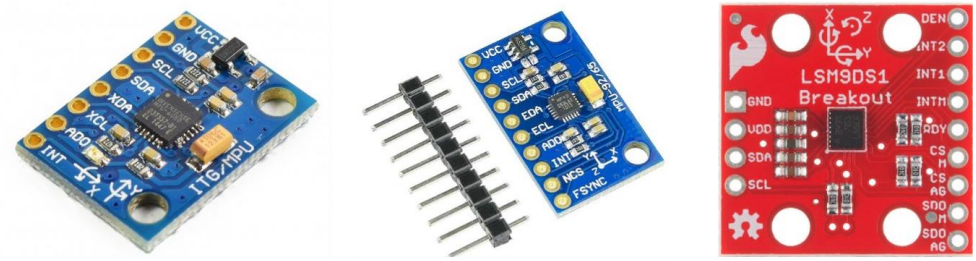


Figura 2.3: Filtro Kalman [18]

En el mercado existe una vasta variedad de sensores inerciales, cada uno con diferentes características centradas principalmente en sus grados de libertad, sensores integrados, precisión, resolución, formato de comunicación, y costo. En la Tabla 2.3 se presenta una comparación de tres sensores inerciales, mostrados en la Figura 2.4 que fueron escogidos dentro de todos los existentes en el mercado, debido principalmente a su costo de adquisición.



a) b) c)

Figura 2.4: a) Sensor MPU6050 [29]. b) Sensor MPU9250 [30]. c) Sensor LSM9DS1 [31]

Tabla 2.3: Comparación sensores inerciales

	Invensense MPU6050	**Invensense MPU9250**	**LSM9DS1**
Tensión de alimentación	5 V, 3.3V	3.3 V	
Grados de Libertad (DOF)	6	9	
Rango Acelerómetro (g)	2,4,8,16		
Resolución Acelerómetro (bits)	16		
Rango Giroscopio (dps)	250 - 2000		245 - 2000
Resolución Giroscopio (bits)	16		
Rango Magnetómetro (uT)	No tiene	4800	400, 800, 1200, 1600
Resolución Magnetómetro (bits)	--------	14	16
Interface de comunicación	I2C	I2C,SPI	
Costo	$5.87	$8.40	$15.95
Observaciones	Estos cuentan con un DMP (Digital Motion Processor) integrado que le permite realizar cálculos internamente de parámetros como ángulos Euler, Quaterniones, etc.		

Como se puede notar en la Tabla 2.3, las 3 opciones presentadas tienen características técnicas similares para llevar a cabo la medición. Sin embargo, la principal diferencia radica en los grados de libertad (DOF, cantidad de sensores en cada eje de medición) proporcionados por cada unidad inercial; así como en el precio. En ese sentido, se optó por utilizar el sensor MPU6050 debido principalmente a su precio, y a que tiene los grados de libertad suficientes para realizar la medición de la columna en el plano frontal.

2.1.3. Módulo de adquisición y de almacenamiento

La información brindada por el sensor debe ser adquirida y almacenada para su posterior evaluación. Resulta necesario utilizar un dispositivo que permita comunicarse con el sensor de forma eficiente y a la vez brinde opciones para almacenar los datos respectivos.

Debido a lo señalado se pueden encontrar opciones como microcontroladores e incluso pequeñas computadoras, cuyo factor determinante en la elección será la capacidad de almacenar toda la data requerida durante la prueba.

Las tecnologías que existen en la actualidad, y serán usadas en el desarrollo de las diversas soluciones, son detalladas a continuación:

A. Microcontroladores Arduino

Arduino es una plataforma de código abierto que se utiliza para la elaboración e implementación de proyectos electrónicos. Esta plataforma cuenta con tarjetas de evaluación de diferentes características diseñadas según los requerimientos del proyecto a desarrollar. Asimismo, cuenta con un software libre que permite la escritura de programas en código para montarlo sobre la placa, y que pueda ejecutarse.

Los tipos de tarjetas de evaluación más usados de esta familia de microcontroladores, son los siguientes:

Tabla 2.4: Comparación tarjetas Arduino [32]–[34]

	ARDUINO UNO	ARDUINO NANO	ARDUINO MEGA
Microcontrolador	ATmega328P	ATmega328	ATmega2560
Voltaje de operación	5V	5V	5V
Voltaje de entrada	7-12V	7-12V	7-12V
Pines I/O digitales	14	22	54
Pines I/O PWM digitales	6	6	15
Pines analógicos	6	8	16
Corriente por pin I/O	20 mA	40 mA	20 mA
Memoria Flash	32 KB	32 KB	256 KB
SRAM	2 KB	2 KB	8 KB
EEPROM	1 KB	1 KB	4 KB
Frecuencia de reloj	16 MHz	16 MHz	16 MHz

B. Raspberry Pi 3 B

Es una placa de evaluación diseñada por la fundación Raspberry. Esta placa cuenta con un sistema operativo propio, lo que le permite denominarse como una minicomputadora, debido a que cuenta con un hardware pequeño. Sobre esta placa se encuentra montado un procesador que le permite desarrollar tareas de alto performance, así como un chip dedicado para desplegar la parte gráfica y una memoria RAM suficiente para almacenar la data necesaria. La tarjeta Raspberry Pi 3 B cuenta con dos versiones, la versión B+ y la versión B estándar. A continuación, se mostrarán las características de esta última.

Características [19][20]:

- CPU Quad Core 1.2GHz Broadcom BCM2837 64bit
- 1GB RAM
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 puertos USB
- Salida de audio y video compuesto
- Full size HDMI
- Puerto Micro SD para cargar el sistema operativo y el almacenamiento de datos
- Alimentación tipo Micro USB de hasta 2.5 A

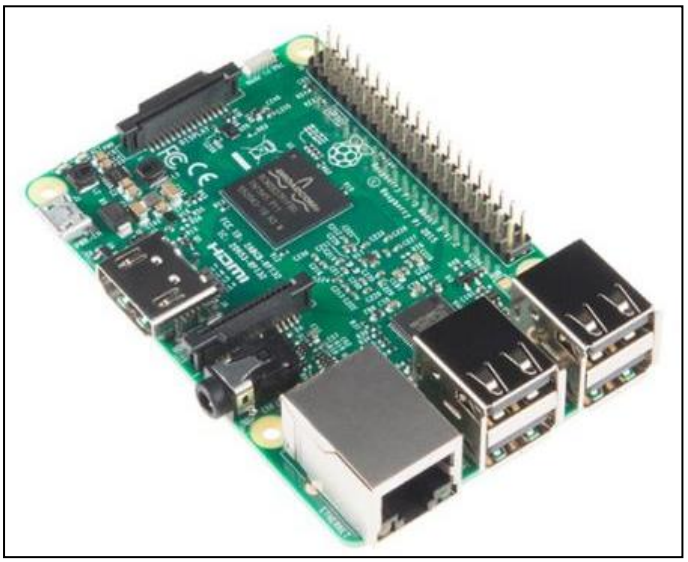


Figura 2.5: Raspberry Pi 3B [35]

2.2. MÉTODOS DE SOLUCIÓN

Durante el desarrollo de la presente tesis se implementaron diversos prototipos, empezando desde lo más simple, hasta alcanzar el diseño que responda de la forma deseada según los requerimientos que se procederán a detallar en la presente sección:

- Alta resolución de imagen para la interfaz utilizada
- Adquisición de datos en tiempo real
- Almacenamiento de datos en tiempo real
- Portabilidad del dispositivo

En la Tabla 2.5 se muestran las soluciones propuestas utilizando diferentes tecnologías. Asimismo, se muestran los problemas presentados por cada una de estas alternativas, y como la solución posterior puede subsanar dichas deficiencias identificadas.

Tabla 2.5: Alternativas de solución

	TECNOLOGÍAS USADAS	DESCRIPCIÓN
PRIMERA ALTERNATIVA DE SOLUCIÓN	ARDUINO UNO - DRIVER VGA + ARDUINO MEGA – SENSOR	- Resolución de video sumamente baja (120x60 pixeles). No es óptimo para llevar a cabo las pruebas. - Sin pérdida de datos.
SEGUNDA ALTERNATIVA DE SOLUCIÓN	RASPBERRY + SENSOR + HDMI	- Alta resolución de video utilizando el estándar HDMI. Permite el desarrollo de la prueba de forma fluida. - Almacenamiento ineficiente provocando la ocurrencia de pérdida de datos.
ALTERNATIVA DE SOLUCIÓN FINAL	RASPBERRY - HDMI + ARDUINO -SENSOR + TARJETA SD	- Alta resolución de video utilizando el estándar HDMI. Permite el desarrollo de la prueba de forma fluida. - Almacenamiento eficiente sin pérdida de datos.

Asimismo, en las siguientes figuras se muestran los prototipos diseñados y detallados en la Tabla 2.5.

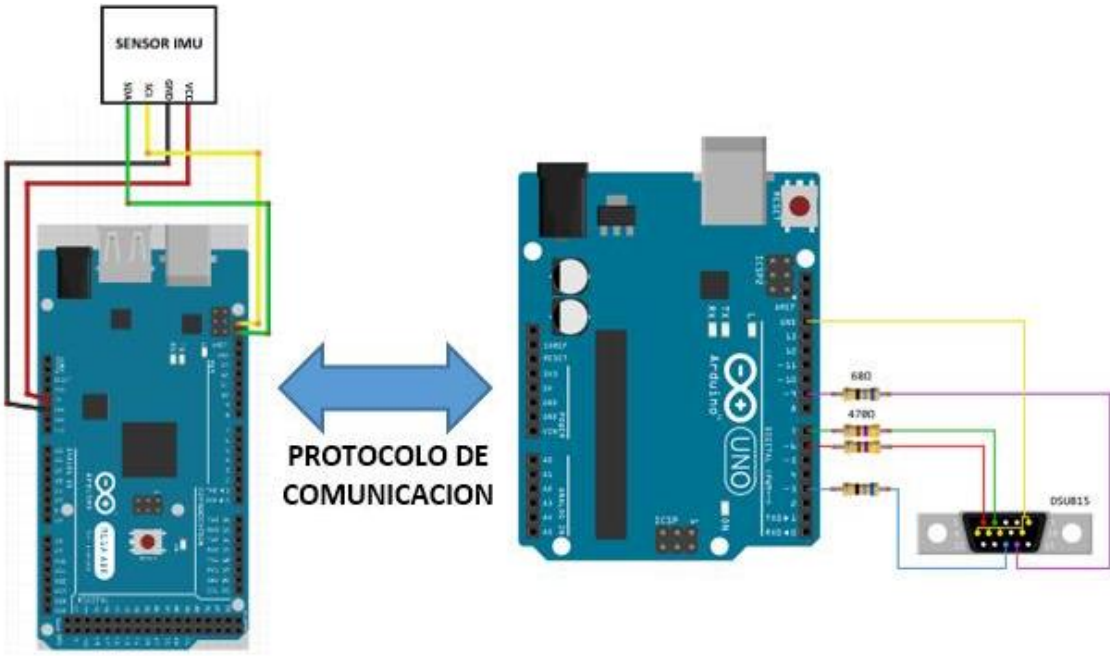


Figura 2.6: Primera alternativa de solución

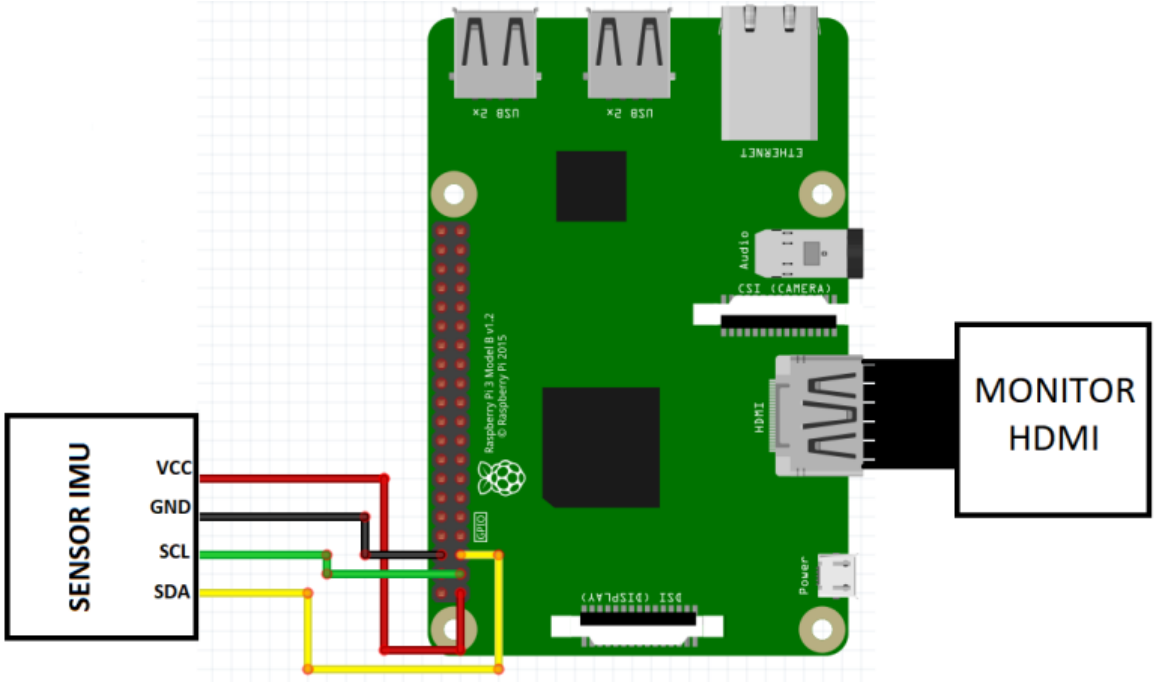


Figura 2.7: Segunda alternativa de solución

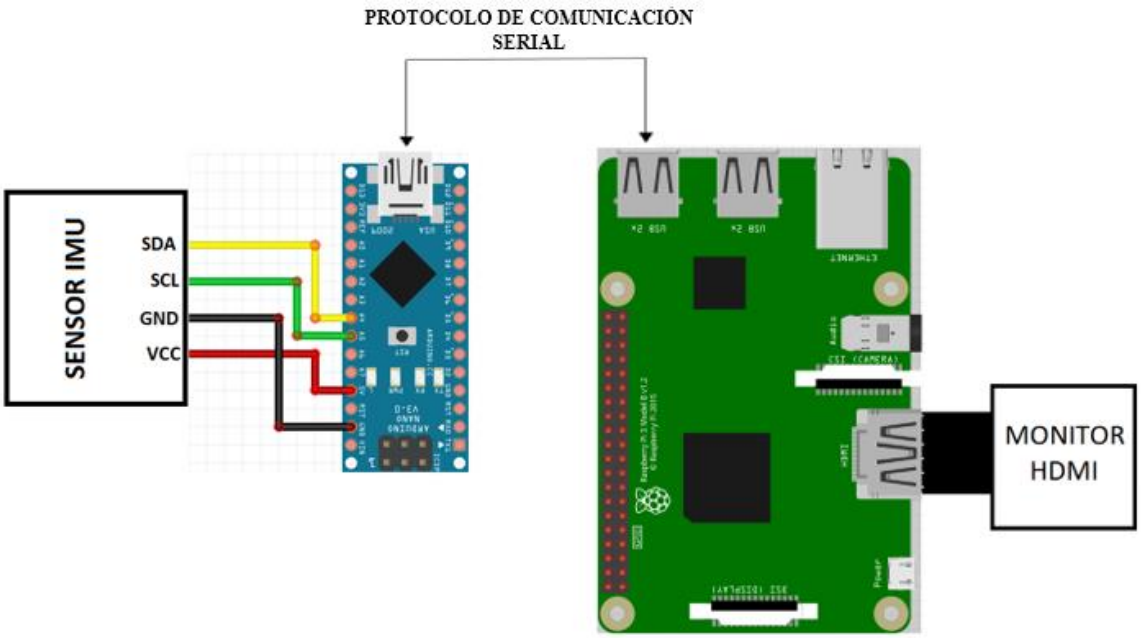


Figura 2.8: Alternativa de solución final

Como se puede notar en la Tabla 2.5, la 'Alternativa de solución final' es el prototipo que mejor responde frente a los requerimientos de calidad de visualización de la interfaz y confiabilidad en el almacenamiento de datos, siendo este último explicado con más detalle en la sección 3.3.3. Esta alternativa se diseñó con la finalidad de superar las falencias presentadas por las propuestas previas y obtener el rendimiento requerido. Es así que el modelo elegido para desarrollar en los siguientes capítulos será el denominado 'Alternativa de solución Final'. Asimismo, las versiones previas a la alternativa de solución Final son explicadas con mayor detalle en la sección A de Anexos.

CAPITULO 3: DESARROLLO DEL SISTEMA DE ADQUISICIÓN PARA LA EVALUACIÓN DEL BALANCE CORPORAL

3.1. INTRODUCCIÓN

En el presente capítulo se presentará de forma más detallada el diseño propuesto para la elaboración del sistema de evaluación requerido, tomando en cuenta los requerimientos intrínsecos que este conlleva, tanto en el aspecto de hardware como en el software. Dichos requerimientos quedan definidos en base a la necesidad de diseñar un sistema de bajo costo, y que a la vez sea portable, permitiendo replicar la prueba en distintas localizaciones. En lo referente al software, es necesario considerar que la prueba a desarrollar debe estar contenida dentro del dispositivo construido, y toda la información recolectada durante ese periodo será almacenada en la unidad correspondiente para su posterior evaluación. Finalmente, para garantizar la confiabilidad del sistema, se debe asegurar que el retardo general del mismo sea el mínimo posible (aprox. 30-40ms) para no mermar el desempeño de la persona evaluada.

A lo largo de la tesis se identificarán los dos usuarios que intervienen de forma activa durante la evaluación:

- Persona evaluada: Persona que desarrolla la prueba.
- Evaluador: Persona a cargo de la prueba.

3.2. MODELO DE SOLUCIÓN

El capítulo anterior propuso la distribución del objetivo general basado en 3 módulos, cada uno de los cuales cumple una función específica pero que requiere de las demás para alcanzar la optimización del mismo. Dichos módulos presentados son: módulo de medición, módulo de adquisición y almacenamiento, y módulo de interfaz gráfica. La Figura 2.1 muestra dicha segmentación del problema.

A partir de lo mostrado en la Figura 2.1 surge la necesidad de plantear una solución que permita satisfacer el funcionamiento de los módulos mencionados. Es así que en la Figura 3.1 se muestra un diagrama de bloques que presenta de forma más clara la solución propuesta para el funcionamiento del sistema.

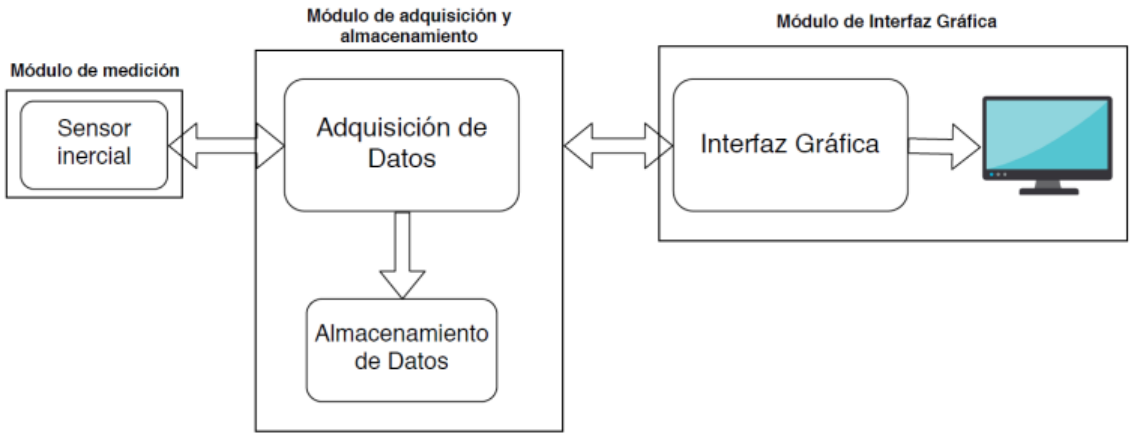


Figura 3.1: Diagrama de bloques del sistema

La Figura 3.1 presenta el diagrama de bloques solución propuesto para el cumplimiento de los requerimientos planteados por el objetivo general de la presente tesis en base a la estrategia presentada líneas arriba.

En principio, el módulo de medición será el encargado de leer el arco de movimiento del tronco de la persona evaluada, al realizar movimientos laterales: izquierda o derecha, y proporcionar la información equivalente proveniente de los acelerómetros y giroscopios en cada uno de sus ejes, los cuales, posteriormente, serán procesados, mediante un algoritmo matemático, para obtener dicha posición en formato de ángulos de rotación del tronco corporal. Como se mencionó en el capítulo 2, dicha función será realizada por una unidad de medición inercial. Dicho módulo debe ser capaz de comunicarse con el módulo de adquisición y almacenamiento, componiéndose de dos partes, la adquisición y el almacenamiento. Por un lado, el módulo de medición obtendrá datos del sensor, los cuales serán enviados a la unidad de Adquisición de Datos, la cual se encargará de asignarle el flujo correspondiente a los datos recibidos en cada instante según requiera el comportamiento del sistema. Asimismo, este módulo será el encargado de procesar la data de los sensores inerciales obteniendo, a partir de estos, el ángulo de rotación del tronco de la persona evaluada, al inclinarse hacia la izquierda o hacia la derecha.

Por otro lado, este módulo también cuenta con una unidad de Almacenamiento de Datos, en la cual se albergará la información procesada por la unidad de adquisición. Sin embargo, con lo descrito hasta ahora no se podría realizar una evaluación de forma correcta ya que no habría manera que la persona evaluada tenga conocimiento acerca de su desempeño en un instante dado mientras desarrolla la prueba. A partir de esto surge la necesidad de contar con un módulo de interfaz gráfica, el cual permitirá que la persona evaluada interactúe de

forma directa con la prueba. Dicho módulo acogerá la información procesada por la unidad de adquisición y la mostrará en una pantalla para que la persona evaluada reciba realimentación visual de su comportamiento y pueda tomar acciones sobre su desempeño.

3.3. DISEÑO DEL HARDWARE

A partir del detalle de la Figura 3.1 se plantea una solución como la mostrada en la Figura 3.2, la cual presenta más a detalle los elementos que permitirán cubrir las funciones de cada módulo descrito.

Los requerimientos que debe satisfacer el hardware del sistema son:

- Ser un sistema de bajo costo.
- Ser capaz de almacenar la información adquirida durante una prueba, la cual tendrá un tiempo de duración de aproximadamente 15min por persona.
- Ser un sistema portable, tal que permita su transporte para la realización de la prueba en diversas localidades.
- Tener la capacidad de desplegar una interfaz gráfica para llevar a cabo la prueba.

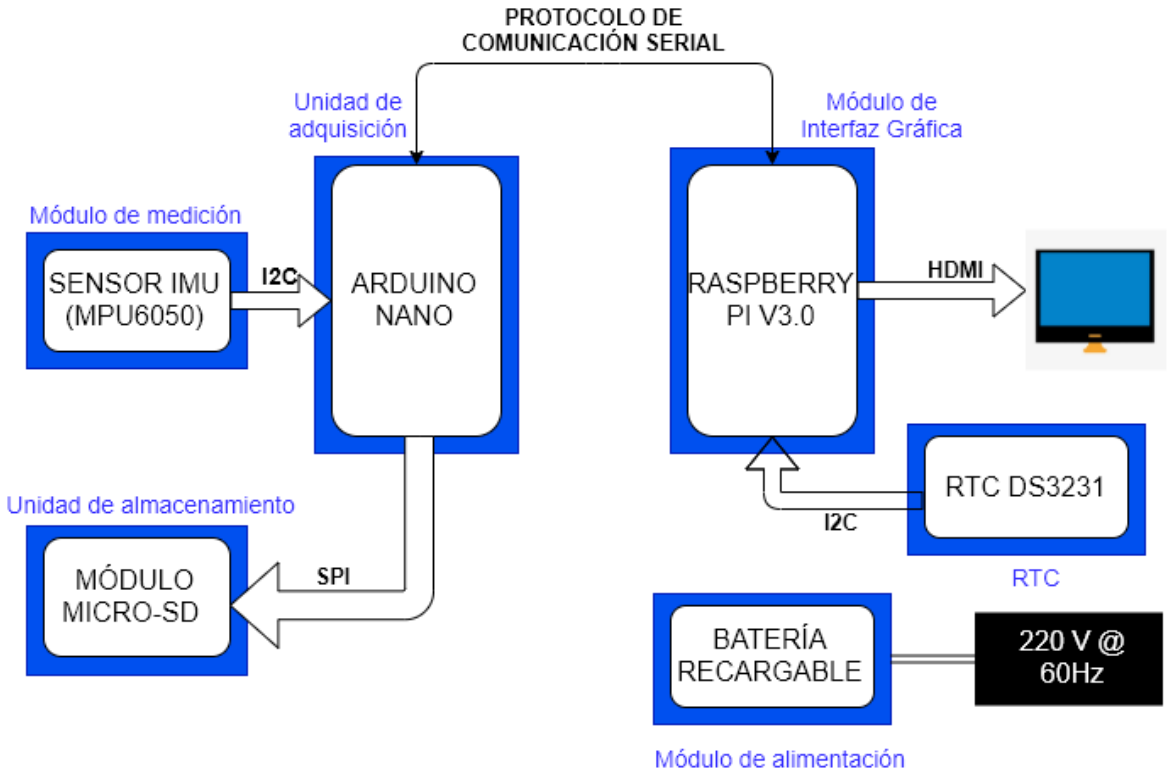


Figura 3.2: Prototipo de solución

3.3.1. Módulo de medición

Como se observa en la Figura 3.2, el módulo de medición estará formado por un sensor inercial, siendo el elegido para el propósito el MPU6050. Dicho sensor se seleccionó, ya que, para el estudio realizado, la persona evaluada rotará su tronco hacia la izquierda o hacia la derecha, según le sea indicado en pantalla. Dicho movimiento es mostrado en la Figura 3.3. Por dicho motivo, solo se requerirá conocer los ángulos de rotación alrededor del eje x, *roll angle*. Para dicho fin, un sensor de 6 DOF (*Degrees of Freedom*: número de sensores considerando todos sus ejes de medición) permite llevar a cabo dicha medición. Las características de dicho sensor son detalladas en la Tabla 3.1. Finalmente, un factor importante es que este módulo posee gran cantidad de información proveniente de la web que permite realimentar y optimizar su desempeño, hecho que lo distingue notoriamente de las demás unidades de medición propuestas.

Tabla 3.1: Características del sensor MPU6050

	MPU6050
Grados de Libertad	6 DOF
Sensores	Acelerómetro y Giroscopio, DMP
Resolución Acelerómetro	16 bits
Resolución Giroscopio	16 bits
Señal	Digital (I2C – Standard Mode(100KHz) and Fast Mode(400KHz))
Rango Acelerómetro	2g,4g,8g,16g
Rango Giroscopio	250 – 2000 DPS
Consumo de corriente Máximo (Acelerómetro, Giroscopio y DMP ON)	3.9 mA
Tensión de operación	3.3V, 5V

Tal como es mostrado en la Tabla 3.1 este sensor posee una resolución de 16 bits en cada uno de sus 6 DOF, 3 ejes del acelerómetro y 3 ejes del giroscopio. Asimismo, posee un rango de medición programable según los criterios de medición requerido por la aplicación. Dichas características son mostradas en la Tabla 3.2.

En base a la Tabla 3.2 se define el formato de configuración del sensor para el desarrollo de la tesis tal como lo muestra la Tabla 3.3.

Tabla 3.2: Resolución del sensor en cada rango [36]

	Full Scale Range (FSR)	Factor de Sensibilidad	Factor de resolución
Giroscopio	+-250 °/s	131 LSB/(°/s)	0.0038 °/s
	+-500 °/s	65.5 LSB/(°/s)	0.0076 °/s
	+-1000 °/s	32.8 LSB/(°/s)	0.0152 °/s
	+-2000 °/s	16.4 LSB/(°/s)	0.0305 °/s
Acelerómetro	+-2 g	16 384 LSB/g	0.0305 mg
	+-4 g	8192 LSB/g	0.0610 mg
	+-8 g	4096 LSB/g	0.1220 mg
	+-16 g	2048 LSB/g	0.2441 mg

Tabla 3.3: Configuración del sensor

Tensión de operación	5 V
Resolución Acelerómetro y Giroscopio	16 bits en cada eje
Rango de operación acelerómetro	2g
Rango de operación giroscopio	2000 DPS
Sensores utilizados	Acelerómetro y Giroscopio
Frecuencia de muestreo	200 Hz
Consumo de corriente	3.8 mA

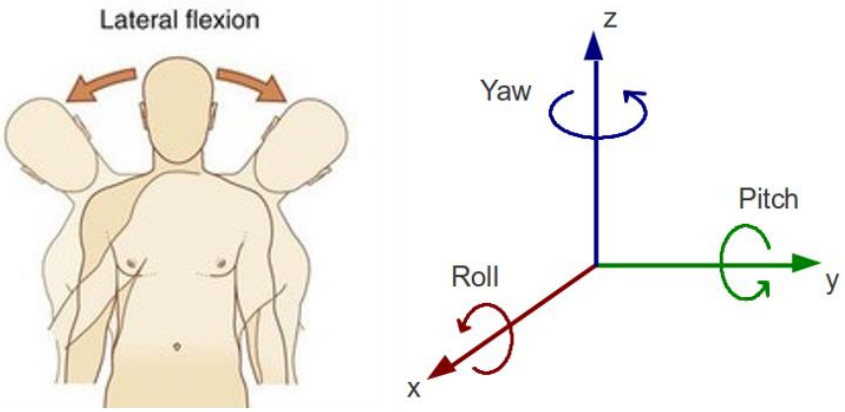


Figura 3.3: Movimiento realizado durante la prueba

3.3.2. Módulo de Interfaz Gráfica

Este módulo es el encargado de desplegar la interfaz gráfica con la que el usuario interactuará durante el desarrollo de la prueba. Para esto, consideraron una serie de requerimientos tal que la performance de la persona evaluada no se vea mermada por la calidad de gráficos de la interfaz. En ese sentido, dichos requerimientos son:

- Alta resolución de gráficos, mínimo 720pixeles
- Alta tasa de refresco de imagen, siendo el valor comercial entre 30 y 60Hz.
- Protocolo de video comercial, HDMI: elegido en el capítulo 2

En busca de satisfacer dichos requerimientos era necesario utilizar una unidad que permita llevar a cabo el procesamiento pertinente para el despliegue de la interfaz gráfica, pero sin dejar de satisfacer los principales objetivos como son la portabilidad y el bajo costo. En base a lo descrito, se optó por utilizar una Raspberry Pi 3B, cuyas características fueron descritas en el Capítulo 2 de la presente. A continuación, se resaltarán las características principales que serán utilizadas para el desarrollo del presente sistema.

- Procesador Quad Core 1.2 GHz BCM2837 64 bits
- GPU Dual Core VideoCore IV@ Multimedia Co-Processor. 1080p30 decode.
- 1GB RAM
- Conexión Ethernet, WIFI y Bluetooth 2.0.
- 4 puertos USB (500mA suministrado por cada puerto USB [21])
- Puerto Full Size HDMI
- Pines GPIO
- Sistema Operativo Raspbian basado en Linux
- Facilidad de programarse en lenguajes como Python, C, C++, Java y Ruby.
- Alimentación Micro USB de hasta 2.5A, consumiendo en operación normal 1A de forma continua.

En base a las características descritas la función cumplida por este módulo será fundamental en la operación del sistema. En primer lugar, puede ser programado en lenguajes de programación como Python, el cual cuenta con una librería denominada "Pygame" para llevar a cabo el desarrollo de una interfaz que sea amigable y llamativa, tal que permita llevar a cabo la evaluación del balance corporal.

Por otra parte, se puede utilizar el puerto de conexión HDMI para desplegar la interfaz gráfica en cualquier pantalla que disponga de este tipo de conexión. Este hecho brinda la posibilidad de poder realizar las pruebas en cualquier lugar, sin necesariamente permanecer en un establecimiento fijo, lo que permite satisfacer el término de portabilidad del sistema. Adicionalmente, este módulo cuenta con 4 puertos USB que permiten la conexión de otros dispositivos que no requieran cantidades elevadas de corriente (hasta 500mA por puerto [21]) para operar de forma correcta. Asimismo, este

módulo dispone de 40 pines GPIO que pueden ser utilizados para manejar diversos periféricos requeridos en cada caso [20], tal como se muestra en la Figura 3.4. Para este caso será utilizado el pin de 5V para servir de alimentación a los diversos periféricos que se vayan a utilizar durante el desarrollo. Esta configuración es realizada ya que dicho pin permite disponer de la corriente que suministra la batería del sistema trabajando a 5V, por lo que cualquier módulo podrá ser alimentado sin problemas a partir de esta conexión.

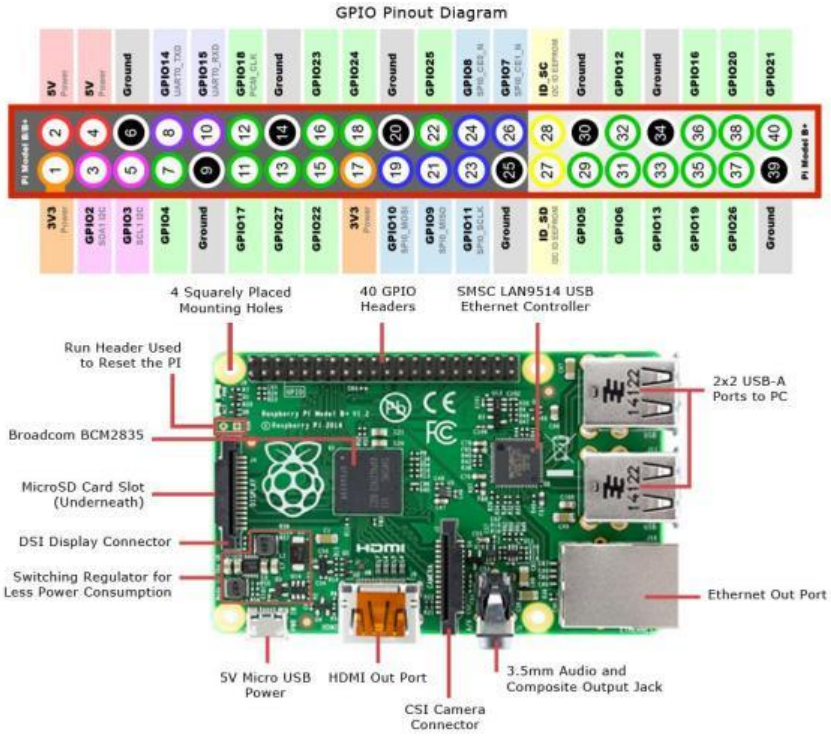


Figura 3.4: Raspberry Pi 3B Pinout [37]

Tal como se aprecia en la Figura 3.2, este módulo requiere de la información manejada por la unidad de adquisición para desplegarlo en pantalla. Para lograrlo, es necesario implementar un protocolo de comunicación entre ambos módulos, el cual será descrito en la etapa del diseño de software del presente capítulo.

Por otro lado, como se muestra en la Figura 3.2, el módulo Raspberry Pi se encuentra conectado a un reloj RTC el cual brindará la hora real al sistema. Dicha información cumplirá una función vital en la etiquetación de las pruebas. Este hecho se explicará con más detalle en la sección 3.4. Sin embargo, es necesario presentar las características técnicas propias del módulo RTC seleccionado, el cual fue elegido principalmente por

el formato de cuenta horaria proporcionado, siendo este un factor importante en la etiquetación de las pruebas.

Tabla 3.4: Características unidad RTC DS3231

Modelo RTC	DS3231
Tensión de operación	3.3V – 5V
Tensión de la batería (en caso de pérdida de suministro externo)	3.3 V
Formato de comunicación	Digital (I2C)
Formato de tiempo	12 horas o 24 horas
Formato de cuenta	Segundos, Minutos, Horas, Día, Mes, Año

La Tabla 3.4 describe las características presentadas por el módulo RTC que será utilizado en el presente trabajo, cuya conexión con la Raspberry Pi es mostrada en la Figura 3.5 usando el protocolo I2C.

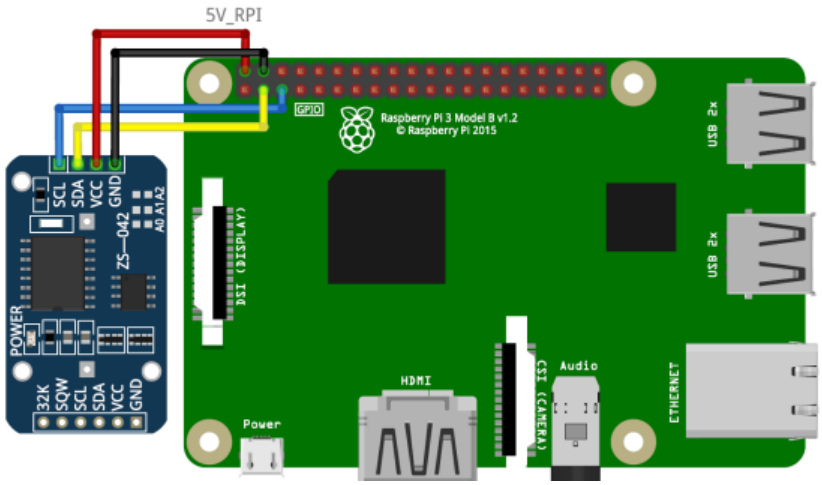


Figura 3.5: Conexión Raspberry Pi – Módulo RTC DS3231.

3.3.3. Módulo de adquisición y almacenamiento

Como se muestra en la Figura 3.2, este módulo está compuesto por dos unidades, las cuales permitirán leer la información proveniente del módulo de medición, procesar dicha data y almacenarlas para su posterior análisis matemático.

a. Unidad de Adquisición

Esta unidad será la encargada de manejar el flujo de información del sistema y asignarlo a cada unidad correspondiente en un momento dado según lo requiera el desarrollo de la prueba.

Por un lado, este bloque deberá recibir la data proveniente del módulo de medición y procesará dichos valores mediante algoritmos matemáticos en busca de obtener información útil para el desarrollo de la prueba (posición angular del tronco durante su movimiento). Por otro lado, la data procesada será enviada a la unidad de almacenamiento para conservar dicha información. Esto con la finalidad de brindar información a los evaluadores del comportamiento de la persona evaluada durante la prueba, en base al delay y lag. Asimismo, dicho parámetro será enviado al módulo de interfaz gráfica cuando sea requerido para su respectiva visualización en la pantalla.

Para cubrir las labores detalladas se eligió un módulo que pudiera realizar dichas tareas y comunicarse de forma eficiente con cada unidad que compone el sistema. Como se muestra en la Figura 3.2, esta labor será cubierta por un módulo Arduino Nano, el cual cuenta con una frecuencia de operación de 16 MHz, pines GPIO, y permite llevar a cabo la transferencia de información en base a los distintos protocolos de comunicación existentes en la actualidad (I2C, SPI, Serial). Este microcontrolador ha sido explicado con mayor detalle en la Tabla 2.4.

Como se muestra en la Figura 3.2, mostrar la información del sensor en la pantalla requiere diseñar un protocolo de comunicación entre el módulo de adquisición y el módulo de interfaz gráfica tal que se logre garantizar el envío y recepción oportuno de datos entre ambos módulos. Para satisfacer este requerimiento se implementará un protocolo de comunicación serial utilizando los puertos USB existentes tanto en el módulo Arduino Nano como en la Raspberry Pi. Asimismo, es posible aprovechar este formato de conexión para energizar al módulo Arduino Nano la sección 3.3.2, los puertos USB de la Raspberry Pi 3B suministran 5V@500mA cada uno, lo cual reduce la cantidad de corriente que el módulo Arduino puede suministrar por pin (40mA, 800mA considerando todos sus pines). Este hecho limita al Arduino para operar de forma correcta con la totalidad de sus pines. Empero, según la configuración propuesta, es posible alimentar al Arduino Nano utilizando el puerto USB de la Raspberry Pi 3B, ya que en la presente tesis solo se utilizarán los pines de I2C (A4-SDA y A5-SCL), así como los pines SPI (13-SCK, 12-MISO, 11-MOSI

y 4-CS). Dicha conexión para la alimentación del Arduino Nano por medio del puerto USB de la Raspberry Pi es mostrada en la Figura 3.6.

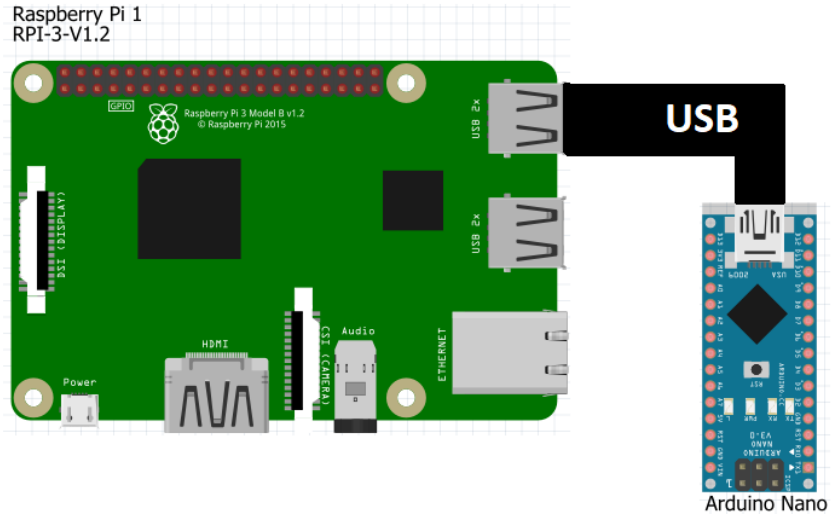


Figura 3.6: Conexión Arduino Nano – Raspberry Pi 3B

b. Unidad de Almacenamiento

Esta unidad se encargará de almacenar la información procesada y manipulada por la unidad de adquisición, la misma que es obtenida durante el desarrollo de la prueba. Para cumplir esta función se eligió usar el Módulo MicroSD. Este módulo permite guardar la información necesaria en una unidad externa de almacenamiento, tal como una tarjeta MicroSD o SDHC existentes en el mercado actual, cuya elección se basa en la cantidad de información de la que se desee disponer. La Tabla 3.5 describe las características que presenta este módulo.

Tabla 3.5: Características del módulo MicroSD

Tensión de operación	5V
Corriente de entrada	Mín: 0.2mA, Típico: 80mA, Máx: 200mA
Formatos de SD Card	Micro SD Card, Micro SDHC Card
Comunicación	Standard SPI interface
Dimensiones	42x24x12mm

La Figura 3.7 muestra el diagrama de conexiones correspondientes a dicho módulo.

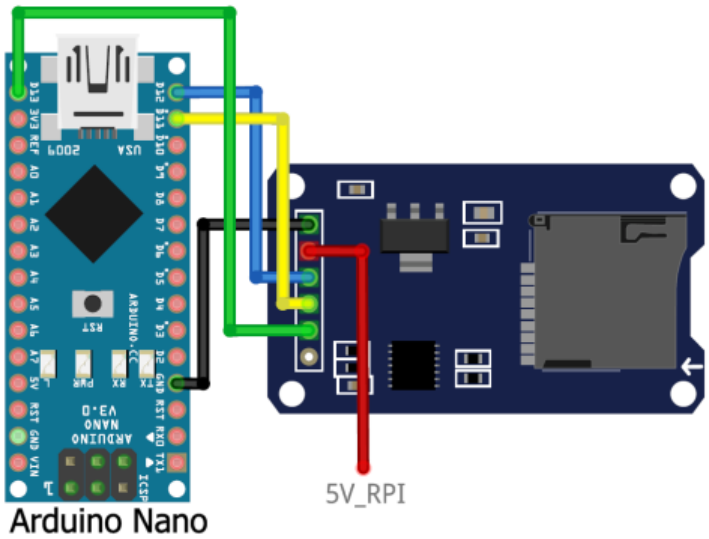


Figura 3.7: Conexión Arduino Nano y módulo MicroSD

Asimismo, se explicará la razón por la que se decidió utilizar una unidad de almacenamiento externa, en lugar de aprovechar elementos ya presentes en el sistema, tales como el Arduino NANO o la tarjeta Raspberry Pi. Sin embargo, a continuación, se detallará cuáles fueron las condiciones ofrecidas por ambas opciones.

- Arduino NANO

  Tal como se explicó en la sección 2.1.3, este microcontrolador posee una memoria RAM que le permite el almacenamiento de variables o datos necesarios que se utilizarán durante el programa. Sin embargo, la capacidad de almacenamiento de dicha memoria es muy reducida (2KB de memoria RAM) en comparación con la cantidad de información que se desea almacenar perteneciente a la duración de toda la prueba (>> 2KB).

- Raspberry Pi 3B

  Al igual que el microcontrolador Arduino, esta unidad también posee una memoria RAM, de 1GB de capacidad, que le permite almacenar la data necesaria, tales como variables utilizadas durante el programa. Asimismo, al ser un pequeño computador permite almacenar archivos de cualquier tipo, tales como documentos de texto, hojas de Excel, scripts, etc., en la memoria en la

que se encuentra cargado su Sistema Operativo. Esta opción brinda una gran facilidad para el almacenamiento de la información deseada.

A pesar de lo mencionado no se utilizó esta opción, ya que la Raspberry, al imprimir los datos en la pantalla, trabaja en una interrupción que no le permitía realizar ninguna acción adicional, por lo que se veía limitado en cuanto a la cantidad de información que podía recibir proveniente de la unidad de adquisición.

De acuerdo a lo ofrecido por cada una de las dos opciones presentadas, se eligió utilizar la unidad de almacenamiento externa mencionada, ya que tiene mayor capacidad de acopio (según la tarjeta MicroSD utilizada), y a la vez, no interrumpe la operación del sistema.

3.3.4. Módulo de alimentación

Este módulo está diseñado tomando como referencia la cantidad de corriente que consume el sistema completo, así como para brindar la autonomía suficiente al producto desarrollado.

La Tabla 3.6 resume las cantidades de corriente necesaria por cada uno de los módulos utilizados, según sus respectivas hojas de datos.

Tabla 3.6: Requerimientos de corriente

	Componente	Corriente consumida (mA)
SUMINISTRO RASPBERRY	Idle Raspberry Pi (incluido mouse y keyboard)	500
	HDMI	50
	Arduino Nano (USB)	45
SUMINISTRO PIN 5V	SD Card Module	80
	RTC Module	0.3
	MPU6050	3.8
Corriente de Consumo Total Requerida		679.1

La Tabla 3.6 muestra que los componentes pertenecientes al bloques ‘SUMINISTRO PIN 5V’ operan a una tensión de 5V DC, por lo cual se utilizará el pin de 5V propio de la Raspberry Pi 3B. Asimismo, es necesario considerar que dicho pin puede suministrar la cantidad de corriente expresada por la siguiente ecuación.

$$Corriente\ pin\ 5V = Corriente\ de\ alimentación\ RPI - (Corriente\ suministro\ Raspberry)$$

Entonces, sumando los valores proporcionados en la Tabla 3.6, se calcula que la cantidad de corriente requerida por el pin de 5V es de 84,1mA. De la misma forma, la corriente requerida del propio suministro de la Raspberry Pi es aproximadamente 595mA. Con estos valores, la corriente total que consumirá el dispositivo será aproximadamente 680mA al trabajar de manera continua.

Luego de haber definido la corriente requerida por los componentes citados en la Tabla 3.6, resulta necesario definir la batería que se utilizará a partir de la duración requerida para el sistema. En ese sentido, se define que las pruebas realizadas tendrán una duración aproximada de 30min por persona. Asimismo, durante el día se planea evaluar aproximadamente a 10 personas. Con dichos datos, se puede deducir que el sistema debe presentar una autonomía aproximada de:

$$30_{min/persona} * 10_{personas/dia} = 300_{min/día} = 5_{horas/dia}$$

Luego, se calcula:

$$Capacidad\ Batería = Cant.horas * corriente\ requerida$$

$$Capacidad\ Batería\ mínimo = 5h * 0.68A = 3.4Ah = 3400mAh$$

El modelo de batería elegida es BTL 606090 3.7v 4000mAh, cuyas características son mostradas en la Tabla 3.7, pudiendo observar dicha batería en la Figura 3.8, con un voltaje de operación de 3.7VDC.

Tabla 3.7: Características Batería

Tipo de Batería	Li-Po Battery Rechargeable
Voltaje de operación	3.7 V
Corriente de suministro	4000mAh

Figura 3.8: Batería modelo BTL 606090 3.7v 4000mAh

Sin embargo, la Raspberry Pi requiere una alimentación de 5VDC. Para ello es necesario amplificar dicho nivel de tensión en busca de obtener los 5V deseados que energicen el sistema. Con la finalidad de realizar dicha acción, se utilizó el módulo Adafruit Power Booster 1000C (ver Figura 3.9). Esta unidad utiliza una fuente conmutada tipo Booster para elevar el nivel de la tensión de entrada (batería) de 3.7V a 5V a su salida, lo cual es necesario para la alimentación de la Raspberry. La Tabla 3.8 muestra un resumen de las características presentadas por dicho módulo. Asimismo, la Figura 3.10 muestra el digrama de conexión entre la batería y el Power Booster 1000C.

Tabla 3.8: Características Adafruit Power Booster 1000C [38]

Nivel de tensión de entrada	3.2 - 4.2 V
Nivel de tensión de salida	5.2V
Corriente de entrada máxima (carga)	1000mA
Corriente de salida	1A - 2A(switch interno)
Corriente pico admitida	2.5A
Corriente de reposo	Habilitado y led encendido: 5mA Deshabilitado y led apagado: 20uA
Indicadores	Batería Baja(<3.2V), Batería Cargando, Batería cargada, Encendido
Conectores	Mini USB para la carga, USB para la salida de 5V
Características	Admite alimentación al sistema mientras se carga

Figura 3.9: Adafruit Power Booster 1000C [39]

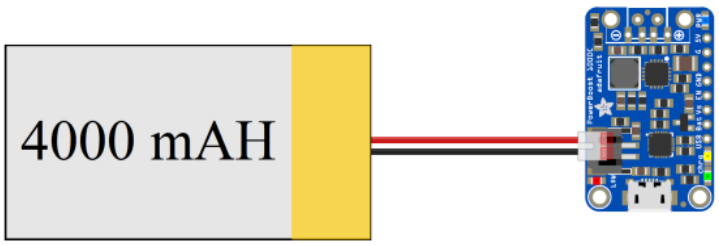


Figura 3.10: Diagrama de conexión Batería – Adafruit Power Booster 1000C

3.3.5. Diseño del Circuito Impreso

En la Figura 3.11 se muestra el diagrama de conexiones general del sistema que se requiere implementar. Se ha diseñado una tarjeta que se colocará sobre la Raspberry Pi para aprovechar el espacio. Ambas secciones se encontrarán unidas por medio de espadines que permitirán a la placa diseñada adherirse a los espadines macho propios de los pines GPIO de la Raspberry. Asimismo, la Raspberry Pi energizará los diferentes componentes empleados por medio de su pin de 5V. De esta forma se aprovechará la corriente suministrada por la batería del sistema.

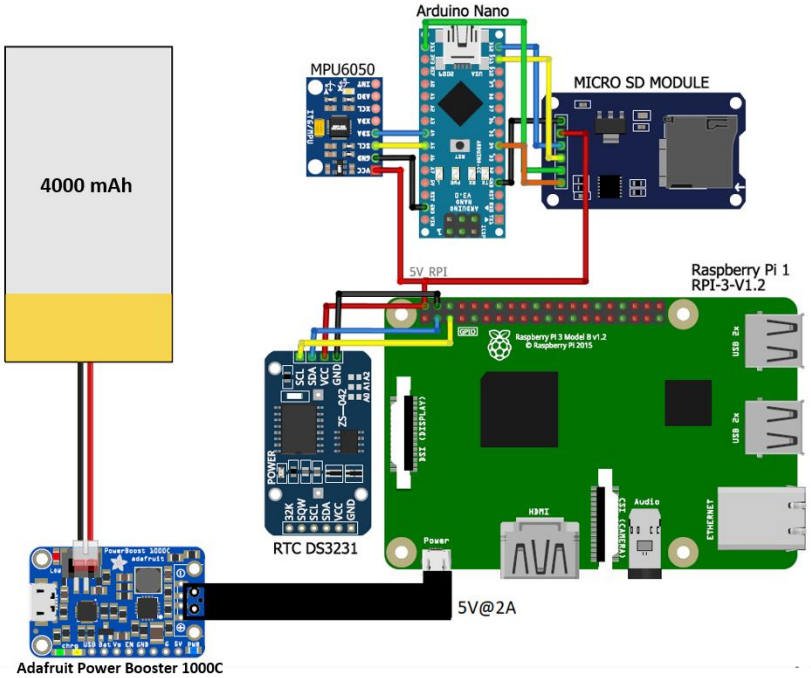


Figura 3.11: Diagrama de conexiones del circuito completo

En la Figura 3.12 se muestra el diagrama esquemático del circuito diseñado en Eagle, así como su respectivo layout, Figura 3.13, que evidencia la construcción de un módulo que se superpone a la Raspberry.

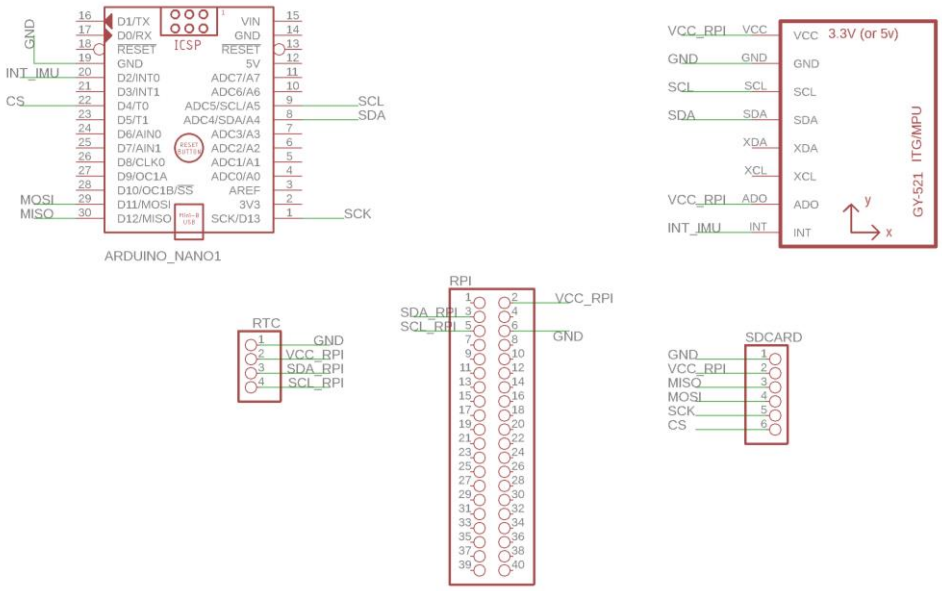


Figura 3.12: Diagrama esquemático del circuito impreso

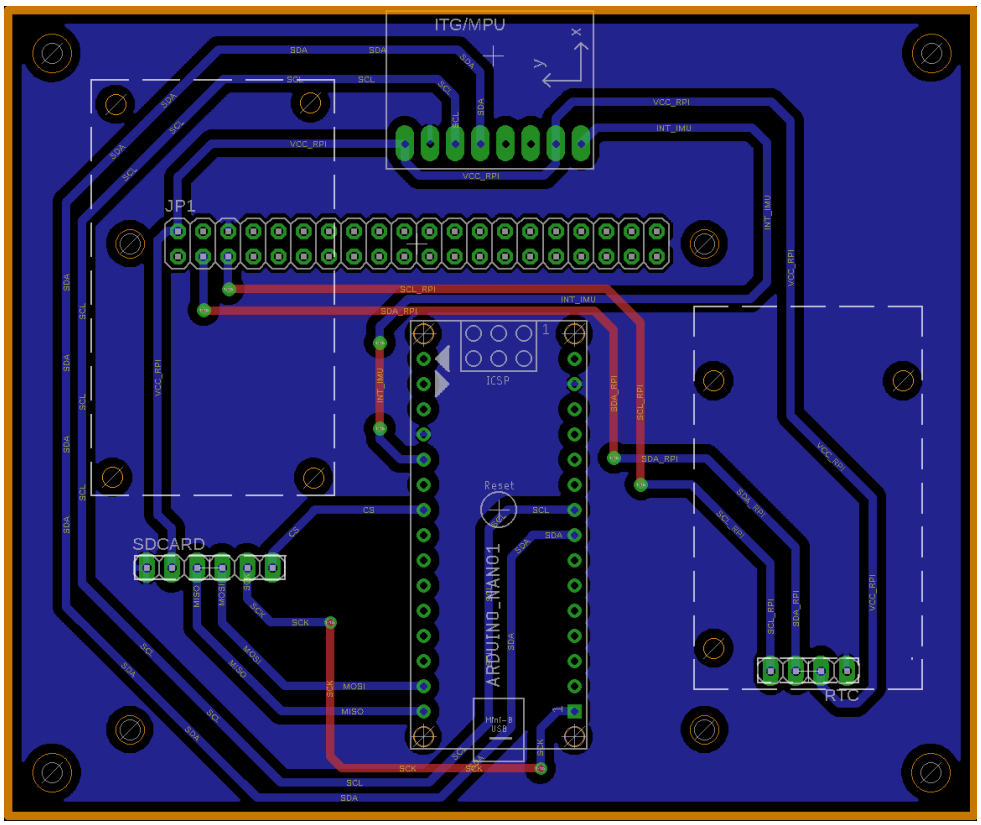


Figura 3.13: Layout del circuito

3.4. DISEÑO DEL SOFTWARE

Esta sección abordará, en detalle, aspectos previamente mencionados del software desarrollado en el presente trabajo.

3.4.1. Manejo de Periféricos

- Lectura del Módulo de Medición

  Como ya se explicó en la sección anterior, el sensor utilizado será el módulo MPU6050, cuyo estándar de transmisión utilizado es el I2C. Por medio de este es posible configurar sus registros internos para definir su modo de operación, así como, leer la data proveniente de sus mediciones.

  Los registros donde se almacenan las mediciones realizadas por los acelerómetros y los giroscopios en cada uno de sus ejes son de 8bits. Debido a que las lecturas de los sensores internos tienen una resolución de 16 bits, entonces se cuentan 2 registros (8 bits cada uno), direccionados de forma contigua, para formar la lectura completa de cada sensor en cada uno de sus ejes. La idea de la formación de los datos de 16 bits se muestra a continuación:

$$Lectura(16bits) = (Registro_{HIGH\ BYTE} \ll 8)|\ (REGITRO_{LOW\ BYTE})$$

  Tomando en cuenta esta condición, es posible conocer las lecturas del acelerómetro (Aceleración) y del giroscopio (Gyro: velocidad angular), en cada uno de sus ejes X, Y, Z, tal como es mostrado en la Figura 3.14.

$$\begin{aligned}
Aceleración\ X &= (Registro_{HIGH\ BYTE} ACCX \ll 8)|(Registro_{LOW\ BYTE} ACCX)\\
Aceleración\ Y &= (Registro_{HIGH\ BYTE} ACCY \ll 8)|(Registro_{LOW\ BYTE} ACCY)\\
Aceleración\ Z &= (Registro_{HIGH\ BYTE} ACCZ \ll 8)|(Registro_{LOW\ BYTE} ACCZ)\\
Gyro\ X &= (Registro_{HIGH\ BYTE} GYROX \ll 8)|(Registro_{LOW\ BYTE} GYROX)\\
Gyro\ Y &= (Registro_{HIGH\ BYTE} GYROY \ll 8)|(Registro_{LOW\ BYTE} GYROY)\\
Gyro\ Z &= (Registro_{HIGH\ BYTE} GYROZ \ll 8)|(Registro_{LOW\ BYTE} GYROZ)
\end{aligned}$$

Figura 3.14: Lectura de los registros del sensor

Las lecturas provenientes de los acelerómetros y giroscopios serán procesadas mediante un algoritmo matemático de Fusión de Datos, el cual va a permitir estimar el ángulo correspondiente a la información ingresada eliminando el ruido y desviaciones propias de las características de los sensores usados, tales como ruido de alta frecuencia, offset y drift (error de acumulación en períodos largos, lo cual

genera cambio del offset en un instante respecto del inicial). Como se mencionó en el Capítulo 2, las técnicas existentes para llevar a cabo dicho procesamiento son los Filtro Complementario y Filtro Kalman. Asimismo, cabe resaltar que en la presente tesis no se diseñará un algoritmo de procesamiento, en su lugar, se utilizarán librerías open source, las cuales implementan el algoritmo elegido tal que la data pueda procesarse en el microcontrolador empleado. Sin embargo, el uso de las librerías puede no tener el efecto esperado sobre la data con los parámetros pre-establecidos, lo cual implica realizar un ajuste manual de dichos parámetros según la respuesta esperada. Asimismo, la configuración de los parámetros utilizados en cada una de las librerías de fusión de datos utilizadas se realizó de forma experimental, es decir, se llevaron a cabo constantes pruebas con la finalidad de obtener la mejor respuesta para cada una de ellas. Dicha configuración puede ser observada en la sección C de Anexos (Códigos de Programa Utilizados). A partir de esta se realiza el análisis cuyos resultados son mostrados en la sección 4.4.

- Módulo Micro SD para el almacenamiento de Datos

  Como se mencionó en la sección 3.3.3, se utilizará el Módulo MicroSD para almacenar toda la data procesada por la unidad de adquisición, la cual servirá para realizar una posterior evaluación del desempeño de la persona evaluada. El módulo seleccionado utiliza el formato de comunicación SPI, por lo cual utiliza 4 pines del microcontrolador, para realizar la transferencia de información. El microcontrolador utilizado, Arduino, cuenta con librerías que permiten la implementación de dicho estándar de comunicación, así como la configuración de forma directa de la unidad de almacenamiento utilizada. El primero de ellos se denomina SPI.h, el cual proporciona funciones que permiten realizar una transferencia directa de información utilizando dicho estándar de comunicación. El segundo, conocido como SD.h, brinda funciones que permiten tanto la configuración de la unidad MicroSD utilizada, así como la creación de archivos (.txt, .xlsx, etc.) o directorios, en los cuales es posible realizar el almacenamiento de información.

  La creación de un archivo en el cual se pueda escribir información importante requiere tener un nombre que permita identificarlo, y sobre el cual se dirigirá las opciones de almacenamiento en el programa implementado. Sin embargo, este

nombre presenta ciertas restricciones que deben cumplirse para poder inicializar el archivo de grabación de forma correcta y sin errores. Dichas restricciones son las siguientes:

- ✓ Nombre de archivo o directorio debe ser una palabra tipo 'String'.
- ✓ Nombre de archivo o directorio debe contener 8 caracteres como máximo sin considerar la extensión de dicho archivo.
- ✓ Para la creación de un directorio es necesario incluir el carácter '/' al final del nombre, este no cuenta como un carácter dentro de los 8 admitidos para el nombre.

El archivo creado será del tipo 'File', una clase de objeto existente en la librería SD.h, que permite direccionar las funciones para la escritura de información dentro del archivo correspondiente.

En este trabajo, la información almacenada dentro del archivo de grabación generado en cada caso será el ángulo roll, correspondiente a la rotación del tronco corporal alrededor del eje x, que ya se encuentra procesado por el filtro correspondiente. Adicionalmente, se almacenará la referencia de posición seguida por la persona evaluada. Con dicha información es posible llevar a cabo el análisis posterior para comparar la posición del tronco de la persona evaluada respecto al patrón ideal a seguir.

3.4.2. Protocolo de Comunicación

Brindar la realimentación visual de posición a la persona evaluada requiere distribuir adecuadamente los datos entre los módulos implicados para la presente labor, el módulo de adquisición y el módulo de interfaz gráfica. Dicho esto, y con la finalidad de garantizar la fiabilidad de la transmisión de datos, resulta necesario la existencia de un Protocolo de Comunicación, el cual regirá el flujo de información entre ambos módulos para lograr una óptima respuesta del sistema. El protocolo de comunicación diseñado se muestra en la Figura 3.15.

Como se puede observar en la Figura 3.15, el protocolo de comunicación se ha estructurado en base a la prueba que se va a desarrollar y como esta será presentada en la interfaz gráfica que se desplegará durante la prueba. Asimismo, la Figura 3.16 presenta el diagrama de Flujo que permite entender mejor el funcionamiento del protocolo de comunicación mostrado en la Figura 3.15.

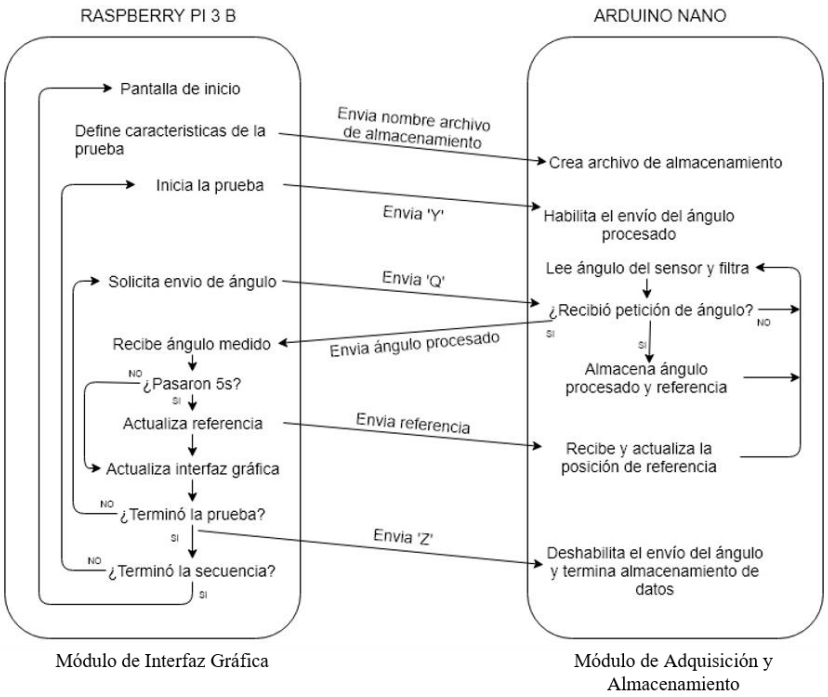


Figura 3.15: Protocolo de comunicación diseñado

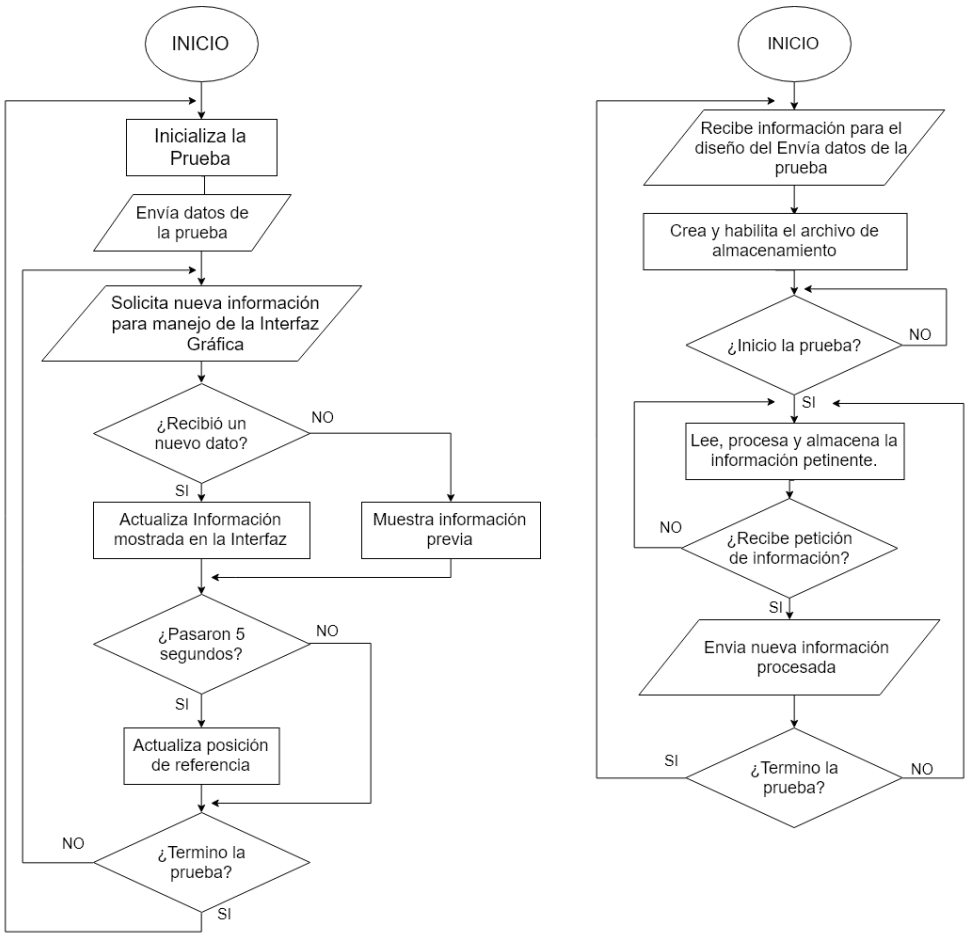


Figura 3.16: Diagrama de Flujo del Protocolo de Comunicación

Al inicio de la prueba, la interfaz gráfica desplegará una pantalla de inicio, en la cual se ofrecerán ciertos criterios de selección para que el evaluador pueda decidir entre dos modos de aplicación: 'Demo' y 'Trial'.

- Demo: Esta opción permite entrenar una serie aleatoria de posiciones tal que la persona evaluada tenga un primer contacto con el funcionamiento del dispositivo.
- Trial: Esta opción permite elegir la secuencia que desea realizar (A, B, C o D) durante la evaluación, donde cada secuencia contiene una serie de pruebas, cada una de las cuales alberga una configuración distinta de posiciones presentadas de forma aleatoria para la persona evaluada. Este modo permite la evaluación de la persona evaluada.

Asimismo, en cada modo de operación se podrá asignar un índice a la prueba que está realizando, caracterizando a la persona evaluada. Una vez terminado dicho proceso, el módulo de interfaz gráfica adquiere dicha información y genera un nombre para el archivo en el cual se almacenará toda la información adquirida durante la prueba respectiva. Dicha etiquetación del archivo depende del formato de prueba elegido.

- Demo:

  $$DIIMMDD.txt$$

  - D: Demo
  - II: Índice de la prueba.
  - MMDD: Fecha en que se lleva a cabo (mes y día)
- Trial:

  En este modo, adicionalmente se creará una carpeta en la que se almacenarán los archivos correspondientes a la secuencia elegida por el usuario.

  - Formato Carpeta: $TIISSMMDD/$
    - T: Trial
    - II: Índice de la prueba
    - SS: Secuencia de la prueba elegida
    - MMDD: Fecha en que se lleva a cabo (mes y día)

  - Formato Archivo: $TNNSSMMDD.txt$
    - T: Trial
    - NN: Número de prueba de la secuencia respectiva

- SS: Secuencia de la prueba elegida
- MMDD: Fecha en que se lleva a cabo (mes y día)

Cuando la unidad de adquisición haya recibido dicha información creará el archivo correspondiente en la unidad de almacenamiento sobre la cual se grabarán los datos del sensor una vez iniciada la prueba.

La prueba se inicia al presionar el botón 'START' de la interfaz gráfica, con lo cual se habilitará la operación de la unidad de adquisición mediante el envío de una bandera. A partir de ese momento dicha unidad empezará a muestrear información proveniente del sensor a la frecuencia indicada en la Tabla 3.3, la procesará haciendo uso de una librería de Fusión de Datos, explicado en la sección 2.1.2, y finalmente almacenará dicho dato en el archivo de grabación previamente creado.

A partir de este punto, el protocolo diseñado juega un rol importante para lograr la sincronización del envío y recepción de datos sin evidenciar pérdida de información. Esto es necesario ya que existe un requerimiento importante que rige el comportamiento de la prueba. Este requerimiento se basa en que la respuesta del sistema debe ser lo más rápida posible con la finalidad que la persona evaluada pueda visualizar en la pantalla el movimiento que está llevando a cabo en dicho instante de tiempo, y no se encuentre afectado por un retardo.

El protocolo de comunicación establecido posiciona a la Raspberry Pi como el módulo maestro, y al módulo Arduino Nano como el esclavo del sistema. En base a esto, al inicio de la prueba, el módulo maestro solicita un dato a la unidad de adquisición (esclavo), y este último envía el dato más reciente que se adquirió del módulo de medición y fue procesado mediante el algoritmo de Fusión de Datos empleado. El maestro recibirá dicho dato, y convertirá el ángulo recibido (en ° sexagesimales) a pixeles a visualizarse en la pantalla, tal como se muestra en (*). Esta acción se repetirá de manera continua durante el tiempo que dure la prueba.

$$Pos_{pixels} = d * \sin\left(Pos_{grados} * \frac{PI}{180°}\right) * \left(\frac{Ancho_{pantalla(pixels)}}{Ancho_{pantalla(cm)}}\right) \qquad (*)$$

Considerar *d* como la distancia vertical medida desde la espalda baja de la persona hasta el centro del monitor. Recordar que, durante la prueba, la persona evaluada estará sentada frente al monitor.

A partir de lo mencionado, la información que se podrá visualizar en la pantalla será la posición objetivo que el usuario debe alcanzar de la forma más precisa y rápida posible, así como la posición del usuario en dicho instante de tiempo. Asimismo, la posición del objetivo a seguir debe actualizarse cada 5s, por lo que el usuario solo dispone de este tiempo para alcanzar dicha posición. Una vez transcurrido dicho tiempo, la posición de referencia del objetivo cambia y dicha información de referencia es notificada al módulo de adquisición. Al recibir este dato, el módulo de adquisición actualiza dicho parámetro que también es almacenado en el archivo de almacenamiento previamente creado.

Con la data recibida por parte del módulo de adquisición, y luego de realizar el procesamiento respectivo, el módulo maestro actualiza la información visualizada en la pantalla. Esto se realiza con la finalidad que la persona evaluada reciba una realimentación adecuada de su movimiento tal que logre controlar su desempeño durante el desarrollo de la prueba.

La duración de la prueba se extenderá dependiendo de la trama de posiciones que conforme la secuencia elegida. Una vez terminada la prueba, la interfaz gráfica mostrará nuevamente la pantalla de inicio, siguiendo, de esta forma, la secuencia descrita líneas arriba. Asimismo, al culminar la prueba, el maestro enviará una bandera hacia el módulo de adquisición indicando el término de la misma, con lo cual se deshabilitará la lectura del sensor y, a la vez, se cerrará el archivo de grabación con la data que logró almacenarse hasta dicho momento. Luego de esto, dicho módulo quedará preparado para dar inicio a una nueva secuencia de evaluación.

Los diagramas de flujo sobre los cuales se rige el comportamiento del sistema completo tanto para la Raspberry Pi (módulo de interfaz gráfica) como para el Arduino Nano (unidad de adquisición) son mostrados en la sección B de Anexos (Diagramas de Flujo).

CAPITULO 4: PRUEBAS Y RESULTADOS

4.1. INTRODUCCIÓN

En el presente capítulo se mostrarán los resultados obtenidos durante el desarrollo de la presente tesis. Dichos resultados estarán segmentados en la implementación de la misma, así como en el proceso de optimización llevado a cabo para obtener un sistema con el menor retardo posible para garantizar su confiabilidad.

4.2. IMPLEMENTACIÓN DEL MÓDULO DE ADQUISICIÓN PARA LA EVALUACIÓN DEL BALANCE CORPORAL

El sistema se desarrolló considerando dos partes claramente definidas, como son el hardware necesario y el software que comandará el funcionamiento eficaz del sistema.

4.2.1. Nivel Hardware

En esta sección se presentan los resultados obtenidos de la implementación del módulo de adquisición para la evaluación del balance corporal en base a la sección 3.3 del capítulo anterior.

Las Figura 4.1 a Figura 4.3 muestran el hardware implementado tomando como referencia las Figura 3.11 a Figura 3.13. Asimismo, todo lo mencionado en el capítulo anterior, relativo al diseño del sistema, se ha distribuido de tal forma que se pueda obtener un módulo que sea, portable y de bajo costo. Tal es así que, durante la realización de la prueba, este sistema estará colocado sobre una faja dorso lumbar, la cual será utilizada por la persona evaluada permitiendo el análisis del movimiento de la espalda baja. Por ello, se debe considerar la distribución adecuada de los componentes empleados con la finalidad de obtener un dispositivo liviano tal que no afecte la performance de la persona evaluada.

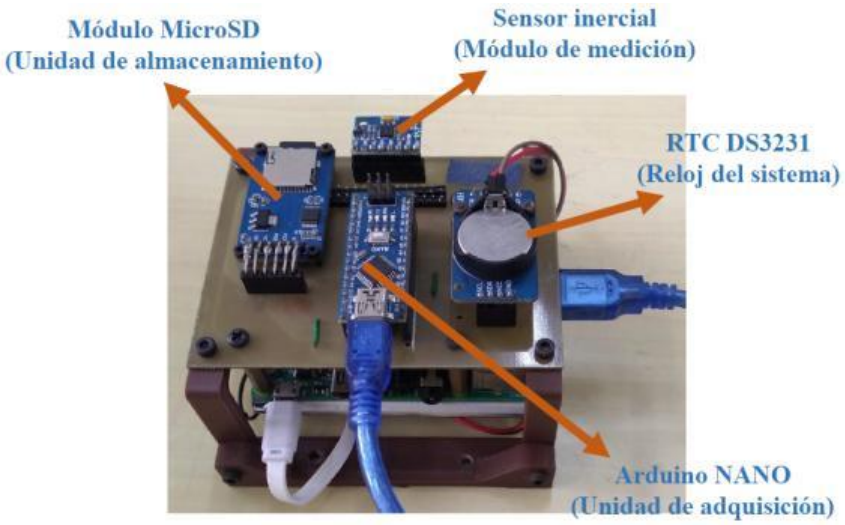


Figura 4.1: Módulo de medición y Módulo de Adquisición y Almacenamiento

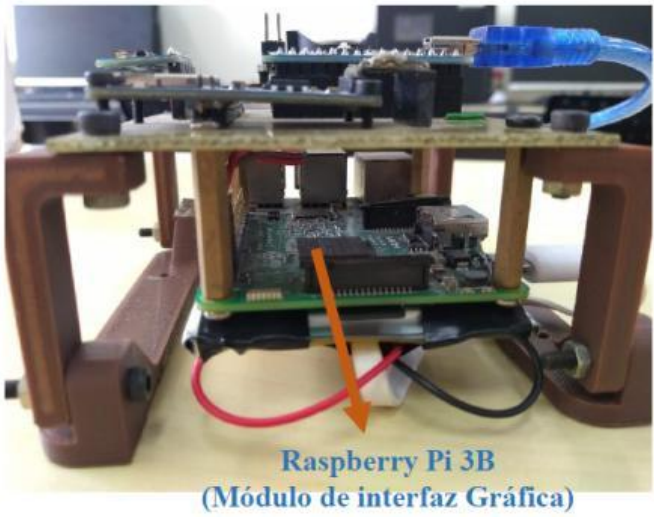


Figura 4.2: Módulo de Interfaz Gráfica

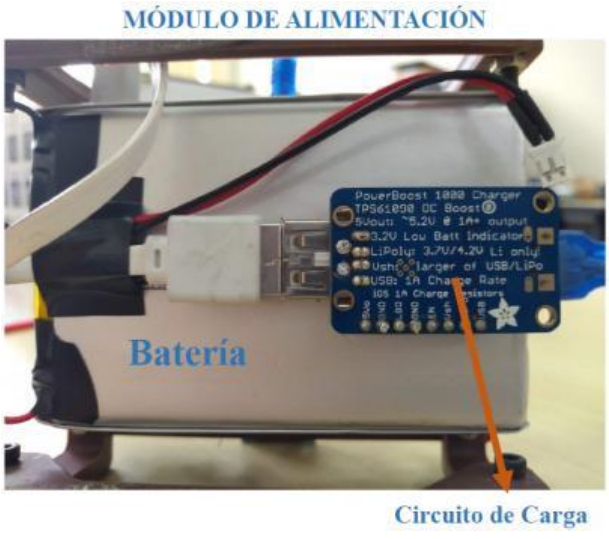


Figura 4.3: Módulo de Alimentación

4.2.2. Nivel Software – Interfaz Gráfica

Esta sección del sistema sirve para llevar a cabo la prueba, desplegando la posición objetivo a seguir por la persona evaluada y, a la vez, será posible visualizar el comportamiento del individuo durante la evaluación para corregir su desempeño en caso lo considere necesario.

Esta interfaz se desarrolló con la finalidad de ser un entorno amigable para el evaluador, así como la persona evaluada. En la Figura 4.4 se muestra la pantalla de inicio de la interfaz, en la cual se podrá elegir entre los dos modos de operación (Demo y Trial) con los que cuenta el sistema para llevar a cabo la prueba, los cuales fueron explicados en el capítulo previo.

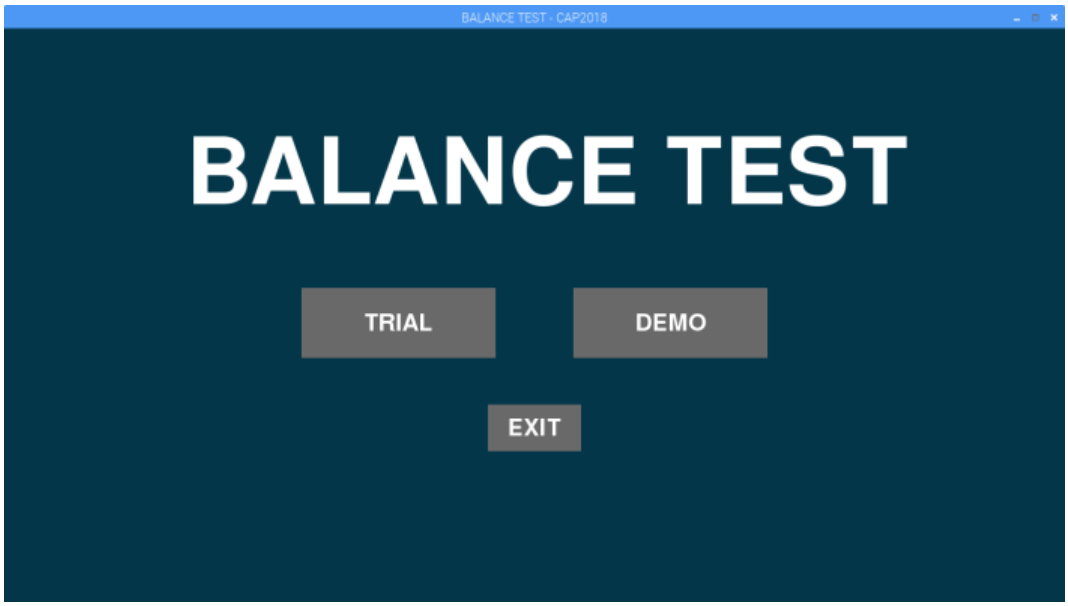
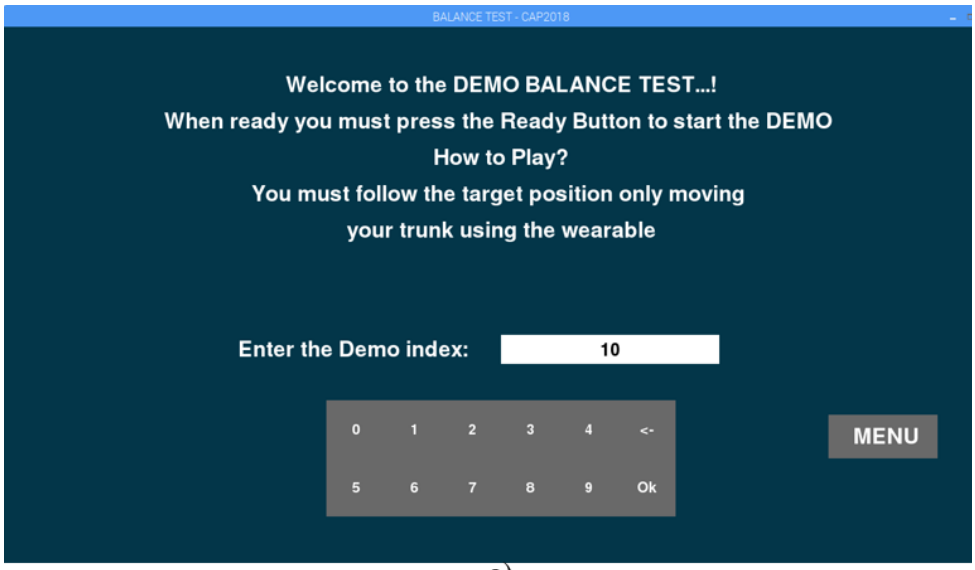


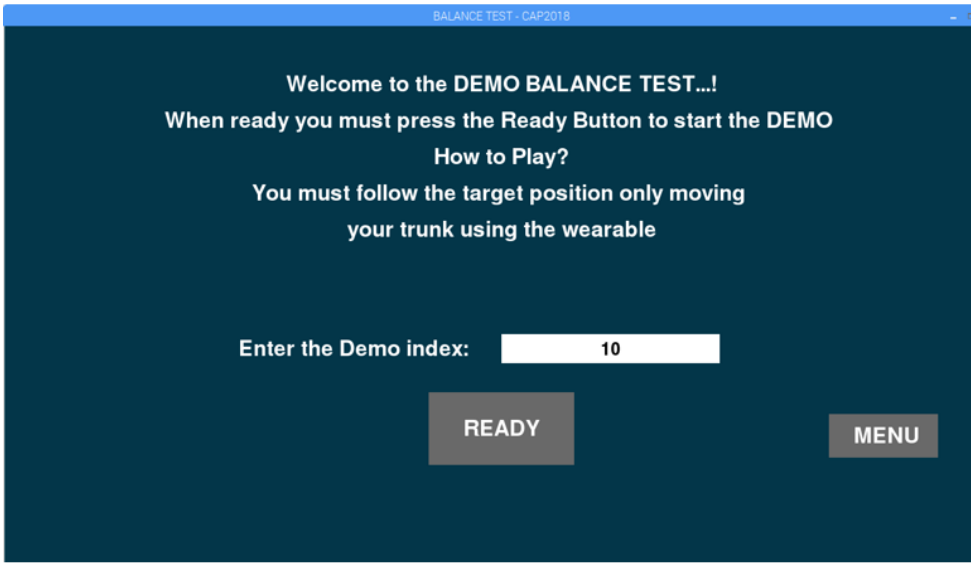
Figura 4.4: Pantalla de inicio.

Por un lado, la Figura 4.5 muestra la forma de operación de la interfaz al elegir el modo de operación “Demo”.

Por otro lado, el sistema también cuenta con otro modo de operación, denominado ‘Trial’, el cual permitirá recolectar la data necesaria proveniente de la persona evaluada. En la Figura 4.6 se muestra las ventanas de inicialización para este caso.

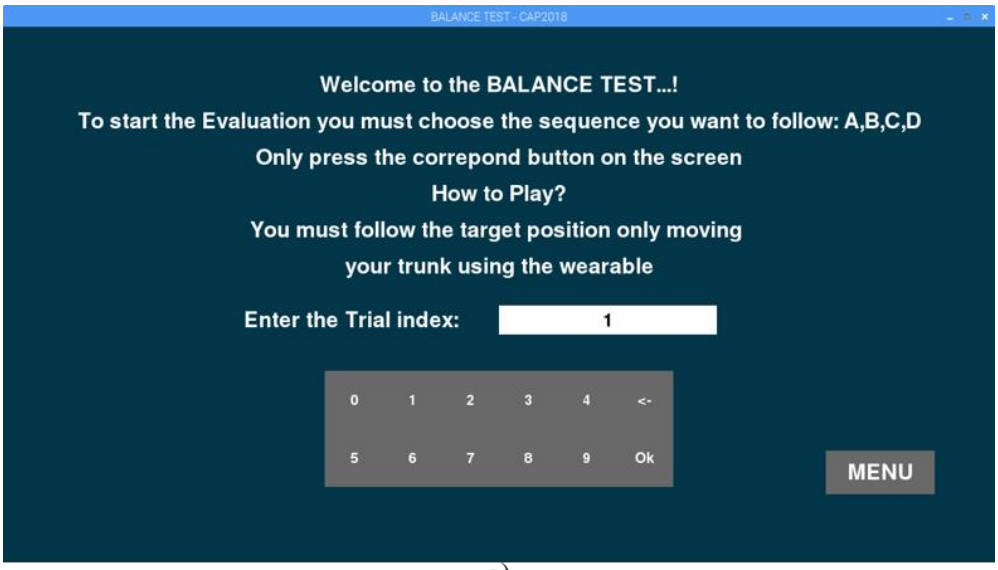


a)

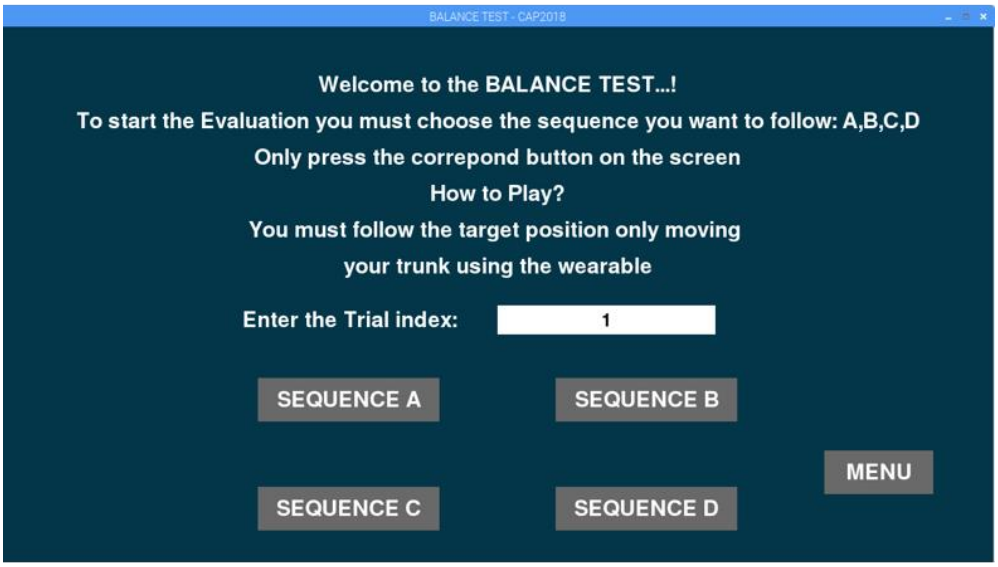


b)

Figura 4.5: Interfaz en modo ‘Demo’



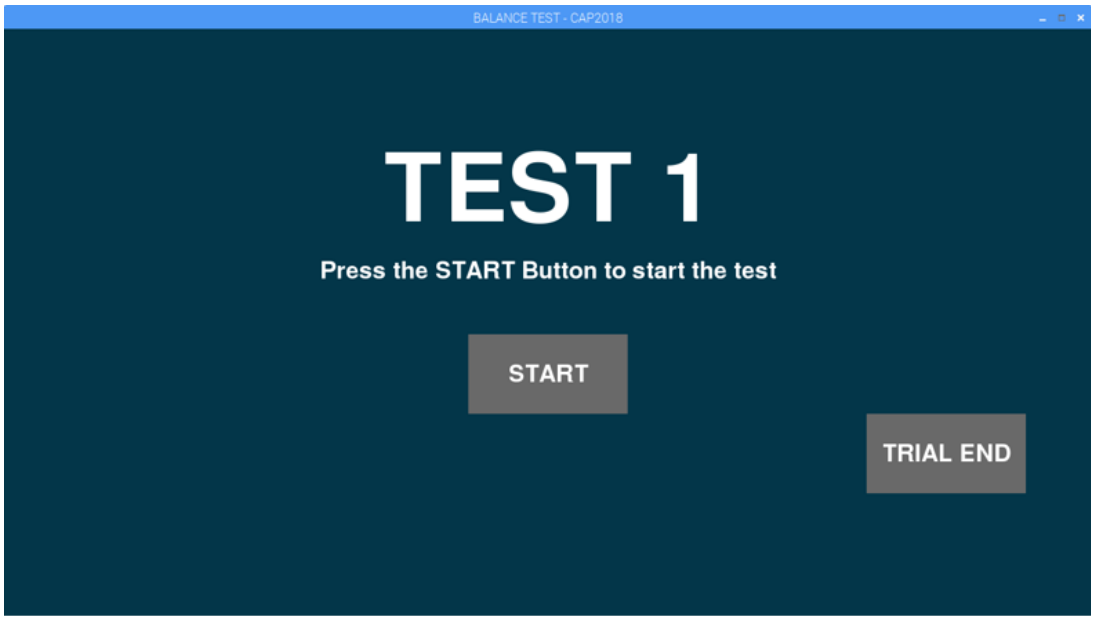
a)



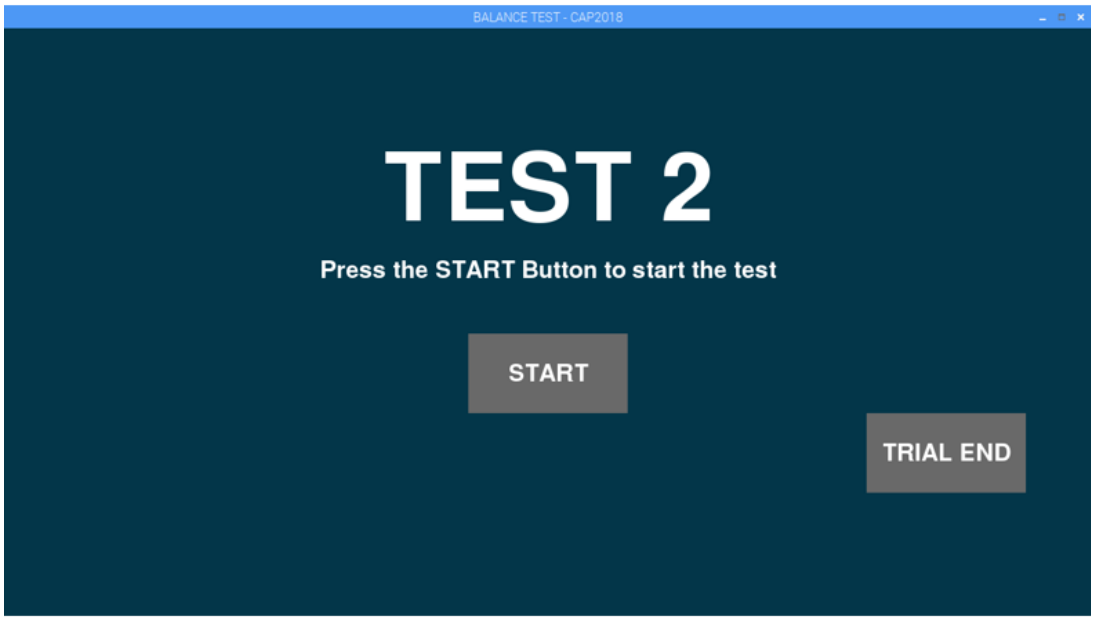
b)

Figura 4.6: Interfaz en modo 'Trial'

Luego de acceder al modo 'Trial, la interfaz permitirá comenzar con el desarrollo de las distintas pruebas que conforman a cada secuencia elegida, las cuales se llevarán a cabo uno a continuación de otro, tal como se puede visualizar en la Figura 4.7.



a)

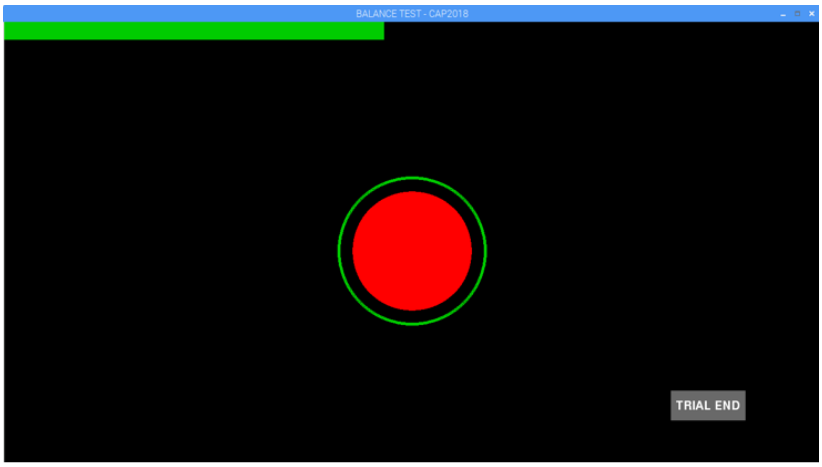


b)

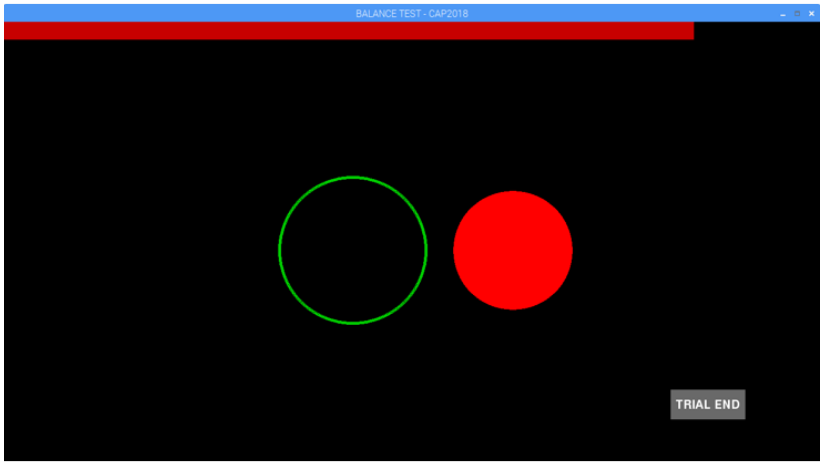
Figura 4.7: Ventanas de los test que conforman la secuencia elegida

Como se puede notar, la interfaz gráfica se ha desarrollado en el idioma inglés debido a que es el investigador Peter Reeves, Sumaq Life LLC, USA, quien, en conjunto con el laboratorio LIBRA, propuso la idea del presente proyecto y busca replicar el presente sistema para llevar a cabo las pruebas de control del equilibrio corporal.

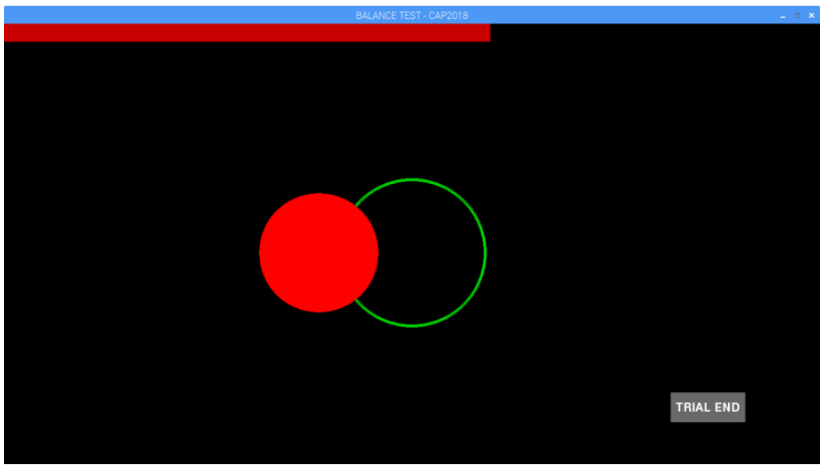
Una vez que se haya llevado a cabo la inicialización de la prueba en el formato elegido, se comenzará con el desarrollo de la misma, cuyos resultados se visualizan en la Figura 4.8.



a)



b)



c)

Figura 4.8: a) Posición 0°, columna recta. B) Posición > 0°, columna inclinada hacia la derecha. c) Posición < 0°, columna inclinada hacia la izquierda

La Figura 4.9 muestra la implementación realizada durante el desarrollo de una prueba de evaluación del balance corporal.

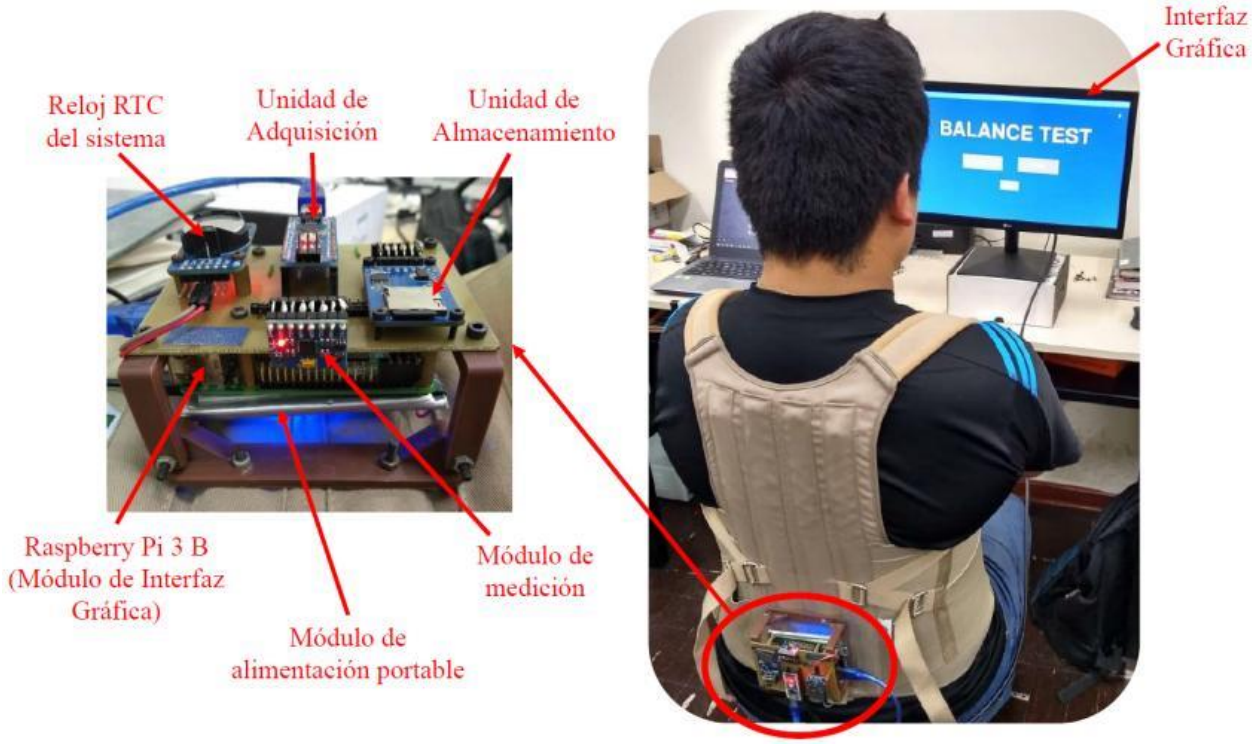


Figura 4.9: Vista panorámica de la implementación del sistema durante la prueba

4.3. ANÁLISIS DE TIEMPOS POR ETAPAS DEL PROCESO

Este análisis de tiempos se ha desarrollado ya que el retardo del sistema es un factor importante en su funcionamiento, tal que no afecte el desempeño de la persona evaluada durante la realización de la prueba. Dicho retardo, está influenciado por el tiempo de ejecución de cada sección desarrollada. Además de ello, cada técnica de Fusión de Datos, mencionadas en el Capítulo 2, también añade cierto nivel de retardo en la actualización de su valor de salida, lo cual será evaluado en la siguiente sección.

4.3.1. Etapa de Lectura, Procesamiento y Almacenamiento de Información

Considerando que el funcionamiento del sistema comienza con la lectura del sensor realizada por el microcontrolador, es posible conocer el tiempo que se necesita para adquirir dicha información. Utilizando el formato de comunicación I2C admitido por el sensor en el modo Fast Mode (400kHz), se obtienen los resultados mostrados en la Figura 4.10.

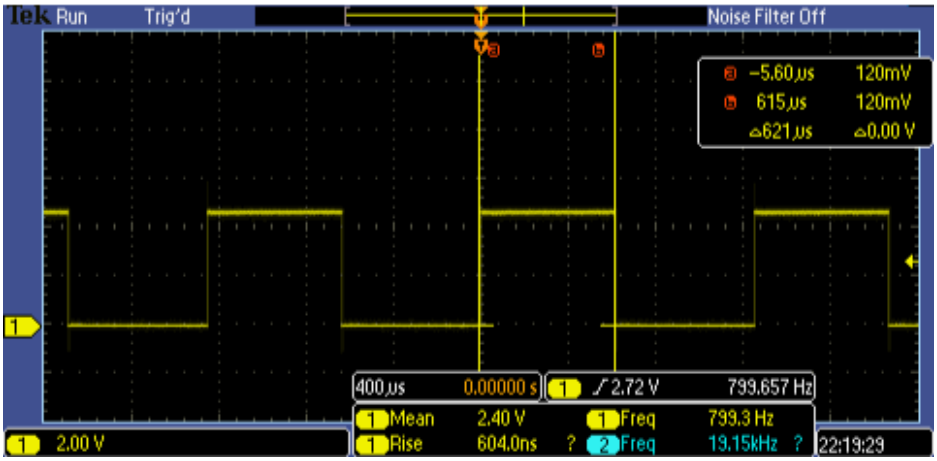


Figura 4.10: Tiempo de lectura del sensor inercial MPU6050: 621us

La data recibida por la unidad de adquisición será procesada para obtener la posición angular del tronco de la persona evaluada en dicho instante de tiempo. Como se menciona en la sección 3.3.3, correspondiente a la lectura del sensor y su procesamiento, se aplicaron dos tipos de filtros, como son, el Filtro Complementario y el Filtro Kalman. Cada uno de estos posee cierta cantidad de instrucciones, los cuales serán ejecutados a la velocidad proporcionada por el microcontrolador (16MHz), requiriendo cierto tiempo de procesamiento durante su ejecución. Dichos tiempo son mostrados en las Figura 4.11 y Figura 4.12, respectivamente. Asimismo, la Tabla 4.1 muestra de forma explícita el tiempo que demora en ejecutarse de forma completa cada filtro utilizado. Como se puede notar, el Filtro Complementario, al tener menos líneas de instrucción, requiere menos tiempo de ejecución. Por otro lado, se observa que el Filtro Kalman presenta un tiempo de ejecución cercano al doble del presentado por el Filtro Complementario. El algoritmo correspondiente a cada técnica de Fusión de Datos es mostrado en la sección 2.1.2.

Tabla 4.1: Comparación de tiempos de ejecución de cada filtro

Filtro utilizado	Tiempo de ejecución
Filtro Complementario	712 us
Filtro Kalman	1300us

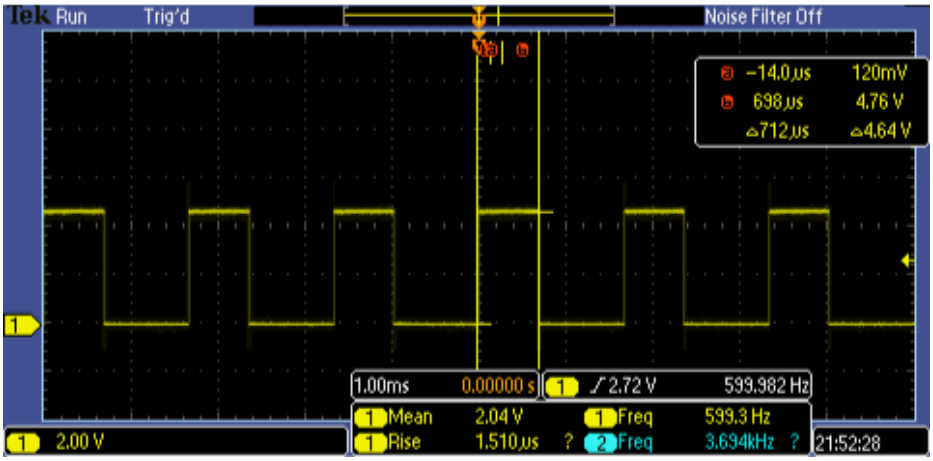


Figura 4.11: Tiempo de ejecución del Filtro Complementario utilizado: 712us

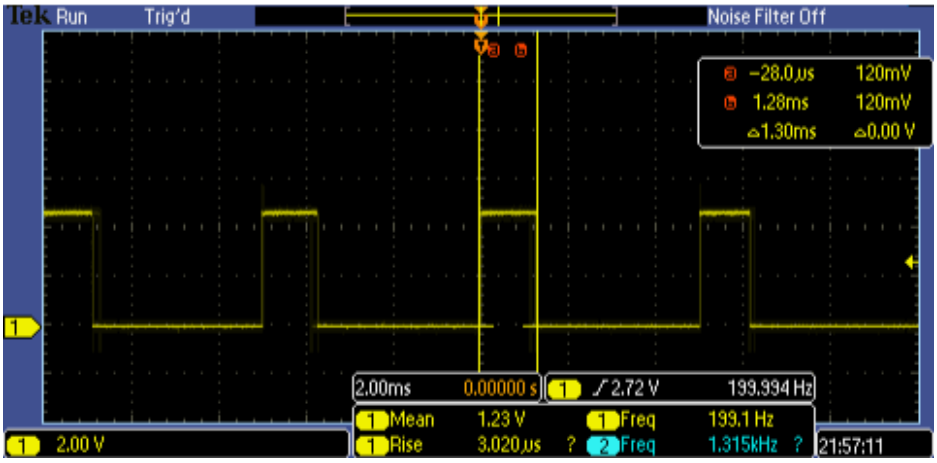
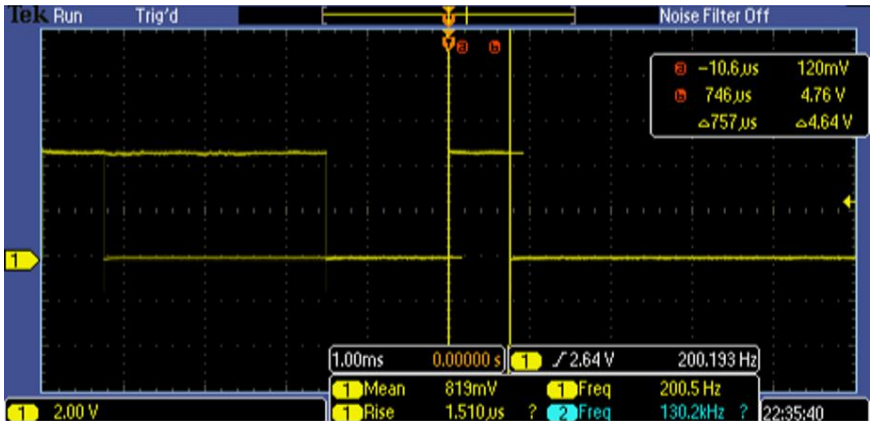


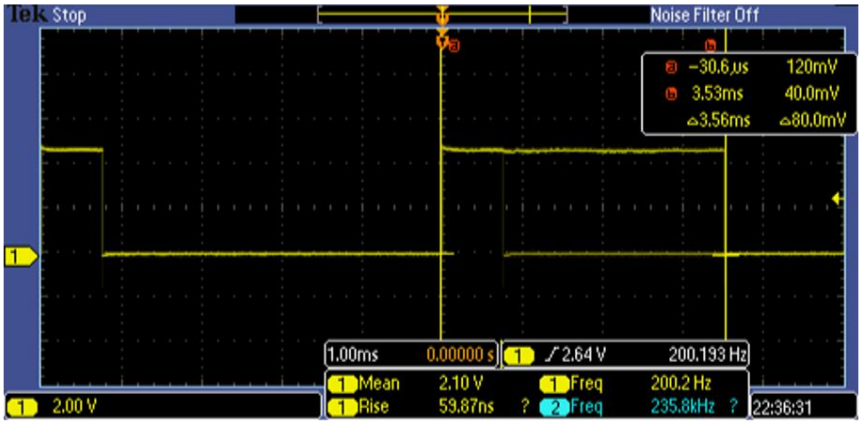
Figura 4.12: Tiempo de ejecución del Filtro Kalman utilizado: 1300us

La data procesada será almacenada en la unidad respectiva para que pueda ser utilizada en análisis posteriores. El almacenamiento de dicha información se realiza utilizando el protocolo SPI, como se muestra en la Figura 3.7. A partir de esto, se puede identificar el tiempo aproximado que demanda dicho almacenamiento en la Unidad SD, el cual es mostrado en la Figura 4.13.

Asimismo, la data filtrada será transmitida hacia el módulo de la interfaz gráfica, cuando este lo requiera, para que pueda mostrar dicha información en pantalla y sirva de realimentación para la persona evaluada. Como ya se mencionó en el capítulo 3, dicho protocolo se basará en la comunicación serial la cual presenta el comportamiento mostrado en las Figura 4.14 y Figura 4.15.



a)



b)

Figura 4.13: Tiempo de ejecución grabación SD. a) Mínimo: 757us. b) Máximo: 3.56ms

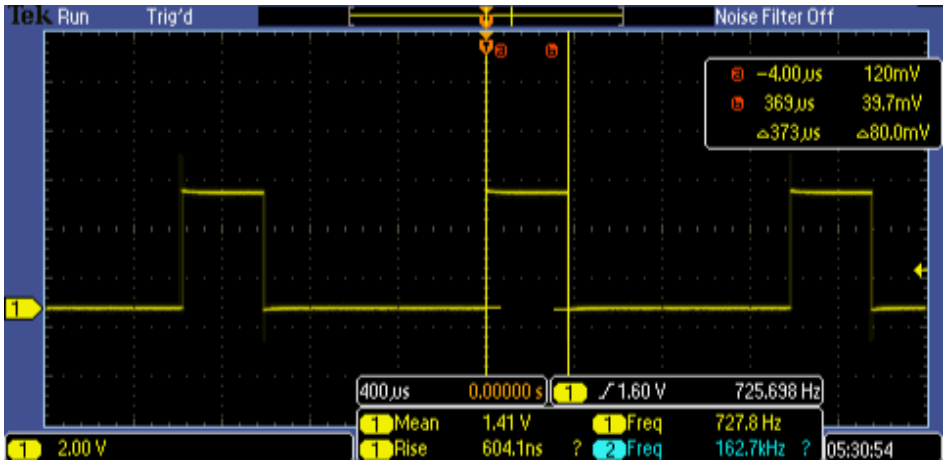


Figura 4.14: Tiempo de ejecución escritura de un dato tipo Float con 4 decimales por serial a 115200 baudios: 373us

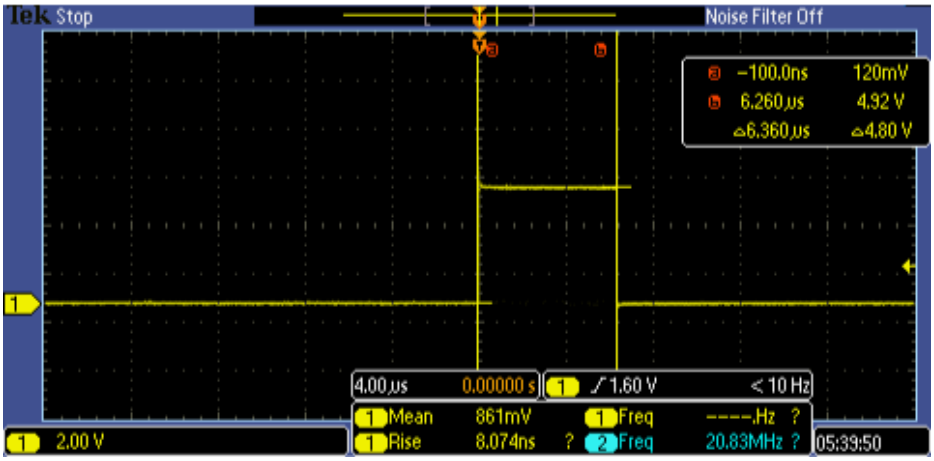


Figura 4.15: Tiempo de ejecución lectura por serial de un carácter a 115200 baudios: 6.36us

4.3.2. Etapa de visualización en Pantalla

La interfaz gráfica desplegada para la realización de la prueba será desarrollada en el lenguaje Python. Para el propósito se utilizará la librería "pygame" [22] con la cual es posible generar una pantalla con la integración de imágenes y animaciones que permiten al usuario tener una experiencia amena durante el desarrollo de la prueba. De esta librería se usaron distintas funciones, como se puede observar en la sección C de Anexos. Cada una de dichas funciones fue evaluada para conocer su desempeño. A continuación, se mostrarán los resultados obtenidos para cada función utilizada.

La pantalla que permitirá al usuario conocer su posición en un instante dado será generada utilizando funciones que permiten añadir imágenes o incluir formas geométricas en la pantalla. Las funciones principales en esta etapa son:

- pygame.draw.rect(parámetros)
- pygame.draw.ellipse(parámetros)

La Figura 4.16 muestra que dichas funciones requieren un tiempo de 732us para generar las pantallas que serán desplegadas en un instante requerido.

Luego de generar la pantalla que será visualizada, ésta debe imprimirse en el monitor que se usará para llevar a cabo la prueba. En otras palabras, la pantalla debe actualizarse con la nueva información generada para que la persona evaluada conozca su desempeño en un instante dado. Para realizar dicha acción se utilizará la función:

- pygame.display.update(parámetros)

La actualización de la pantalla demanda un tiempo de 2.58ms, tal como se muestra en la Figura 4.17.

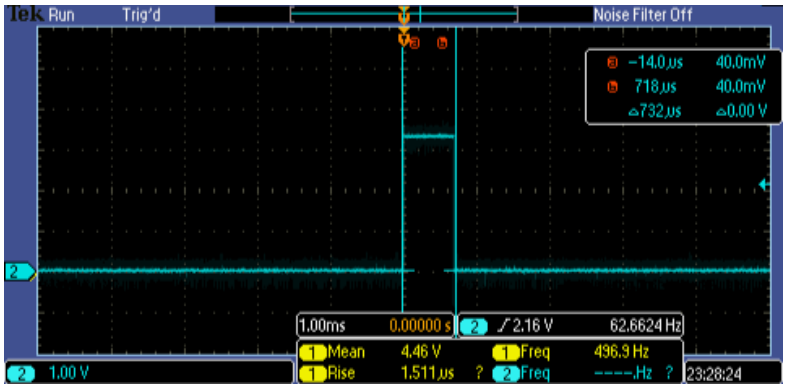


Figura 4.16: Tiempo de ejecución para la generación de la pantalla: 732us

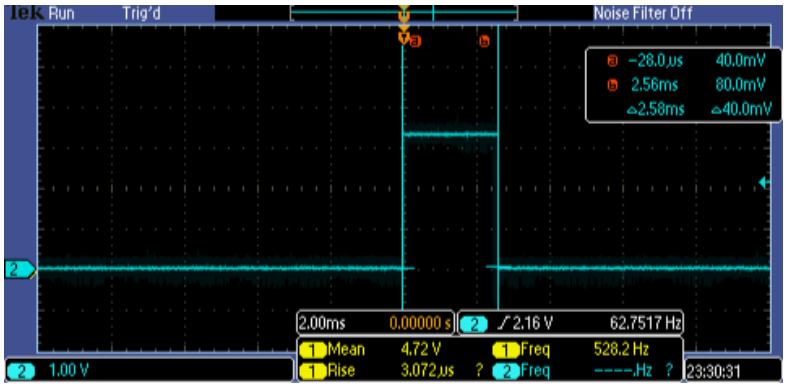


Figura 4.17: Tiempo de ejecución para la actualización de la pantalla: 2.58ms

Luego de actualizar la pantalla y haber sido visualizada por la persona evaluada, dicha pantalla es 'borrada' para imprimir una nueva con la información actualizada. Este proceso es similar al de la generación de pantalla, y requiere un tiempo de 685us tal como se aprecia en la Figura 4.18.

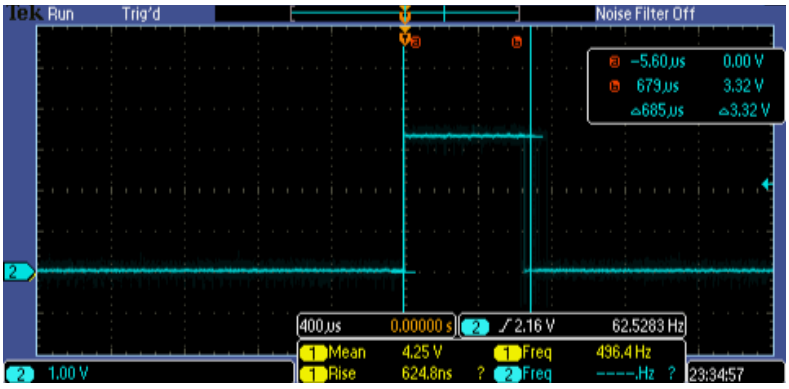


Figura 4.18: Tiempo de ejecución de limpieza de la pantalla: 685us

Con la finalidad de actualizar la interfaz con la posición actual del tronco de la persona, se implementó un protocolo de comunicación entre el módulo de adquisición y el

módulo de interfaz gráfica. En base a dicho protocolo se puede conocer el tiempo que demora la ejecución de la lectura y escritura de un dato en la Raspberry Pi. Dicha información es mostrada en las Figura 4.19 y Figura 4.20, respectivamente.

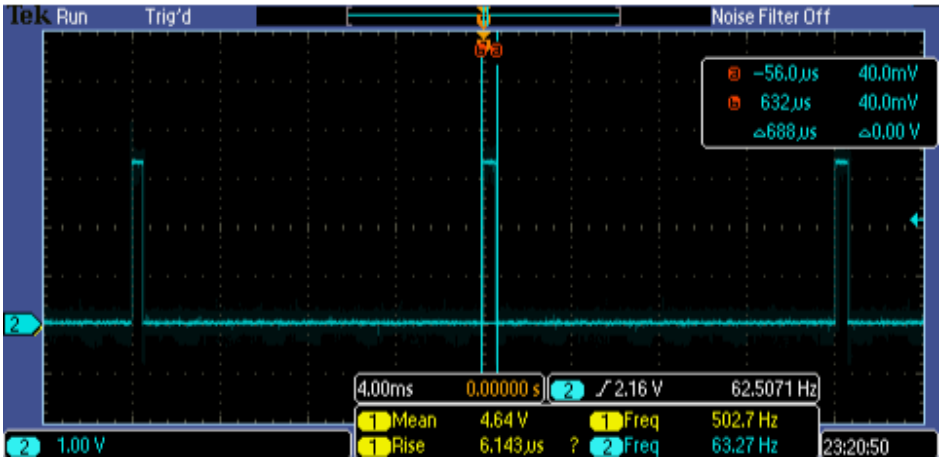


Figura 4.19: Tiempo de ejecución de lectura de un dato tipo Float con 4 decimales por serial a 115200 baudios: 688us

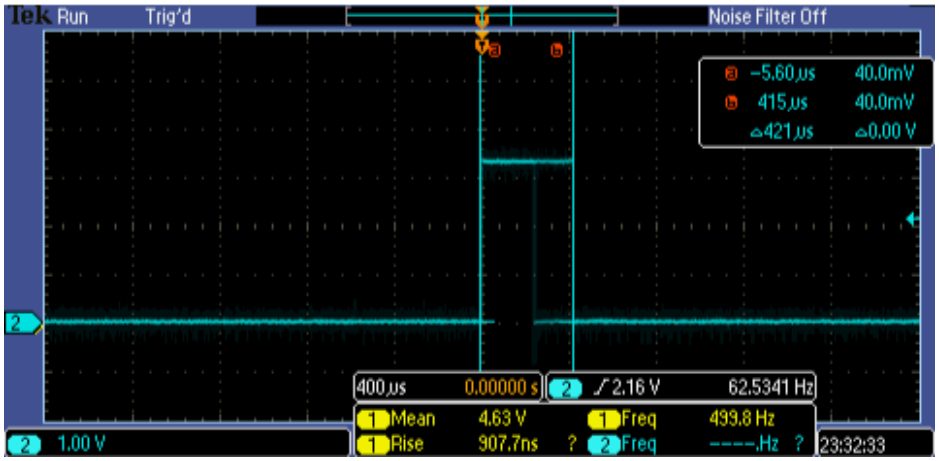


Figura 4.20: Tiempo de ejecución de escritura por serial de un caracter a 115200 baudios: 421us

Las Tabla 4.2 y Tabla 4.3 muestran un resumen de los tiempos de ejecución descritos en la presente sección.

Tabla 4.2: Resumen de tiempos: Etapa de lectura, procesamiento y almacenamiento de información

Tiempo (us)	621	712us o 1300us	2158.5
Actividad	Leer IMU	Procesamiento de la señal (Filtro Complementario o Filtro Kalman)	Almacenamiento SD promedio
ETAPA DE LECTURA, PROCESAMIENTO Y ALMACENAMIENTO DE INFORMACIÓN			

Tabla 4.3: Resumen de tiempos: Etapa de visualización en pantalla

Tiempo (us)	688	732	2580	685	421
Actividad	Lectura Serial	Generar Pantalla	Actualización de Pantalla	Limpieza de Pantalla	Envio del Flag hacia el Arduino
ETAPA DE VISUALIZACIÓN EN PANTALLA					

La Figura 4.21, muestra un diagrama de tiempos que evidencia el funcionamiento del sistema completo, el cual trabajará con dos actividades principales ejecutándose en ciertos intervalos de tiempo. A partir de esto, se puede notar que el módulo de adquisición trabaja de forma periódica cada 5ms (200Hz) leyendo el dato del sensor, filtrándolo y almacenándolo. Asimismo, cuando este módulo recibe la petición de información proveniente del módulo de la interfaz gráfica, se enviará el valor obtenido como resultado del procesamiento hasta el momento antes de haber recibido dicha petición. En caso dicha petición se reciba antes de terminar de procesar el último dato muestreado, se enviará el dato producto del muestreo anterior. Luego de haber recibido dicho valor, la interfaz gráfica es actualizada para mostrar a la persona evaluada su posición durante el desarrollo de la prueba. La actualización de la interfaz, luego de recibir el dato, demora alrededor de 5.2ms, como se puede observar en la Figura 4.21.

Puede notarse que en base a dicho diagrama de tiempos el sistema desarrollado demoraría alrededor de 10.2ms en mostrar el movimiento realizado por el usuario en un instante dado. Sin embargo, en la sección 4.4. se muestra un análisis que permite identificar cuanto retardo añade la actualización de cada técnica de fusión de datos utilizado, lo cual es importante para calcular el retardo total del sistema. Asimismo, algunos estudios mencionan que el ojo humano es capaz de percibir cambios entre fotogramas que demoren entre 100ms y 150ms en actualizarse, por lo que un sistema que se actualice en menos tiempo, es considerado óptimo ya que no mermará el desempeño de la persona evaluada.

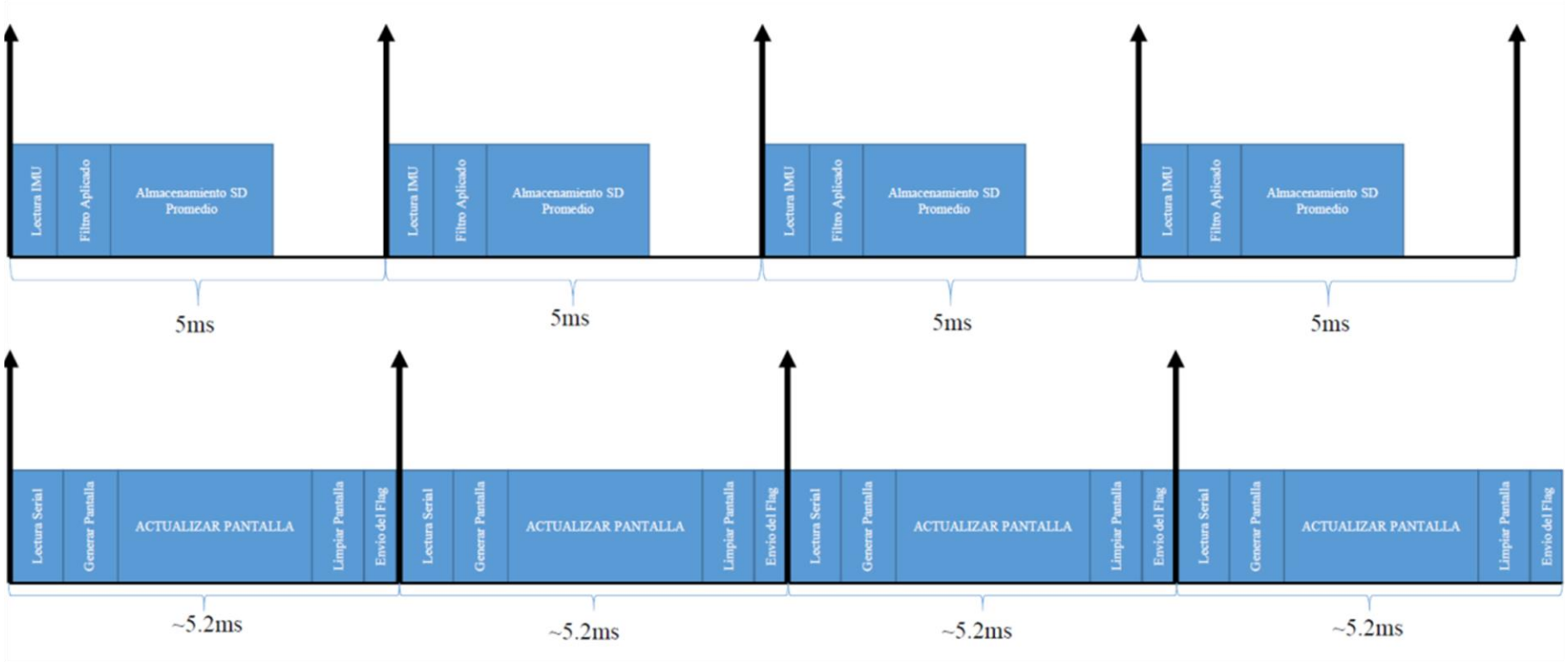


Figura 4.21: Diagrama de tiempos del sistema completo

4.4. COMPARACIÓN DE FILTROS

Este trabajo no se enfocó principalmente en el desarrollo de un algoritmo para el procesamiento de la señal proveniente del sensor inercial, debido a ello, se utilizaron librerías que permiten llevar a cabo dicho procesamiento basándose en las técnicas de 'Sensor Fusion' mencionadas en el capítulo 2 de la presente. Por medio de éstos se logró obtener resultados similares acerca de la información del ángulo correspondiente a la posición de la persona evaluada en un instante dado.

Asimismo, para medir el retardo de estimación proporcionado por cada uno de los filtros aplicados a la señal se utilizó un mecanismo de referencia que permita simular de forma precisa los movimientos realizados por la persona evaluada durante el desarrollo de la prueba. El mecanismo que se utilizó es el *Mikrolar – R3000 Rotopod* [23], el cual es un robot hexápodo giratorio de gran precisión que permite simular traslaciones en los ejes x, y, z, así como rotaciones alrededor de dichos ejes, conocidos como roll (rotación alrededor del eje x), pitch (rotación alrededor del eje y), yaw (rotación alrededor del eje z). Las Figura 4.22 y Figura 4.23 muestran la implementación realizada en dicho mecanismo para llevar a cabo las pruebas.

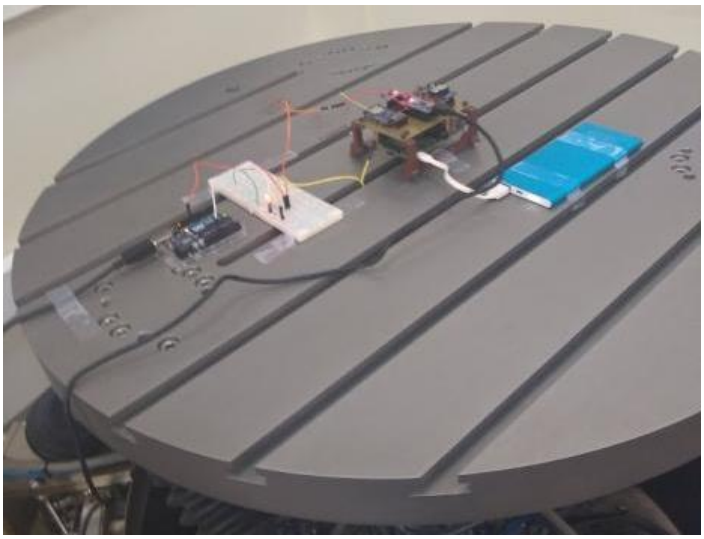


Figura 4.22: Implementación del sistema sobre la plataforma del mecanismo Rotopod para llevar a cabo las pruebas

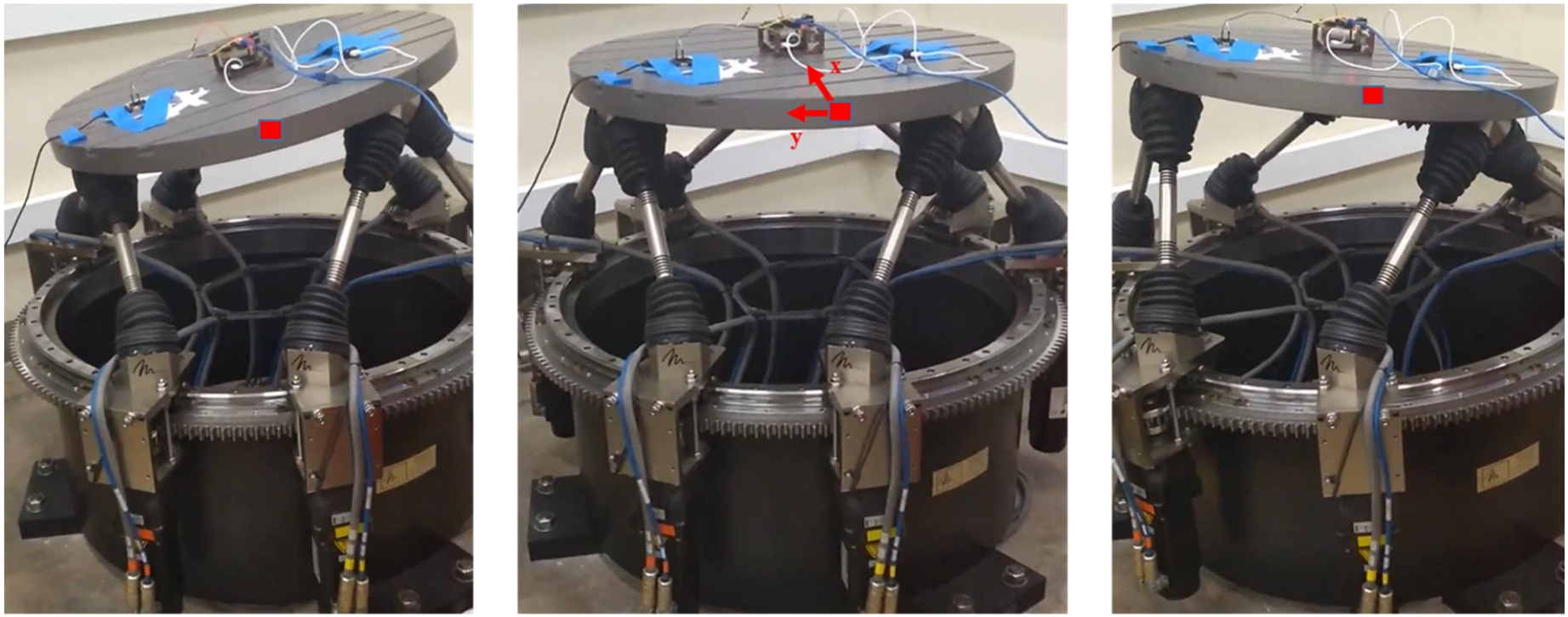


Figura 4.23: Rotación del Rotopod alrededor del eje x (sistema de referencia en rojo). a) Rotación en sentido anti-horario b) Posición de equilibrio c) Rotación en sentido horario

4.4.1. Filtro Complementario

En esta sección se mostrará el efecto que tiene aplicar el Filtro Complementario en la señal adquirida del sensor. La Figura 4.24 muestra como se ve la señal sin procesamiento alguno (señal azul), y luego a dicha señal se le aplica dicho filtro generando la respuesta mostrada (señal roja).

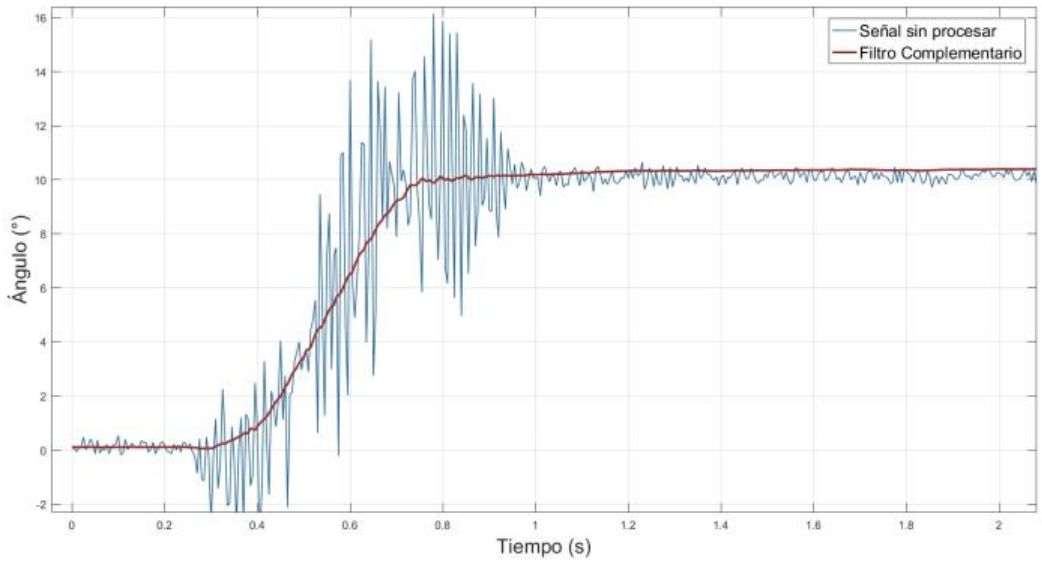


Figura 4.24: Comparación señal sin procesar – Salida del Filtro Complementario

Con la finalidad de conocer el tiempo que le demanda a este filtro actualizarse al valor deseado se realizaron pruebas con el sistema Mikrolar Rotopod, cuyos resultados son mostrados en las Figura 4.25 y Figura 4.26.

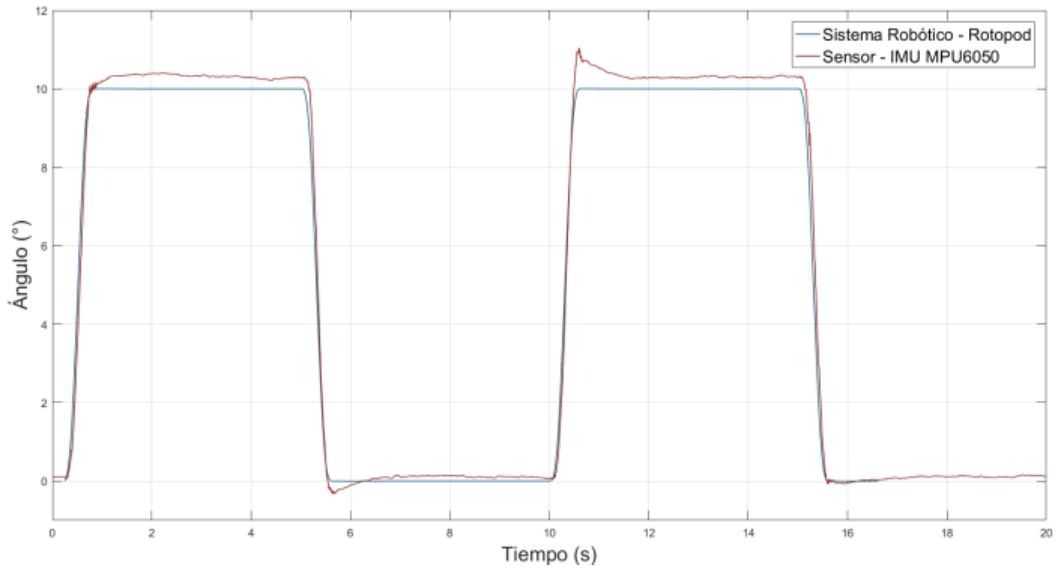
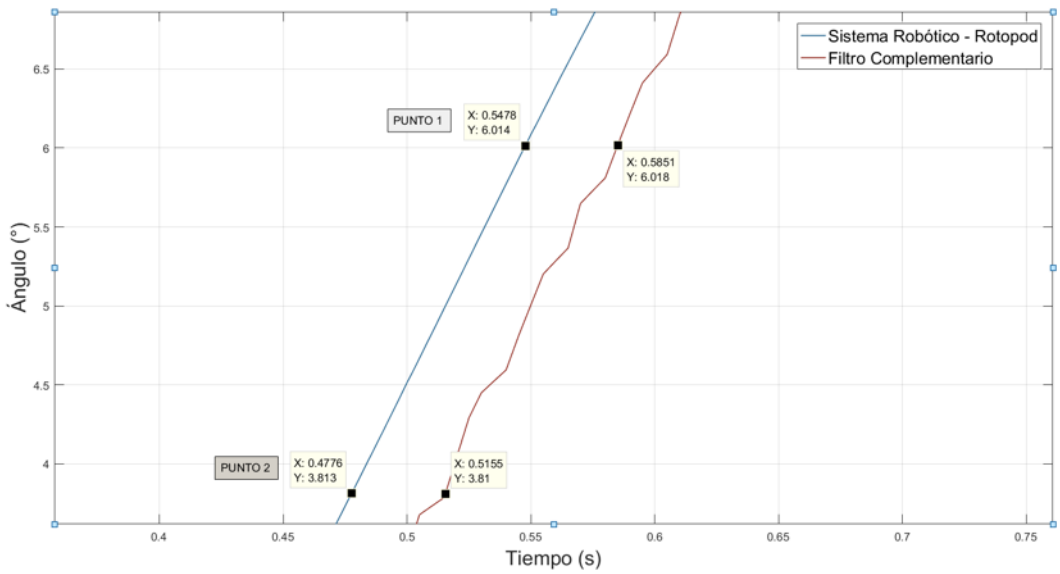
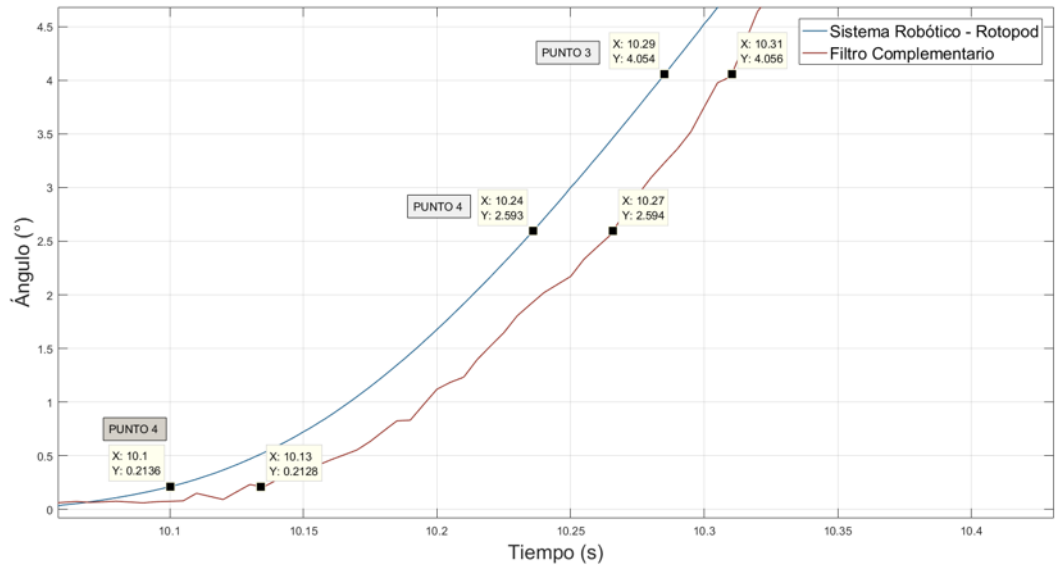


Figura 4.25: Comparación del desempeño Mikrolar Rotopod – Filtro Complementario



a)



b)

Figura 4.26: a) Medición de retardos para la primera subida b) Medición de retardos para la segunda subida

La Tabla 4.4 muestra el retardo total aproximado del sistema completo, en base al número de muestras que requiere el filtro para actualizarse al valor final deseado, y al tiempo de retardo de la impresión de la interfaz gráfica (5.2ms). El cálculo de dicho valor es mostrado en la siguiente ecuación:

$$Retado\ total\ del\ sistema - Complementario\ (ms) = \#\ de\ muestras * \frac{1}{F_s} + 5.2ms$$

Tabla 4.4: Retardos de estimación del Filtro Complementario

	T1 (s)	T2 (s)	Retardo de estimación (ms)	Numero de muestras (Fs=200Hz)	Retardo total aproximado del sistema completo (ms)
PUNTO 1	0.5478	0.5851	37.3	8	45.2
PUNTO 2	0.4776	0.5155	37.9	8	45.2
PUNTO 3	10.29	10.31	20	5	30.2
PUNTO 4	10.24	10.27	30	6	35.2
PUNTO 5	10.1	10.13	30	7	40.2

Como conclusión, se puede decir que utilizando este filtro se puede obtener un sistema que tenga un retardo aproximado entre 30.2ms y 45.2ms.

4.4.2. Filtro Kalman

En esta sección se mostrará el efecto que tiene aplicar el filtro Kalman en la señal adquirida del sensor. La Figura 4.27 muestra como se ve la señal del sensor sin procesamiento (señal azul), y luego a dicha señal se le aplica el Filtro Kalman utilizado generando la respuesta mostrada (señal roja).

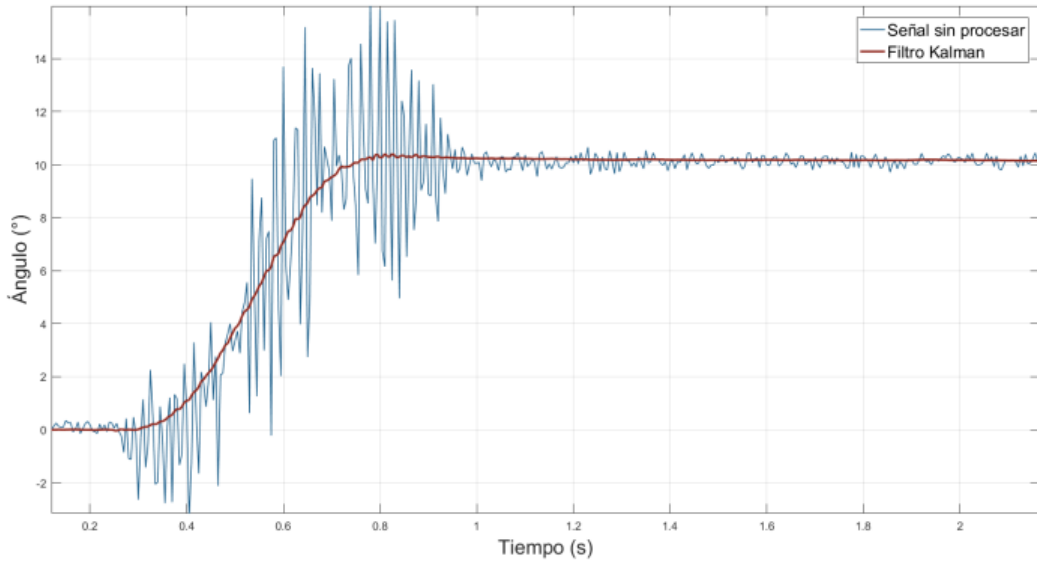


Figura 4.27: Comparación señal sin procesar – Salida del Filtro Kalman

Al igual que en el caso anterior, se utilizó el mecanismo Mikrolar Rotopod y se comparó su desempeño con el resultado obtenido en la salida del Filtro Kalman. Los resultados se muestran en la Figura 4.28 y Figura 4.29.

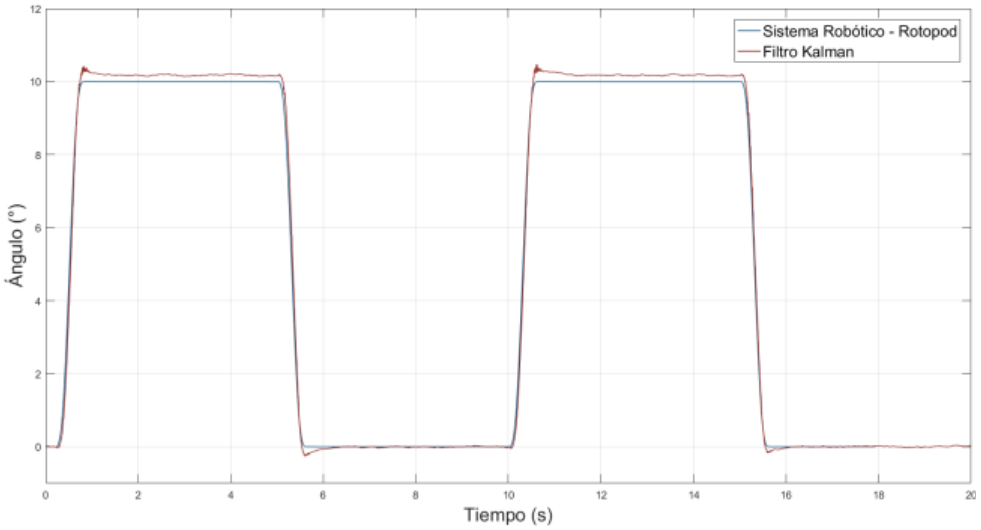
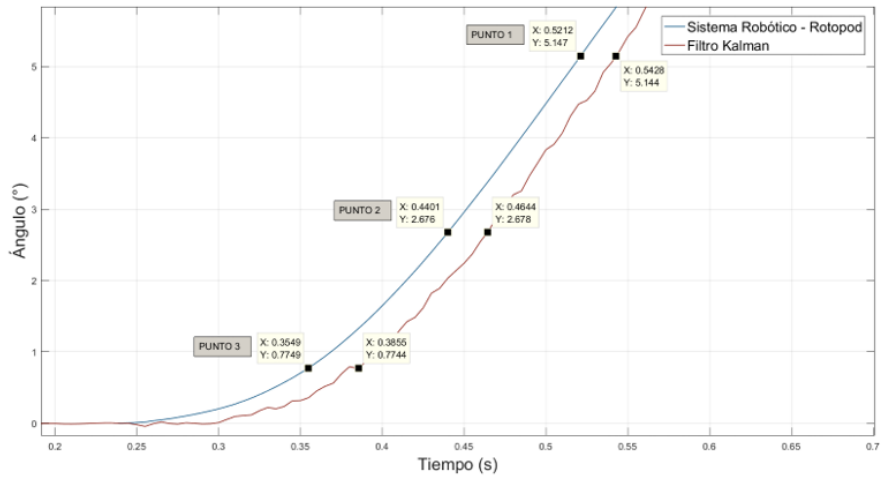
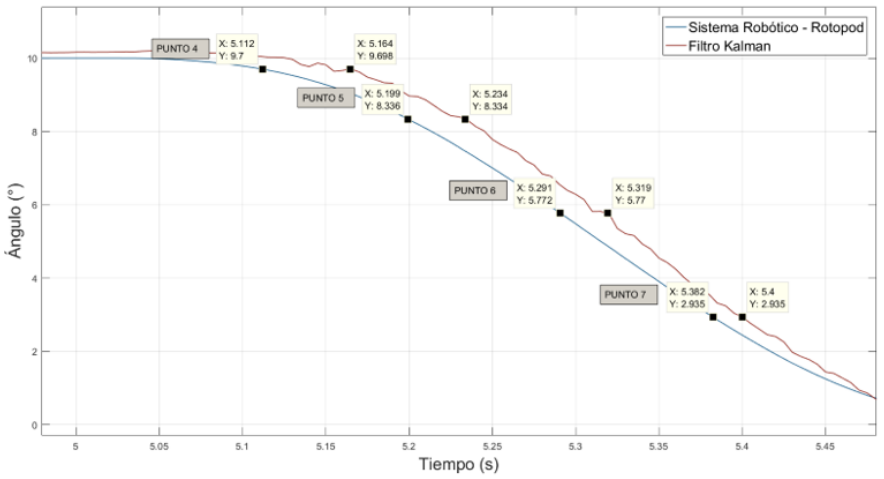


Figura 4.28: Comparación del desempeño Mikrolar Rotopod – Filtro Kalman



a)



b)

Figura 4.29: a) Medición de retardos para la primera subida b) Medición de retardos para la primera bajada

La Tabla 4.5 muestra el retardo total aproximado del sistema completo, en base al número de muestras que requiere el presente filtro para actualizarse al valor final deseado, y al tiempo de retardo de la impresión de la interfaz gráfica (5.2ms). El cálculo de dicho valor es mostrado en la siguiente ecuación:

$$Retado\ total\ del\ sistema - Kalman\ (ms) = \#\ de\ muestras * \frac{1}{F_s} + 5.2ms$$

Tabla 4.5: Retardos de estimación del Filtro Kalman

	T1 (s)	T2 (s)	Retardo de estimación (ms)	Numero de muestras (Fs=200Hz)	Retardo total aproximado del sistema completo (ms)
PUNTO 1	0.5212	0.5428	21.6	5	30.2
PUNTO 2	0.4401	0.4644	24.3	5	30.2
PUNTO 3	0.3549	0.3855	30.6	7	40.2
PUNTO 4	5.112	5.164	52	11	60.2
PUNTO 5	5.199	5.234	35	8	45.2
PUNTO 6	5.291	5.319	28	6	35.2
PUNTO 7	5.382	5.4	18	4	25.2

Como conclusión, se puede decir que utilizando este filtro se puede obtener un sistema que tenga un retardo aproximado entre 25.2ms y 60.2ms.

4.5. OPTIMIZACIÓN DE TIEMPOS DEL SISTEMA

4.5.1. Reducción del tiempo de impresión de la pantalla

Al inicio del desarrollo del proyecto se notaba que la impresión de la pantalla visualizada, lo cual comprende la generación de la misma, su actualización y su limpieza correspondiente, requería mucho tiempo para ejecutarse de forma completa y óptima. Esto conllevaba a que el sistema presente un retraso mayor, que el mostrado en la sección de resultados, durante su ejecución. Dicho tiempo era de aproximadamente 20ms para imprimir la totalidad de la interfaz. Con la finalidad de reducir dicho tiempo, se modificó la forma en la que cada instrucción era utilizada en el programa. Con esto se logró reducir dicho tiempo de ejecución a los 5.2ms que se tienen actualmente, permitiendo la reducción del retardo total del sistema.

La Tabla 4.6 muestra una comparación respecto a la forma en cómo se aplicaron las funciones de la librería en cada caso, y como se obtuvieron las mejoras descritas.

Tabla 4.6: Mejoras desarrolladas en la impresión de la interfaz

	Tiempo de ejecución	Generación de Pantalla	Actualización de pantalla	Limpieza de la Pantalla
Forma de impresión inicial	**~20ms**	Impresión de imágenes en formato .jpg, .png, etc	Refresco de la totalidad de la pantalla	Limpieza de la pantalla en su totalidad
Forma de impresión final	**~5.2ms**	Impresión de formas geométricas (circulos, rectángulos, cuadrados)	Refresco de ciertas zonas de la pantalla que requerian actualizarse	Limpieza de la pantalla solo en las zonas donde ocurrían cambios

4.5.2. Transferir tarea de grabación a la Raspberry

Como se explicó en la sección 3.3.3, el sistema diseñado almacena toda la información correspondiente a una prueba en una unidad de almacenamiento externa, utilizando el módulo Micro SD detallado. Esta elección se realizó ya que, como se mencionó en el ítem 4.5.1, el tiempo que demoraba la Raspberry Pi en actualizar la interfaz gráfica en la pantalla era de aprox. 20ms. Durante ese periodo, se evidenciaban pérdidas de datos debido a la condición mostrada en la Tabla 4.7.

Tabla 4.7: Número de muestras adquiridas durante el intervalo de impresión

	Tiempo de impresión (ms)	Tiempo de muestreo de la unidad de adquisición (ms)	Número de muestras adquiridas en dicho intervalo
Primera versión de impresión de pantalla	20	5	4

La Tabla 4.7 muestra que un dato se enviaba a la Raspberry Pi cada 20ms, siendo muestreados, en dicho período, 4 datos por la unidad de adquisición. Dado que en ese intervalo de tiempo no se enviaba información a la Raspberry, dicha información correspondiente a los 3 datos restantes, se ignoraba, generando un retardo mayor en la realimentación hacia la persona evaluada.

Empero, al reducir el tiempo de impresión de la pantalla a 5.2ms, se observó que era posible transferir la tarea de almacenamiento hacia la Raspberry, sin evidenciar pérdida de información considerable como en lo previamente mostrado. Dicho cambio se desarrolló a nivel de software. En la sección C de Anexos se puede visualizar los códigos desarrollados para ambos casos.

A continuación, se procederán a presentar algunas de las características del dispositivo desarrollado. En ese sentido la Figura 4.30, evidencia el costo del sistema desarrollado, así

como una comparación con los precios de los dispositivos presentes en el mercado actual. En conclusión, se puede notar que el dispositivo cumple con el requisito de bajo costo.

Componente	Costo
Sensor inercial MPU6050	S/ 8.00
Power Booster 1000C	S/ 65.80
Bateria BTL 6090 3.7v 4000mAh	S/ 20.00
Módulo Micro SD	S/ 12.00
RTC DS3231	S/ 13.00
Arduino Nano	S/ 25.00
Raspberry Pi 3B	S/ 210.00
Tarjeta SD 32 GB para la Raspberry	S/ 25.00
Tarjeta SD 8 GB para el almacenamiento	S/ 16.00
Total	S/ 394.80
	$ 116.12

OptiTrack	$ 36,852.00
NeuroCom Balance Master System	$ 8,000.00
BTrackS Assess Balance Advanced System	$ 2,100.00

Figura 4.30: Comparación de precios del sistema desarrollado vs tecnologías existentes

Asimismo, se midió que el dispositivo desarrollado tiene un peso de 190g, lo cual lo hace un sistema liviano para ser usado sobre la espalda baja de la persona sin afectar su desempeño. Asimismo, en base a la R.M. 375-2008-TR [24], la cantidad de peso máximo que puede cargar una persona depende de su sexo, siendo de 25kg para los hombres y 15 kg para las mujeres. Por lo tanto, se puede concluir que el peso del sistema es adecuado y no afectará el desempeño de la persona evaluada. De esta forma es posible tener un sistema portable ya que su configuración y su peso facilita su traslado hacia el lugar donde se desee llevar a cabo la prueba.

Finalmente, se puede notar, en base a la comparación de tiempos realizada en la sección 4.4., que la técnica de Fusión de Datos a emplear en el presente sistema es el Filtro Complementario, ya que evidencia una respuesta óptima en términos de tiempo de respuesta, tiempo de actualización y precisión, con el hardware desarrollado. Dicho esto, las bondades de un Filtro Kalman podrían ser mejor aprovechadas teniendo un hardware apropiado para llevar a cabo dicho procesamiento para la fusión de los datos utilizados.

CONCLUSIONES

1. Este trabajo desarrolla un sistema de adquisición para el análisis de la respuesta en tiempo real de la columna vertebral al realizar movimientos de inclinación, hacia la derecha o a la izquierda, al sentarse frente a un monitor que despliega la interfaz gráfica desarrollada. Los ángulos de inclinación de la columna vertebral se obtienen a partir de la fusión de la data adquirida tanto del acelerómetro como del giroscopio del sensor inercial utilizado. Esta información ha sido procesada utilizando los algoritmos de Filtro de Kalman y Filtro Complementario. Asimismo, se analizaron los tiempos de respuesta presentada por cada filtro para compararlos y llegar a la conclusión que ambos filtros presentan una respuesta similar, siendo el filtro Complementario el que alcanzó un retardo promedio menor que el filtro Kalman utilizado.

2. Las tecnologías descritas a lo largo de la tesis fueron utilizadas y se desarrolló un sistema que permite adquirir la data del sensor seleccionado, almacenar dicha data en una unidad de almacenamiento externo, y a la vez, permite mostrar la posición del tronco de la persona evaluada, en un instante de tiempo, en una pantalla. La información almacenada permitirá realizar análisis posteriores, no correspondiente a la tesis, para identificar el delay y lag presentes en el seguimiento del objetivo visual por parte de la persona evaluada.

3. Se desarrolla una interfaz gráfica programada en el módulo Raspberry Pi 3. De esta manera, se puede realizar la prueba conectando el dispositivo a cualquier monitor mediante el estándar HDMI. Asimismo, dicha interfaz se desarrolló con la finalidad de ser un entorno amigable, permitiendo su configuración entre dos modos de operación según disponga el evaluador, los cuales son modo ‘Demo’ y modo ‘Trial’, así como permitiendo la etiquetación de la persona evaluada en cada caso. En base a lo descrito, se concluye que es posible evaluar el balance corporal de una persona en cualquier establecimiento debido a la portabilidad, autonomía y manejabilidad que presenta la interfaz desarrollada.

4. Se logra reducir el retardo presentado por el sistema desarrollado en cada una de sus etapas presentadas. Siendo, en un inicio, el caso más crítico, la impresión de la pantalla, la cual demoraba alrededor de 20ms, reduciendo dicho valor a aprox. 5.2ms. Con esto,

se puede decir que el sistema desarrollado se encuentra optimizado en términos de tiempo de respuesta para la realización de las pruebas del balance corporal.

RECOMENDACIONES

- En las pruebas realizadas, la unidad de adquisición muestrea la data del sensor cada 5ms, por lo que la respuesta del filtro permanecía condicionado a dicho periodo de muestreo. En ese sentido, se recomienda incrementar dicha frecuencia de muestreo, siendo 1kHz el valor máximo aceptado por el sensor, con lo cual los filtros utilizados recibirían más datos en un intervalo menor produciendo que cada filtro se actualice en menor tiempo. Asimismo, es necesario tener en consideración que, para realizar el muestreo de la señal a dicha frecuencia, los filtros deben ser lo suficientemente livianos en código, pero igualmente eficientes, tal que logren ejecutarse completamente en dicho intervalo de 1ms.

- Migrar la unidad de adquisición a un sistema de 32 bits y con mayor velocidad de procesamiento mejorará la eficiencia de procesamiento del filtro Kalman, el cual debe ser implementado en unidades de este tipo ya que requiere de operaciones matemáticas de gran complejidad. Por el contrario, una unidad de 8 bits, como es el microcontrolador Arduino Nano, no tiene la capacidad de procesamiento suficiente ni el hardware apropiado para llevar a cabo dichas operaciones (multiplicaciones y divisiones de punto flotante) en una sola instrucción, lo que conllevo a que demande más tiempo de procesamiento durante su ejecución, añadiendo más retardo al sistema. Por otro lado, la unidad de 32 bits permitiría llevar a cabo dichas operaciones en menos cantidad de instrucciones, permitiendo reducir el retardo de respuesta del filtro, así como el retardo del sistema completo.

BIBLIOGRAFÍA

[1] Y.-H. Pua, P.-H. Ong, R. A. Clark, D. B. Matcher, and E. C.-W. Lim, "Falls efficacy, postural balance, and risk for falls in older adults with falls-related emergency department visits: prospective cohort study," *BMC Geriatr.*, vol. 17, no. 1, p. 291, 2017.

[2] "Leading Causes of Traumatic Brain Injury," *BrainLine*. [Online]. Available: https://www.brainline.org/slideshow/infographic-leading-causes-traumatic-brain-injury. [Accessed: 28-Oct-2018].

[3] N. P. Reeves, A. Luis, E. C. Chan, V. G. Sal y Rosas, and M. L. Tanaka, "Assessing delay and lag in sagittal trunk control using a tracking task," *J. Biomech.*, vol. 73, pp. 33–39, 2018.

[4] A. Mansfield, A. L. Peters, B. A. Liu, and B. E. Maki, "A perturbation-based balance training program for older adults: Study protocol for a randomised controlled trial," *BMC Geriatr.*, vol. 7, pp. 1–14, 2007.

[5] "NeuroCom _ OrthoBalance Physical Therapy." [Online]. Available: https://www.orthobalancept.com/neurocom/%0A.

[6] R. A. Clark *et al.*, "Gait & Posture Validity of the Microsoft Kinect for assessment of postural control," *Gait Posture*, vol. 36, no. 3, pp. 372–377, 2012.

[7] D. Lim, "Use of the Microsoft Kinect system to characterize balance ability during balance training," pp. 1077–1083, 2015.

[8] R. Y. W. Lee, J. Laprade, and E. H. K. Fung, "A real-time gyroscopic system for three-dimensional measurement of lumbar spine motion," vol. 4533, no. January, 2004.

[9] Y. Zhang, Y. Fei, L. Xu, and G. Sun, "Micro-IMU-Based Motion Tracking System for Virtual Training," *2015 34th Chinese Control Conf.*, pp. 7753–7758, 2015.

[10] J. M. Cortell-tormo, M. García-jaén, D. Ruiz-fernández, and S. Member, "Lumbatex : A Wearable Monitoring System Based on Inertial Sensors to Measure and Control the Lumbar Spine Motion," vol. 27, no. 8, pp. 1644–1653, 2019.

[11] X. Yan, H. Li, A. R. Li, and H. Zhang, "Automation in Construction Wearable IMU-

based real-time motion warning system for construction workers ’ musculoskeletal disorders prevention,” *Autom. Constr.*, vol. 74, pp. 2–11, 2017.

[12] R. Peter, “Sumaq Life _.” [Online]. Available: https://sumaqlife.com/.

[13] P. Iii, “Parte III Diseño de sensores inerciales micromaquinados,” pp. 255–266.

[14] S. Colton, “The Balance Filter, A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform,” *Chief Delphi White Pap.*, p. 20, 2007.

[15] S. Colton, “The Balance Filter, A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform,” *Chief Delphi White Pap.*, p. 20, 2007.

[16] Mechatronics Naylamp, “Tutorial MPU6050, Acelerómetro y Giroscopio,” 2018. [Online]. Available: https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Acelerómetro-y-Giroscopio.html%0Ahttp://www.naylampmechatronics.com/blog/45_Tutorial-MPU6050-Acelerómetro-y-Giroscopio.html.

[17] E. Malinen, “Fusion of Data from Quadcopter ’ s Inertial Measurement Unit Using Complementary Filter,” 2015.

[18] T. Ameid, A. Menacer, H. Talhaoui, and I. Harzelli, “Rotor resistance estimation using Extended Kalman filter and spectral analysis for rotor bar fault diagnosis of sensorless vector control induction motor,” *Measurement*, 2017.

[19] Newark Farnell, “Raspberry Pi 3 Model B RASPBERRYPI3-MODB-1GB RPI3-MODB-16GB-NOOBS,” *Datasheets*, 2018. [Online]. Available: http://www.farnell.com/datasheets/2032161.pdf.

[20] Sparkfun Electronics, “Raspberry Pi 3 - SparkFun Electronics.” .

[21] U. S. B. Hub and E. Controller, “LAN9514/LAN9514i USB 2.0 Hub and 10/100 Ethernet Controller Data Sheet,” pp. 1–54, 2016.

[22] “pygame.” [Online]. Available: https://www.pygame.org/news. [Accessed: 20-Sep-2018].

[23] “Mikrolar - R3000.” [Online]. Available: http://mikrolar.com/r3000.html. [Accessed:

07-Feb-2019].

[24] S. D. E. Ergonom, D. E. P. D. E. Evaluaci, and D. E. R. Disergon, “Rm 375-2008-tr.” 2008.

[25] “A fondo_ La cabeza y la columna vertebral_ EL RAQUIS_ MOVIMIENTOS DE LA COLUMNA VERTEBRAL.” [Online]. Available: http://cabezaycolumnavertebral.blogspot.com/2014/12/el-raquis-movimientos-de-la-columna.html.

[26] “What is VGA (Video Graphics Array)_.” [Online]. Available: https://www.computerhope.com/jargon/v/vga.htm.

[27] “Todo HDMI - Tecnología & Informática.” [Online]. Available: https://tecnologia-informatica.com/todo-hdmi/.

[28] “HDMI (High-Definition Multimedia Interface) Definition.” [Online]. Available: https://techterms.com/definition/hdmi.

[29] M. Mod-mpu, “Módulo MPU6050 , Acelerómetro , Giroscopio I2C,” 2018. [Online]. Available: https://naylampmechatronics.com/sensores-posicion-inerciales-gps/33-modulo-mpu6050-acelerometro-giroscopio-i2c.html.

[30] “MPU9250 9-DOF 3-Axis Accelerometer, Gyro, & Magnetometer Philippines _ Makerlab Electronics.” [Online]. Available: https://www.makerlab-electronics.com/product/mpu9250-9-dof-3-axis-accelerometer-gyro-magnetometer/.

[31] “LSM9DS1 Breakout Hookup Guide - learn.” [Online]. Available: https://learn.sparkfun.com/tutorials/lsm9ds1-breakout-hookup-guide/all.

[32] ___, “Arduino Uno Rev3,” *Store.Arduino.Cc*. p. 1, 2018.

[33] Arduino, “Arduino Nano.” 2019.

[34] Arduino, “Arduino Mega 2560 Rev3,” *Consultado 02 de Enero del 2018*, 2017. [Online]. Available: https://store.arduino.cc/usa/arduino-mega-2560-rev3.

[35] Newark Farnell, “Raspberry Pi 3 Model B RASPBERRYPI3-MODB-1GB RPI3-MODB-16GB-NOOBS,” *Datasheets*, 2018. .

[36] B. Ave, D. Number, and R. Date, “MPU-6000 and MPU-6050 Product Specification,” vol. 1, no. 408, 2013.

[37] Raspberry Pi, “Raspberry Pi Pinout Diagram_ Navigating the Raspberry Pi 3 Model B.” [Online]. Available: https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html. [Accessed: 09-May-2019].

[38] P. M. Utc, “Adafruit Powerboost 1000C LEDs Assembly Attaching USB connector On / Off Switch,” 2019.

[39] “PowerBoost 1000 Charger - Rechargeable 5V Lipo USB Boost @ 1A [1000C] ID_ 2465 - $19.” .

ANEXOS

A) Desarrollo de prototipos

A.1) Primera Alternativa de Solución

ARDUINO UNO – DRIVER VGA + ARDUINO MEGA – SENSOR IMU

Esta tesis comenzó con la idea de implementar un sistema de bajo costo para realizar el análisis del balance corporal. Con este fin, se elaboró una primera solución la cual consistió en utilizar microcontroladores de bajo costo, tal es el caso de los microcontroladores Arduino, encargados de toda la operación del sistema.

Este prototipo se basa en la utilización de dos módulos. Por un lado, uno de ellos se dedica a capturar la data proveniente del sensor y almacenarla en un espacio de memoria, ya sea externo o interno. Por otro lado, el segundo módulo se dedicará a desplegar la interfaz gráfica con la cual se llevará a cabo la prueba. Asimismo, ambos módulos estarán comunicados por medio de un protocolo de comunicación tal que puedan intercambiar información. Dicho intercambio es necesario para poder desplegar la posición del tronco de la persona (lectura del sensor IMU) en la interfaz, y de esta forma el paciente pueda mejorar su desempeño.

- Interfaz Gráfica:

  Para empezar con las pruebas, se prestó mayor énfasis en desarrollar una interfaz que evidencie gran desempeño tanto en calidad de gráficos como en rapidez de visualización. Por esta razón, se ideó una primera solución basada en el desarrollo de la interfaz gráfica en un microcontrolador Arduino UNO. Para este propósito se utilizó una librería que permitía desplegar gráficos en una pantalla por medio del estándar VGA utilizando el Microcontrolador Arduino Uno. Asimismo, esta implementación requería la fabricación de un driver que permita la conexión de pines del microcontrolador con los pines correspondientes del puerto VGA, vea la Figura A.1 . Dicha librería se denomina “VGAX Library for Arduino UNO” [1] . Esta librería opera en microcontroladores de la familia ATMega328 utilizada por los módulos Arduino UNO, Arduino Nano y Arduino Mini-Pro.

  Dicha librería utiliza los 3 Timers propios del microcontrolador (Timer0, Timer1 y Timer2), y a la vez hace uso de todas las interrupciones proporcionadas por dicho

módulo. Este hecho restringe la utilización de dichas capacidades para otro fin distinto dentro del programa. Asimismo, esta librería utiliza gran cantidad de variables, las cuales son almacenadas en la memoria RAM propia del microcontrolador (Arduino UNO o NANO = 2KB). Por dicho motivo, es necesario que el programa implementado sea liviano tanto en cantidad de instrucciones como en cantidad de variables utilizadas, para que el Arduino funcione de forma correcta, de lo contrario no será posible la programación del mismo.

Este prototipo desarrollado contará con las siguientes características:

- Mostrará un objetivo cuya posición vista desde el plano frontal cambiará de izquierda a derecha, considerando ángulos de 4, 6 y 8° cada cierto intervalo de tiempo establecido.
- Mostrar la lectura del sensor mientras la persona interactúa con la interfaz para seguir la posición del objetivo mostrado en pantalla.
- Los cambios de posición para la persona evaluada son factores aleatorios desconocidos para la persona evaluada.

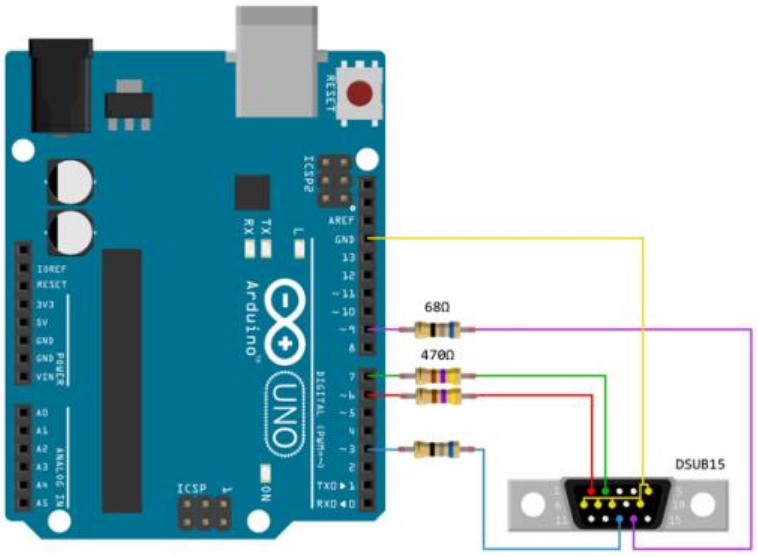


Figura A.1: Conexión Arduino UNO con los pines del puerto VGA [1]

Como se observa en la Figura A.1., los pines del puerto VGA se conectan por medio de resistencias hacia los pines del microcontrolador. Esta librería permite el despliegue de diversos tipos de colores por pixel, cuya selección depende del tipo de conexionado realizado entre los pines 6 y 7 del microcontrolador, con los pines 1, 2 y 3 del conector VGA.

Luego de haber implementado el driver para llevar a cabo las pruebas pertinentes, se procedió a realizar la implementación del programa que permita mostrar la interfaz requerida en la pantalla utilizando el estándar de imagen VGA. Sin embargo, el programa desarrollado, considerando todas las variables requeridas por la librería y el por el propio programa, ocupaba alrededor del 97% de la memoria RAM del Arduino UNO. Debido a esto, resultaba prácticamente imposible incluir tareas adicionales como la comunicación serial, o comunicación I2C, motivo por el cual no se podía tener información adicional proveniente de sensores externos. Por dicho motivo, no se logró utilizar el sensor inercial MPU6050 en esta versión debido a lo descrito previamente, en su reemplazo se simuló dicho comportamiento utilizando un potenciómetro, el cual permitía visualizar una posición en la pantalla.

Los resultados obtenidos referente a la interfaz gráfica desarrollada se muestran en las siguientes figuras, así como su respectiva descripción.

Figura A.2: Pantalla de inicio del programa

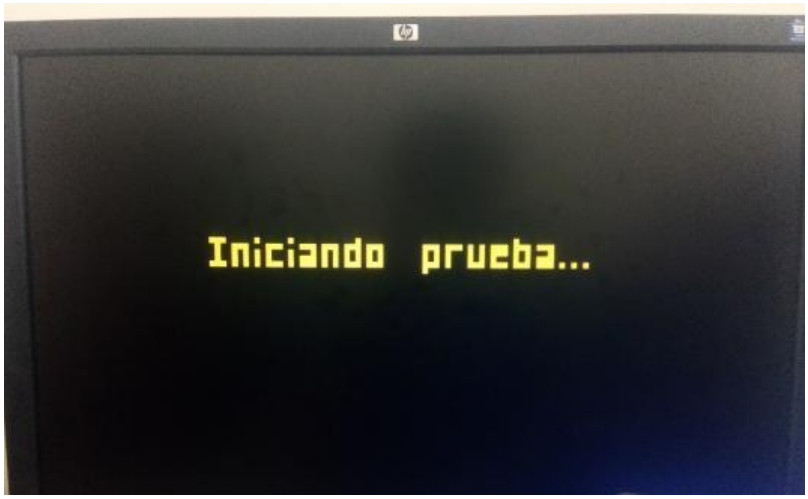


Figura A.3: Pantalla de inicio de la prueba

Figura A.4: Cuenta atrás

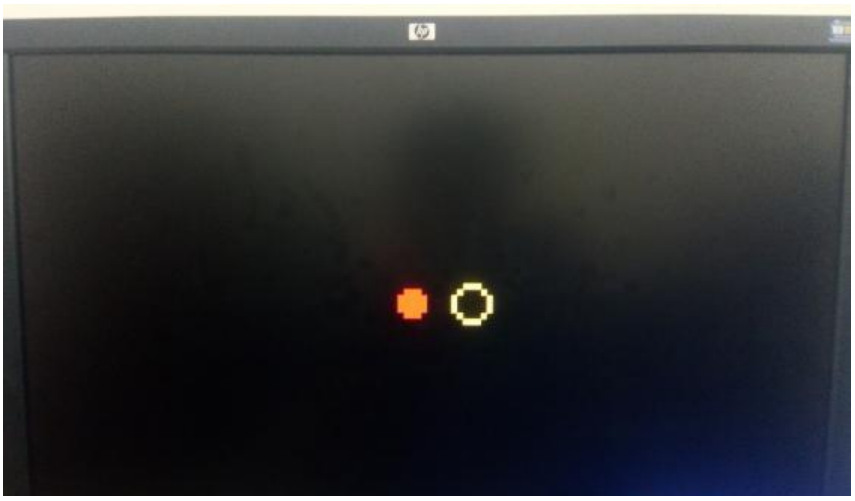


Figura A.5: Desarrollo de la prueba (circulo amarillo: objetivo, círculo rojo: posición potenciómetro)

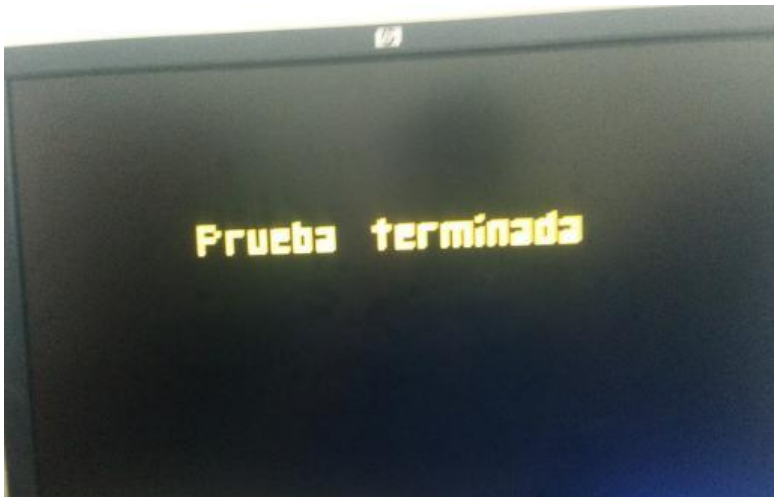


Figura A.6: Pantalla de fin de la prueba

Finalmente, este prototipo permitió el despliegue de una interfaz gráfica con una resolución de gráficos máxima de 120x60pixels, lo cual, como se puede notar en las Figuras anteriores, distorsiona la calidad visual de la interfaz. Por esta razón, no era adecuada dicha opción para llevar a cabo la prueba. Asimismo, se notó que dicho programa ocupaba prácticamente la totalidad del espacio de memoria RAM disponible en el módulo Arduino UNO. Dicho factor impedía la adición de funcionalidades al programa, tales como intercambiar información con el módulo de adquisición. Debido a lo descrito, el presente prototipo fue descartado, sin necesidad de implementar la unidad de adquisición, y se procedió con el diseño de otro prototipo.

A.2) Segunda Alternativa de Solución

RASPBERRY PI 3B + SENSOR IMU + HDMI

Con la finalidad de desarrollar una interfaz que supere la calidad visual del prototipo previo, se decidió utilizar un estándar de transmisión de video comercial. Asimismo, se consideró necesario utilizar un dispositivo con mayor capacidad de memoria para llevar a cabo la ejecución del programa en su totalidad.

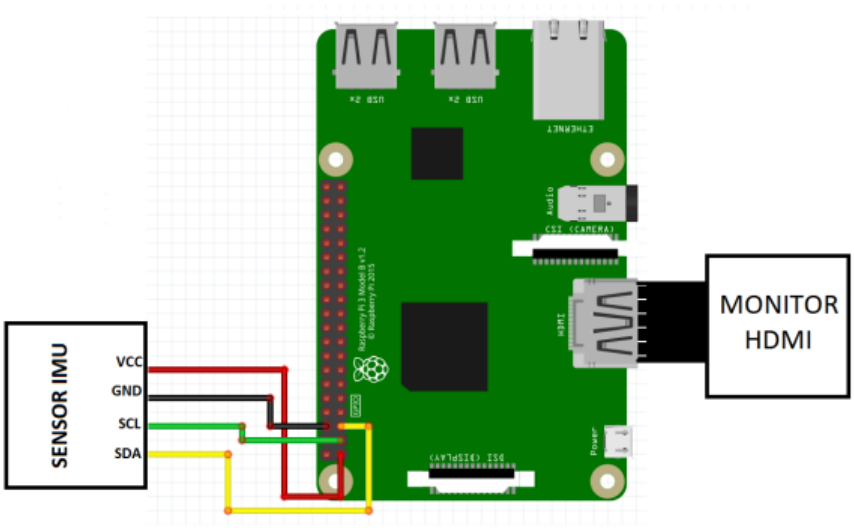


Figura A.7: Diagrama de conexiones del segundo prototipo

Como se puede apreciar en la Figura A.7, esta solución está compuesta por un único módulo encargado de desplegar la interfaz gráfica requerida para llevar a cabo la evaluación. A la vez, dicho módulo deberá capturar la información proveniente del sensor, procesarla y almacenarla en un espacio de memoria para su posterior evaluación.

Según los requerimientos del sistema, este prototipo consistió en la utilización de una tarjeta de evaluación Raspberry Pi 3B, la cual es considerada como una minicomputadora ya que cuenta con su propio sistema operativo. Las principales características de dicha tarjeta son:

- CPU Quad Core 1.2GHz Broadcom BCM2837 64bit
- 1GB RAM
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 puertos USB
- Salida de audio y video compuesto
- Full size HDMI
- Puerto Micro SD para cargar el sistema operativo y el almacenamiento de datos
- Alimentación tipo Micro USB de hasta 2.5 A

Tal como se describe en las características, la Raspberry Pi 3B cuenta con 40 pines GPIO, por medio de los cuales es posible leer la información del sensor inercial

utilizando el protocolo I2C. Asimismo, dicho módulo cuenta con un conector HDMI que permite el despliegue de gráficos de alta calidad, superando lo mostrado por el prototipo anterior.

- Lectura del sensor

  Para poder acceder a la información proporcionada por el sensor inercial se utilizó una librería enteramente desarrollada en el lenguaje Python. Dicha librería se denomina "RTIMULib" [2], y permite llevar a cabo la lectura y fusión de los distintos datos provenientes del sensor. La fusión de datos permite tener una medición mucho más precisa de la posición de la persona en un instante específico. Asimismo, a partir de dicha Fusión de datos es posible obtener información útil para la prueba como los ángulos de rotación del sensor alrededor de los distintos ejes, tales como roll, pitch y yaw.

  La librería puede trabajar con sensores inerciales pertenecientes a diversas marcas, sin embargo, el único sensor con el que se contaba de dicha extensa lista era el sensor inercial LSM9DS1.

  Para poder obtener mediciones precisas con los datos proporcionados por el sensor, es necesario realizar una calibración previa del módulo inercial utilizado. Este procedimiento permite obtener los parámetros óptimos para llevar a cabo la Fusión de Datos de la información disponible. Luego de llevar a cabo dicha calibración la posición brindada por el sensor tendrá gran precisión.

La Figura A.8 muestra la conexión realizada entre el sensor inercial LSM9DS1 y la tarjeta Raspberry Pi 3B. A partir de esta conexión, es posible obtener las lecturas mostradas en la Figura A.9, donde se muestra la posición del sensor, en un instante dado, en términos de roll, pitch y yaw.

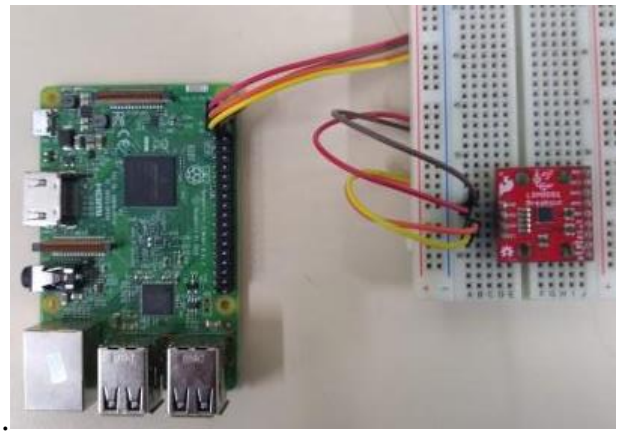


Figura A.8: Implementación de la conexión entre el sensor LSM9DS1 y la Raspberry Pi 3B

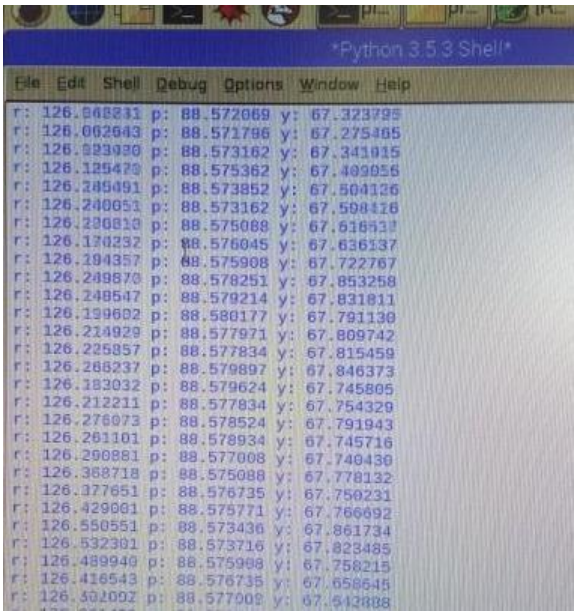


Figura A.9: Data obtenida de la fusión de datos de la señales del sensor

- Desarrollo de la Interfaz Gráfica

  Para el desarrollo de la interfaz gráfica se utilizó una librería desarrollada en Python que permite la implementación de videojuegos en dos dimensiones, mediante el despliegue de gráficos en pantalla, conocida como “pygame” [3]. Dicha librería despliega la interfaz a través del protocolo de video HDMI propio de la RAPSBERRY PI. Por dicho motivo, en este prototipo, la resolución obtenida depende exclusivamente de la pantalla con la que se dispone para llevar a cabo la prueba.

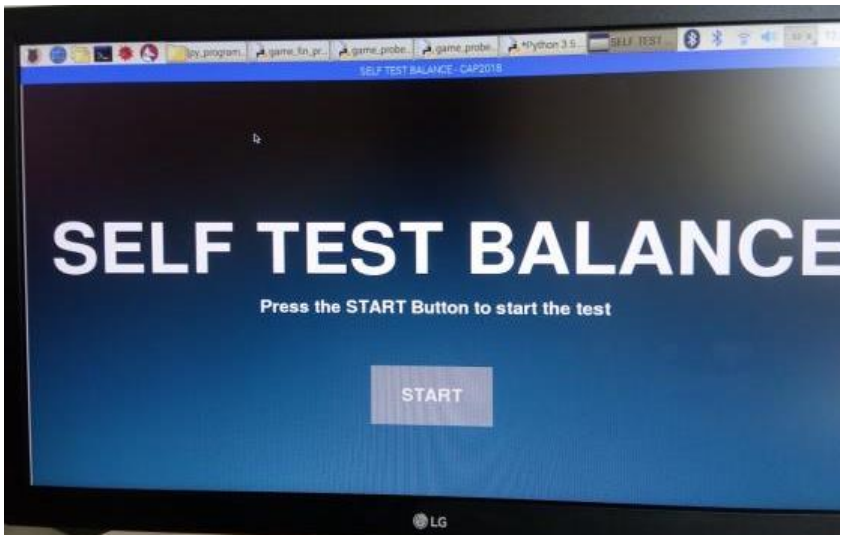


Figura A.10: Pantalla de inicialización de la interfaz gráfica

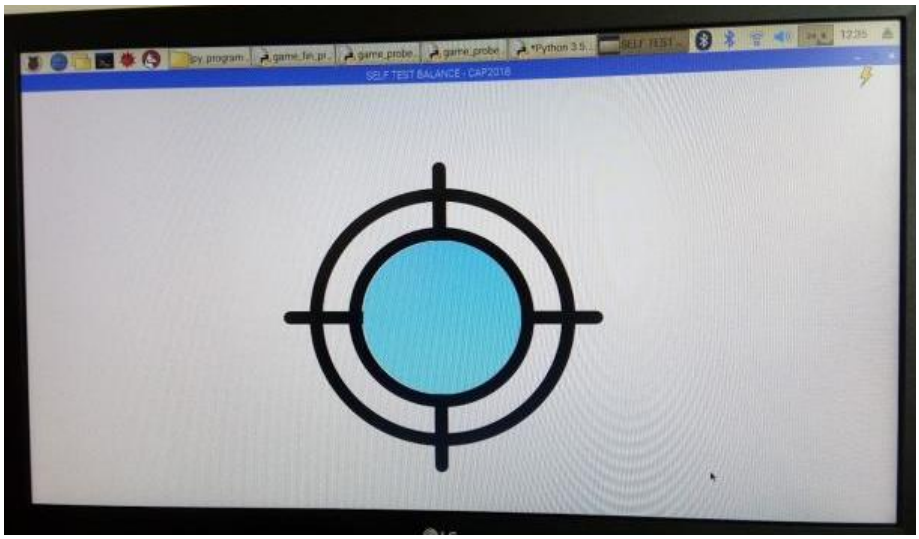


Figura A.11: Pantalla durante el desarrollo de la prueba

De lo mostrado durante la realización de pruebas, se pudo notar que las mediciones obtenidas por el sensor utilizado luego de aplicar la Fusión de Datos propia de la librería utilizada (RTIMULib) son sumamente precisas. Dicha información no presenta variación significativa tal que afecte el desempeño de la persona evaluada durante el desarrollo de la prueba.

Sin embargo, luego de llevar a cabo las pruebas se observó un problema crítico presente en este prototipo. Dicho problema se relaciona con la cantidad de datos que se podían almacenar durante la duración de la prueba. Según los requerimientos del proyecto, la frecuencia de muestreo de datos del sensor debía ocurrir cada 60Hz. Esto implicaba que cada 60s se debían tener almacenados

3600 datos correspondientes al desempeño del usuario durante la prueba. Por el contrario, se notó que la cantidad de datos que se encontraban almacenados estaban alrededor de 3000 a 3200 muestras, lo cual significaba que el sistema perdía una cantidad de datos considerable para el desarrollo de la investigación pertinente.

Se analizaron las causas del problema, y se encontró que la librería "pygame" utiliza interrupciones internas para poder utilizar las funciones que permitían desplegar la interfaz gráfica de la prueba en la pantalla. Dado ese caso, la Raspberry Pi no podía realizar alguna otra acción hasta concluir la operación pertinente de dicha interrupción. En ese sentido, si la instrucción de lectura de datos se recibía en dicho instante, dicha solicitud era ignorada, por lo cual se manifestaba la pérdida de datos importantes para el análisis. En consecuencia, en vista de que ambas acciones no se ejecutaban en paralelo, ocurría la perdida de datos.

Por otra parte, la ventaja presentada por este modelo de solución se evidenció en la resolución de la interfaz gráfica desplegada en pantalla. Dicha resolución mostrada permitía contar con un entorno visualmente amigable tal que el usuario pueda realizar la prueba de forma óptima.

Finalmente, con la finalidad de subsanar las fallas presentadas por este módulo y a la vez aprovechar sus ventajas, se ideó la Alternativa de Solución Final, en la cual está basada el desarrollo de la presente tesis.

B) DIAGRAMAS DE FLUJO:

B.1) Módulo de Interfaz Gráfica

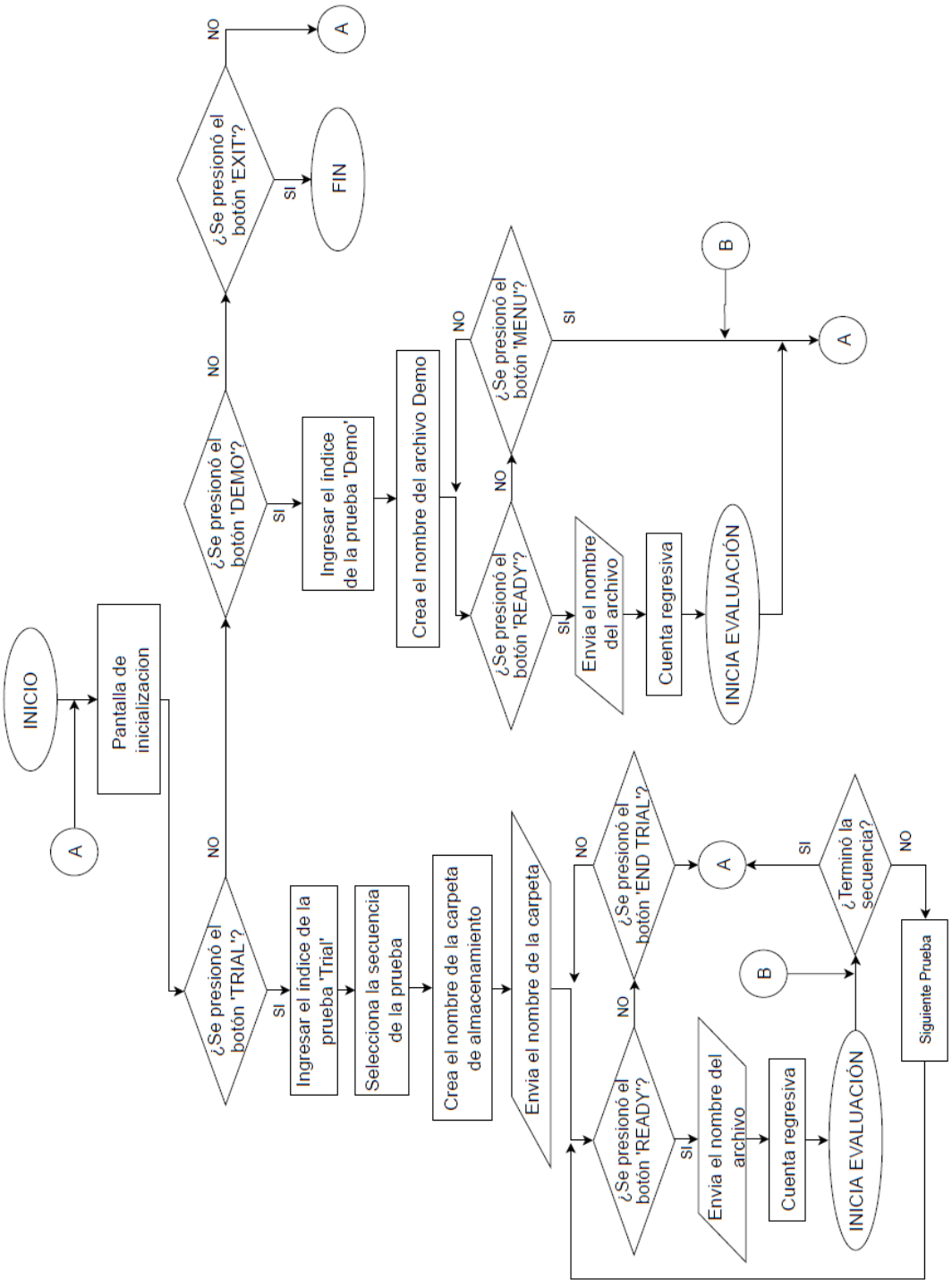


Figura B.1: Diagrama de Flujo de inicialización de la prueba por medio de la interfaz gráfica

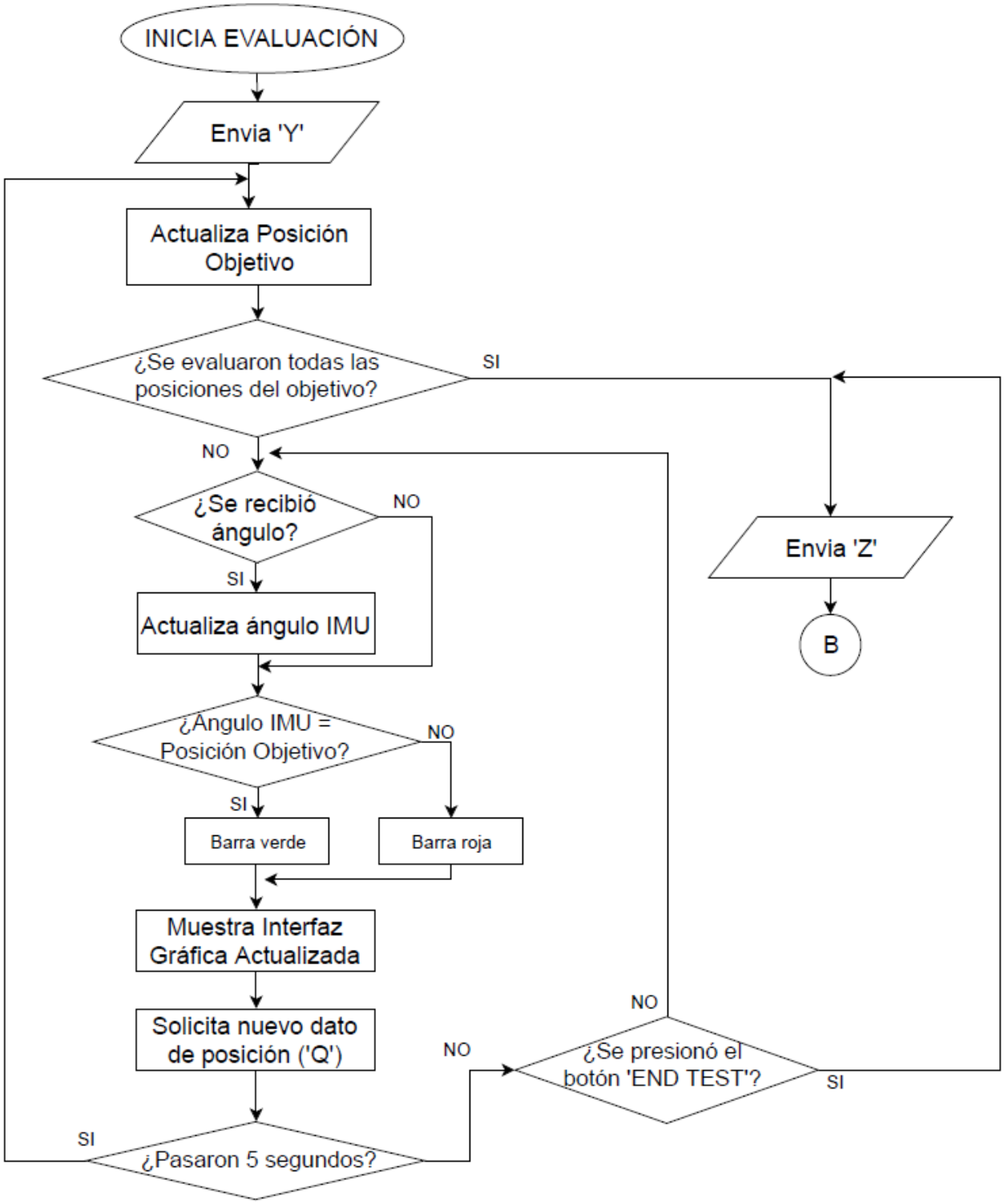


Figura B.2: Diagrama de Flujo de la Interfaz Gráfica durante el desarrollo de la prueba

B.2) Módulo de Adquisición y Almacenamiento

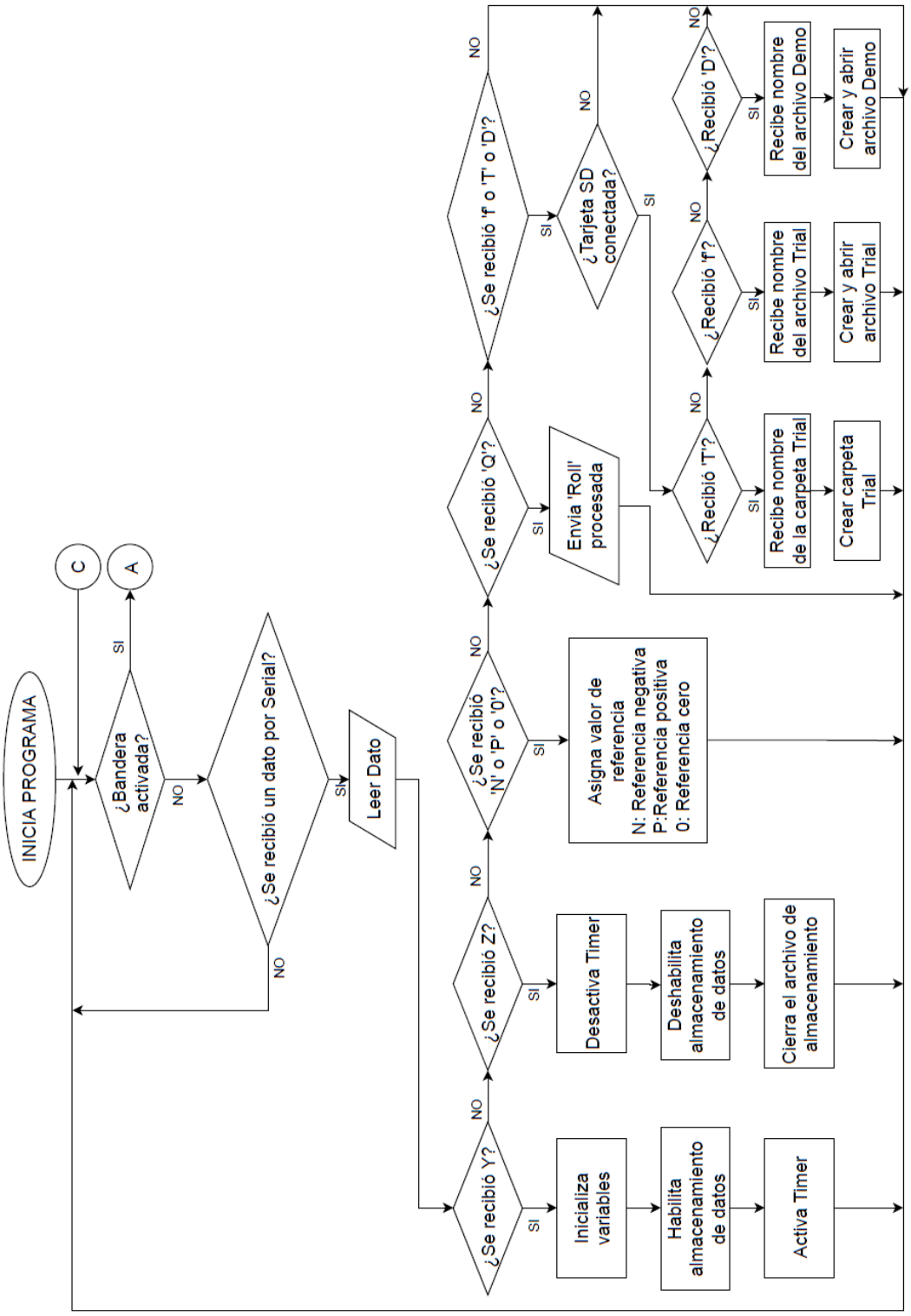


Figura B.3: Diagrama de Flujo Interacción del Módulo de Adquisición con el Módulo de Interfaz Gráfica

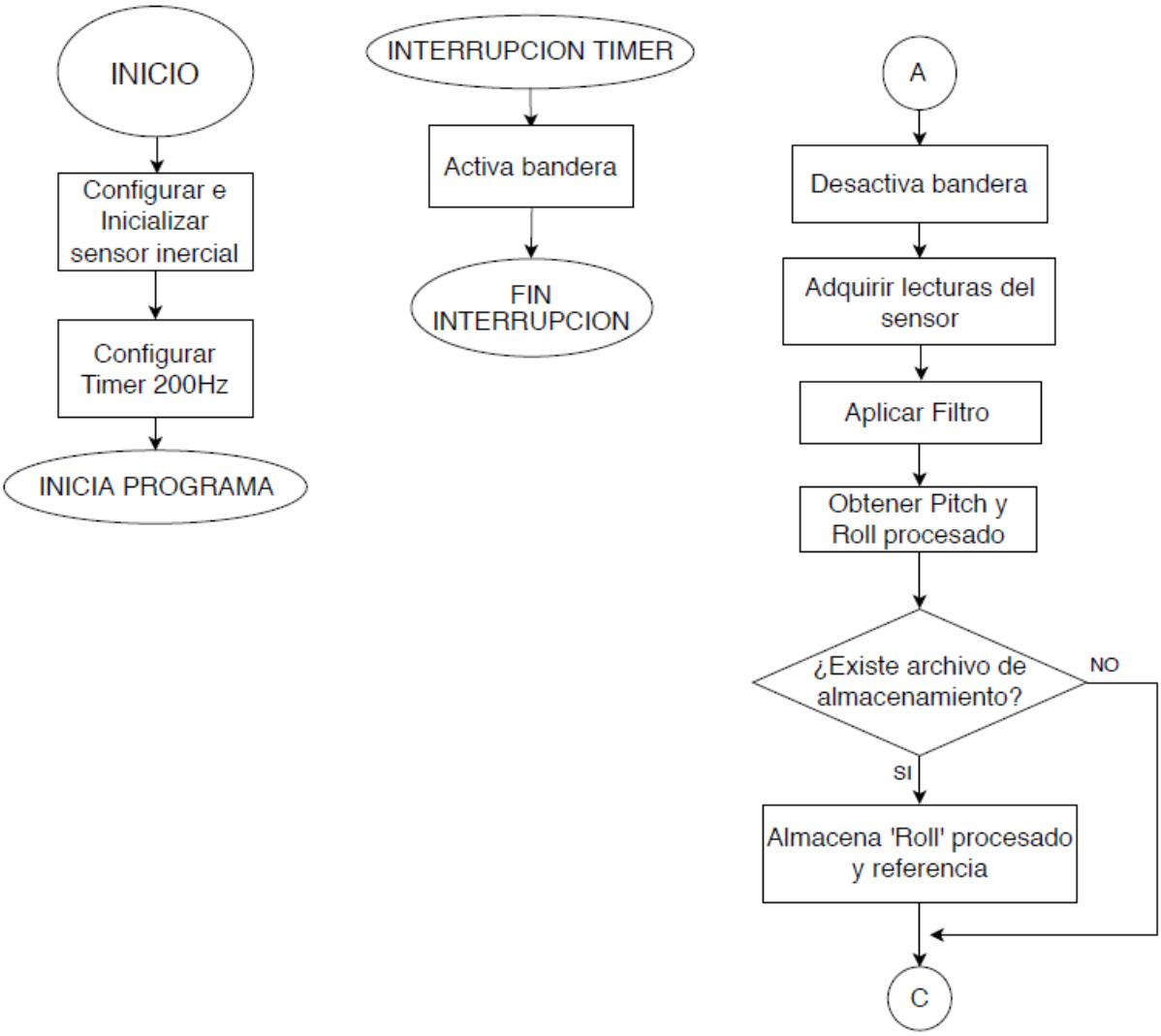


Figura B.4: Diagrama de Flujo Adquisición de Datos del sensor, Procesamiento y Almacenamiento

C) CÓDIGOS DE PROGRAMA UTILIZADOS

C.1) Unidad de Adquisición y Almacenamiento

1) Filtro Complementario

```
/************************************************************************
*********************************
 *
 * Título: Diseño e Implementación de un Sistema de Adquisición para la
Evaluación del Balance Corporal
 * Tesista: Juan Marcos Luna Jaramillo
 * Asesora: MsC. Rocio Callupe Pérez
 * Co-Asesor: Dr. Damián Sal y Rosas Celi
 *
 * Código: Módulo de Adquisición y Almacenamiento aplicando Filtro
Complementario
 *

************************************************************************
*********************************/

#include <SPI.h>
#include <SD.h>
#include <avr/pgmspace.h>

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

//**********Declaramos archivo SD
File myFile;


//**********************

//**********Definimos constantes de IMU MPU6050
#define MPU6050_ADDRESS_AD0_LOW     0x68 // address pin low (GND),
default for InvenSense evaluation board
#define MPU6050_ADDRESS_AD0_HIGH    0x69 // address pin high (VCC)

#define MPU6050_ADDRESS  MPU6050_ADDRESS_AD0_HIGH//Modelo de dispositivo
usado
#define MPU6050_R_PWR_MGMT_1        0x6B
#define MPU6050_PWR1_CLKSEL_BIT     2
#define MPU6050_PWR1_CLKSEL_LENGTH  3
#define MPU6050_CLOCK_PLL_XGYRO     0x01
#define MPU6050_PWR1_SLEEP_BIT      6
#define MPU6050_PWR1_SLEEP_LENGTH   1

#define GYRO_FULL_SCALE_250_DPS   0x00
#define GYRO_FULL_SCALE_500_DPS   0x08
#define GYRO_FULL_SCALE_1000_DPS  0x10
#define GYRO_FULL_SCALE_2000_DPS  0x18

#define ACCEL_FULL_SCALE_2_G    0x00
#define ACCEL_FULL_SCALE_4_G    0x08
#define ACCEL_FULL_SCALE_8_G    0x10
#define ACCEL_FULL_SCALE_16_G   0x18
```

```
//***********************

//****Defición pines
#define SDPIN       9
#define rotopod_pin 3
#define IntPin 4
#define OscPin 10

//****************Variables para etiquetar los archivos de texto
String nameTrialDirec = "";
String nameTrialFile = "";
String nameDemoFile = "";


//****************Variables usadas para las mediciones
char cad[30];
int16_t ax, ay, az;
int16_t gx, gy, gz;
double roll;
double pitch;
int reference;
const float fact_g = 32768 / 2000.0;
uint32_t timer;
uint8_t Buf[14];
int digit = 0;

float pitchAcc, rollAcc;

//****************Variables usadas por el programa
int posicion = 0;
long indexdata = 0;
bool sendimu = false;


//******************* Funciones para comunicación I2C
************************//

//Esta funcion lee Nbytes del dispositivo I2C direccionado (ADDRESS)
void I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t*
Data) {
  //Inicia la direccion del registro
  Wire.beginTransmission(Address);
  Wire.write(Register);
  Wire.endTransmission();

  //Leer Nbytes
  Wire.beginTransmission(Address);
  Wire.requestFrom(Address, Nbytes);
  uint8_t index = 0;
  while (Wire.available()) {
    Data[index++] = Wire.read();
  }
  Wire.endTransmission();
}

//Escribe un byte(Data) en el registro(Register) del dispositivo
direccionado(Address)
void I2Cwrite(uint8_t Address, uint8_t Register, uint8_t Data) {
  //Iniciar la direccion del registro
  Wire.beginTransmission(Address);
  Wire.write(Register);
```

```
  Wire.write(Data);
  Wire.endTransmission();
}

//Escribe bits especificos en direcciones especificas de un registro
void I2CwriteBits(uint8_t Address, uint8_t Register, uint8_t bitStart,
uint8_t leng, uint8_t Data) {
  uint8_t data_send;
  I2Cread(Address, Register, 1, &data_send);
  if (data_send != 0) {
    uint8_t mask = ((1 << leng) - 1) << (bitStart - leng + 1); //Crea la
mascara
    Data <<= (bitStart - leng + 1); //Traslada el dato a la posicion
correcta
    Data &= mask;//Escribe cero en todos los bits no importantes
    data_send &= ~mask;//Escribe cero en los bits importantes
    data_send |= Data;//Combina el dato con el byte existente
    I2Cwrite(Address, Register, data_send);
  }
}

//**************** Función para configurar el Timer
**********************//
void ConfigureTimer() {

  //Constantes para las interrupciones
  const uint16_t t1_load = 0;
  const uint16_t t1_comp = 9999; //Para 60Hz(33333) con pre-escalador de
8//9999(200Hz)//2666(750Hz)//4199(476Hz)

  //Reset Timer1 Control Reg A
  TCCR1A = 0;

  //Set CTC mode(reset the count of the Timer)
  TCCR1B &= ~(1 << WGM13);
  TCCR1B |= (1 << WGM12);
  TCCR1B &= ~(1 << WGM11);
  TCCR1B &= ~(1 << WGM10);

  //Set the prescaler of 8
  TCCR1B &= ~(1 << CS12);
  TCCR1B |= (1 << CS11);
  TCCR1B &= ~(1 << CS10);

  //Reset Timer1 and set compare value
  TCNT1 = t1_load;
  OCR1A = t1_comp;

  //Enable global interrupts
  sei();//interrupts()
}

volatile bool intFlag = false;
//Funcion de interrupcion cada 60Hz para la lectura de datos
ISR(TIMER1_COMPA_vect) {
  intFlag = true;
}

//Interrupción rotopod
void ROTOPOD() {
  digit = 1;
```

```
}


//*************** Función para configurar los parámetros del IMU
***************//
void ConfigureIMU() {
  I2Cwrite(MPU6050_ADDRESS, 25, 7);

  //Inicializa el reloj(PLL with X-axis gyroscope reference)
  I2CwriteBits(MPU6050_ADDRESS, MPU6050_R_PWR_MGMT_1,
MPU6050_PWR1_CLKSEL_BIT, MPU6050_PWR1_CLKSEL_LENGTH,
MPU6050_CLOCK_PLL_XGYRO);

  //Configuracion de los filtros a 184Hz
  //Para el acelerometro y giroscopio(delay 19ms y 18.6ms)
  I2Cwrite(MPU6050_ADDRESS, 26, 0x01);

  //Configuracion rango del aceleromtero
  I2Cwrite(MPU6050_ADDRESS, 28, ACCEL_FULL_SCALE_2_G);

  //Configura rango del giroscopio
  I2Cwrite(MPU6050_ADDRESS, 27, GYRO_FULL_SCALE_2000_DPS);

  //Configura en modo sleep
  I2CwriteBits(MPU6050_ADDRESS, MPU6050_R_PWR_MGMT_1,
MPU6050_PWR1_SLEEP_BIT, MPU6050_PWR1_SLEEP_LENGTH, 0);
}


void setup() {
  // put your setup code here, to run once:
  Wire.begin();
  Wire.setClock(400000);
  Serial.begin(115200);//115200
  //Start SD library
  SD.begin(SDPIN);
  delay(10);

  //Inicializa el IMU
  ConfigureIMU();
  delay(1000); // Wait for sensor to stabilize

  //Configurando interrupcion de lectura cada 60Hz
  ConfigureTimer();

  pinMode(OscPin, OUTPUT);
  timer = micros();
}

void loop() {
  // put your main code here, to run repeatedly:
  uint8_t inicio;

  while (!intFlag) {
    if (Serial.available()) {
      while (Serial.available() > 0) {
        delay(1);
        cad[posicion] = Serial.read();
        posicion++;
      }
      posicion = 0;
```

```
      //Serial.println(cad);

      if (cad[0] == 'Y') {//E:enable reading
        reference = 0;
        indexdata = 0;
        sendimu = true;
        TIMSK1 |= (1 << OCIE1A);
      }
      else if (cad[0] == 'Z') {//D:disable reading
        TIMSK1 &= ~(1 << OCIE1A);
        sendimu = false;

        myFile.println(indexdata);
        myFile.close();
        indexdata = 0;
      }
      else if (cad[0] == 'Q') {
        Serial.println(roll, 4);
      }
      else if ((cad[0] == 'N') || (cad[0] == 'P') || (cad[0] == '0')) {
//I:imu
        if (cad[0] == 'N') {
          reference = (int(cad[1]) - 48) * (-1);
        }
        else if (cad[0] == 'P') {
          reference = (int(cad[1]) - 48);
        }
        else if (cad[0] == '0') {
          reference = 0;
        }
      }
      else if ((cad[0] == 'f') || (cad[0] == 'D') || (cad[0] == 'T')) {
        inicio = SD.begin(SDPIN);
        delay(10);

        if (inicio) {
          if (cad[0] == 'T') {
            for (int i = 0; cad[i] != '\0'; i++) {
              nameTrialDirec.concat(cad[i]);
            }
            SD.mkdir(nameTrialDirec);
          }
          else if (cad[0] == 'D') {
            for (int i = 0; cad[i] != '\0'; i++) {
              nameDemoFile.concat(cad[i]);
            }
            myFile = SD.open(nameDemoFile, FILE_WRITE);
          }
          else if (cad[0] == 'f') {
            for (int i = 1; cad[i] != '\0'; i++) {
              nameTrialFile.concat(cad[i]);
            }
            myFile = SD.open(nameTrialFile, FILE_WRITE);
          }
        }
        nameTrialDirec = "";
        nameTrialFile = "";
        nameDemoFile = "";
      }
      else {
        Serial.end();
```

```
        Serial.begin(115200);
      }

      for (int i = 0; i < sizeof(cad); i++) {
        cad[i] = '\0';
      }
    }

  }

  intFlag = false;

  //********Procesamiento del acelerometro y giroscopio
  //Lectura de aceleromtero y giroscopio
  uint8_t Buf[14];
  I2Cread(MPU6050_ADDRESS, 0x3B, 14, Buf);

  //Crea los valores de 16 bits a partir de los de 8 bits
  //Para el acelerometro
  ax = -(Buf[0] << 8 | Buf[1]);
  ay = -(Buf[2] << 8 | Buf[3]);
  az = (Buf[4] << 8 | Buf[5]);

  //Para el giroscopio
  gx = -(Buf[8] << 8 | Buf[9]);
  gy = -(Buf[10] << 8 | Buf[11]);
  gz = (Buf[12] << 8 | Buf[13]);

  double dt = (double)(micros() - timer) / 1000000; // Calculate delta
time
  timer = micros();

  // Integrate the gyroscope data -> int(angularSpeed) = angle
  roll += ((float)gz / fact_g) * dt; // Angle around the X-axis
  pitch -= ((float)gy / fact_g) * dt;    // Angle around the Y-axis

  int forceMagnitudeApprox = abs(ax) + abs(ay) + abs(az);
  if (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768)
  {
    // Turning around the X axis results in a vector on the Y-axis
    rollAcc = atan2f((float)ay, (float)az) * 180 / M_PI;
    roll = roll * 0.98 + rollAcc * 0.02;

    // Turning around the Y axis results in a vector on the X-axis
    pitchAcc = atan2f((float)ax, (float)az) * 180 / M_PI;
    pitch = pitch * 0.98 + pitchAcc * 0.02;
  }

  if (myFile) {
    indexdata++;
    myFile.print(roll);
    myFile.print(",");
    myFile.print(reference);
    myFile.print(",");
    myFile.println(digit);
    digit = 0;
  }
}
```

ET LUX IN TENEBRIS LUCET
MCMXVII

2) Filtro Kalman

```
/*************************************************************************
**********************************
 *
 * Título: Diseño e Implementación de un Sistema de Adquisición para la
Evaluación del Balance Corporal
 * Tesista: Juan Marcos Luna Jaramillo
 * Asesora: MsC. Rocio Callupe Pérez
 * Co-Asesor: Dr. Damián Sal y Rosas Celi
 *
 * Código: Módulo de Adquisición y Almacenamiento aplicando Filtro Kalman
 *

*************************************************************************
*********************************/

#include <SPI.h>
#include <SD.h>
#include <Kalman.h> // Source:
https://github.com/TKJElectronics/KalmanFilter
#include <avr/pgmspace.h>

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

//**********Definicion Filtro Kalman usado
#define RESTRICT_PITCH // Comment out to restrict roll to ±90deg instead
- please read:
http://www.freescale.com/files/sensors/doc/app_note/AN3461.pdf

Kalman kalmanX; // Create the Kalman instances
Kalman kalmanY;


//**********Declaramos archivo SD
File myFile;


//**********************

//**********Definimos constantes de IMU MPU6050
#define MPU6050_ADDRESS_AD0_LOW     0x68 // address pin low (GND),
default for InvenSense evaluation board
#define MPU6050_ADDRESS_AD0_HIGH    0x69 // address pin high (VCC)

#define MPU6050_ADDRESS  MPU6050_ADDRESS_AD0_HIGH//Modelo de dispositivo
usado
#define MPU6050_R_PWR_MGMT_1        0x6B
#define MPU6050_PWR1_CLKSEL_BIT     2
#define MPU6050_PWR1_CLKSEL_LENGTH  3
#define MPU6050_CLOCK_PLL_XGYRO     0x01
#define MPU6050_PWR1_SLEEP_BIT      6
#define MPU6050_PWR1_SLEEP_LENGTH   1

#define GYRO_FULL_SCALE_250_DPS   0x00
#define GYRO_FULL_SCALE_500_DPS   0x08
#define GYRO_FULL_SCALE_1000_DPS  0x10
#define GYRO_FULL_SCALE_2000_DPS  0x18
```

```
#define ACCEL_FULL_SCALE_2_G    0x00
#define ACCEL_FULL_SCALE_4_G    0x08
#define ACCEL_FULL_SCALE_8_G    0x10
#define ACCEL_FULL_SCALE_16_G   0x18
//***********************

//****Defincion pines
#define SDPIN       4
#define rotopod_pin 10
#define IntPin 9
#define OscPin 3

//****************Variables para etiquetar los archivos de texto
String nameTrialDirec = "";
String nameTrialFile = "";
String nameDemoFile = "";


//****************Variables usadas para las mediciones
char cad[30];
double ax, ay, az;
double gx, gy, gz;
double roll;
double pitch;
double Kalmanroll, Kalmanpitch;
int reference;
const float fact_g = 32768 / 2000.0;
uint32_t timer;
uint8_t Buf[14];
char digit = '0';
uint8_t cant = 1;


//****************Variables para el filtro Kalman
double gxangle, gyangle; // Angle calculate using the gyro only
double kalAngleX, kalAngleY; // Calculated angle using a Kalman filter

long tiempo_prev, dt;

//****************Variables usadas por el programa
int posicion = 0;
long indexdata = 0;
bool sendimu = false;


//******************* Funciones para comunicación I2C
************************//

//Esta funcion lee Nbytes del dispositivo I2C direccionado (ADDRESS)
void I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t*
Data) {
  //Inicia la direccion del registro
  Wire.beginTransmission(Address);
  Wire.write(Register);
  Wire.endTransmission();

  //Leer Nbytes
  Wire.beginTransmission(Address);
  Wire.requestFrom(Address, Nbytes);
  uint8_t index = 0;
```

```
  while (Wire.available()) {
    Data[index++] = Wire.read();
  }
  Wire.endTransmission();
}

//Escribe un byte(Data) en el registro(Register) del dispositivo
direccionado(Address)
void I2Cwrite(uint8_t Address, uint8_t Register, uint8_t Data) {
  //Iniciar la direccion del registro
  Wire.beginTransmission(Address);
  Wire.write(Register);
  Wire.write(Data);
  Wire.endTransmission();
}

//Escribe bits especificos en direcciones especificas de un registro
void I2CwriteBits(uint8_t Address, uint8_t Register, uint8_t bitStart,
uint8_t leng, uint8_t Data) {
  uint8_t data_send;
  I2Cread(Address, Register, 1, &data_send);
  if (data_send != 0) {
    uint8_t mask = ((1 << leng) - 1) << (bitStart - leng + 1); //Crea la
mascara
    Data <<= (bitStart - leng + 1); //Traslada el dato a la posicion
correcta
    Data &= mask;//Escribe cero en todos los bits no importantes
    data_send &= ~mask;//Escribe cero en los bits importantes
    data_send |= Data;//Combina el dato con el byte existente
    I2Cwrite(Address, Register, data_send);
  }
}

//**************** Función para configurar el Timer
**********************//
void ConfigureTimer() {

  //Constantes para las interrupciones
  const uint16_t t1_load = 0;
  const uint16_t t1_comp = 9999; //Para 60Hz(33333) con pre-escalador de
8//9999//5599(357.1Hz)

  //Reset Timer1 Control Reg A
  TCCR1A = 0;

  //Set CTC mode(reset the count of the Timer)
  TCCR1B &= ~(1 << WGM13);
  TCCR1B |= (1 << WGM12);
  TCCR1B &= ~(1 << WGM11);
  TCCR1B &= ~(1 << WGM10);

  //Set the prescaler of 8
  TCCR1B &= ~(1 << CS12);
  TCCR1B |= (1 << CS11);
  TCCR1B &= ~(1 << CS10);

  //Reset Timer1 and set compare value
  TCNT1 = t1_load;
  OCR1A = t1_comp;

  //Enable global interrupts
```

```
  sei();//interrupts()
}

volatile bool intFlag = false;
//Funcion de interrupcion cada 60Hz para la lectura de datos
ISR(TIMER1_COMPA_vect) {
  intFlag = true;
}

//Interrupción rotopod
void ROTOPOD() {
  digit = 'S';
}


//*************** Función para configurar los parámetros del IMU
***************//
void ConfigureIMU() {
  I2Cwrite(MPU6050_ADDRESS, 25, 7);

  //Inicializa el reloj(PLL with X-axis gyroscope reference)
  I2CwriteBits(MPU6050_ADDRESS, MPU6050_R_PWR_MGMT_1,
MPU6050_PWR1_CLKSEL_BIT, MPU6050_PWR1_CLKSEL_LENGTH,
MPU6050_CLOCK_PLL_XGYRO);

  //Configuracion de los filtros a 5Hz
  //Para el acelerometro y giroscopio(delay 19ms y 18.6ms)
  I2Cwrite(MPU6050_ADDRESS, 26, 0x01);

  //Configuracion rango del aceleromtero
  I2Cwrite(MPU6050_ADDRESS, 28, ACCEL_FULL_SCALE_2_G);

  //Configura rango del giroscopio
  I2Cwrite(MPU6050_ADDRESS, 27, GYRO_FULL_SCALE_2000_DPS);

  //Configura en modo sleep
  I2CwriteBits(MPU6050_ADDRESS, MPU6050_R_PWR_MGMT_1,
MPU6050_PWR1_SLEEP_BIT, MPU6050_PWR1_SLEEP_LENGTH, 0);
}


void setup() {
  // put your setup code here, to run once:
  Wire.begin();
  Wire.setClock(400000);
  Serial.begin(115200);//115200
  //Start SD library
  SD.begin(SDPIN);
  delay(10);
  pinMode(OscPin, OUTPUT);

  pinMode(rotopod_pin, INPUT);
  attachInterrupt(digitalPinToInterrupt(rotopod_pin), ROTOPOD, RISING);
  pinMode(IntPin, OUTPUT);

  //Inicializa el IMU
  ConfigureIMU();
  delay(1000); // Wait for sensor to stabilize

  //Datos iniciales para el filtro
  Buf[14];
```

```
  I2Cread(MPU6050_ADDRESS, 0x3B, 14, Buf);
  ax = -(Buf[0] << 8 | Buf[1]);
  ay = -(Buf[2] << 8 | Buf[3]);
  az = (Buf[4] << 8 | Buf[5]);

  // It is then converted from radians to degrees
#ifdef RESTRICT_PITCH // Eq. 25 and 26
  roll  = atan2(ax, az) * RAD_TO_DEG;
  pitch = atan(-ax / sqrt(ay * ay + az * az)) * RAD_TO_DEG;
#else // Eq. 28 and 29
  double roll  = atan(ay / sqrt(ax * ax + az * az)) * RAD_TO_DEG;
  double pitch = atan2(-ax, az) * RAD_TO_DEG;
#endif

  kalmanX.setAngle(roll); // Set starting angle
  kalmanY.setAngle(pitch);
  gxangle = roll;
  gyangle = pitch;

  //Configurando interrupcion de lectura cada 60Hz
  ConfigureTimer();

  timer = micros();
}

void loop() {
  // put your main code here, to run repeatedly:
  uint8_t inicio;

  while (!intFlag) {
    if (Serial.available()) {

      while (Serial.available() > 0) {
        delay(1);
        cad[posicion] = Serial.read();
        posicion++;
      }
      posicion = 0;

      if (cad[0] == 'Y') {//E:enable reading
        reference = 0;
        indexdata = 0;
        TIMSK1 |= (1 << OCIE1A);
      }
      else if (cad[0] == 'Z') {//D:disable reading
        TIMSK1 &= ~(1 << OCIE1A);

        myFile.println(indexdata);
        myFile.close();
        indexdata = 0;
      }
      else if (cad[0] == 'Q') {
        Serial.println(Kalmanroll, 4);
      }
      else if ((cad[0] == 'N') || (cad[0] == 'P') || (cad[0] == '0')) {
//I:imu
        if (cad[0] == 'N') {
          reference = (int(cad[1]) - 48) * (-1);
        }
        else if (cad[0] == 'P') {
          reference = (int(cad[1]) - 48);
```

```
        }
        else if (cad[0] == '0') {
          reference = 0;
        }
      }
      else if ((cad[0] == 'f') || (cad[0] == 'D') || (cad[0] == 'T')) {
        inicio = SD.begin(SDPIN);
        delay(10);

        if (inicio) {
          //Serial.println("hi");
          if (cad[0] == 'T') {
            for (int i = 0; cad[i] != '\0'; i++) {
              nameTrialDirec.concat(cad[i]);
            }
            SD.mkdir(nameTrialDirec);
          }
          else if (cad[0] == 'D') {
            for (int i = 0; cad[i] != '\0'; i++) {
              nameDemoFile.concat(cad[i]);
            }
            myFile = SD.open(nameDemoFile, FILE_WRITE);
          }
          else if (cad[0] == 'f') {
            for (int i = 1; cad[i] != '\0'; i++) {
              nameTrialFile.concat(cad[i]);
            }
            myFile = SD.open(nameTrialFile, FILE_WRITE);
          }
        }
        nameTrialDirec = "";
        nameTrialFile = "";
        nameDemoFile = "";
      }
      else {
        Serial.end();
        Serial.begin(115200);
      }

      for (int i = 0; i < sizeof(cad); i++) {
        cad[i] = '\0';
      }
    }
  }

  intFlag = false;

  //********Procesamiento del acelerometro y giroscopio
  //Lectura de aceleromtero y giroscopio
  uint8_t Buf[14];
  I2Cread(MPU6050_ADDRESS, 0x3B, 14, Buf);

  //Crea los valores de 16 bits a partir de los de 8 bits
  //Para el acelerometro
  int16_t ax = -(Buf[0] << 8 | Buf[1]);
  int16_t ay = -(Buf[2] << 8 | Buf[3]);
  int16_t az = (Buf[4] << 8 | Buf[5]);

  //Para el giroscopio
  int16_t gx = -(Buf[8] << 8 | Buf[9]);
  int16_t gy = -(Buf[10] << 8 | Buf[11]);
```

```
  int16_t gz = (Buf[12] << 8 | Buf[13]);

  //digitalWrite(OscPin,HIGH);
  double dt = (double)(micros() - timer) / 1000000; // Calculate delta
time
  timer = micros();

  double roll  = atan2(ay, az) * RAD_TO_DEG;
  double pitch = atan(-ax / sqrt(ay * ay + az * az)) * RAD_TO_DEG;

  double gxrate = gx / fact_g; // Convert to deg/s
  double gyrate = gy / fact_g; // Convert to deg/s

  // This fixes the transition problem when the accelerometer angle jumps
between -180 and 180 degrees
  if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
    kalmanX.setAngle(roll);
    kalAngleX = roll;
    gxangle = roll;
  } else
    kalAngleX = kalmanX.getAngle(roll, gxrate, dt); // Calculate the
angle using a Kalman filter

  if (abs(kalAngleX) > 90)
    gyrate = -gyrate; // Invert rate, so it fits the restriced
accelerometer reading

  kalAngleY = kalmanY.getAngle(pitch, gyrate, dt);

  gxangle += gxrate * dt; // Calculate gyro angle without any filter
  gyangle += gyrate * dt;

  // Reset the gyro angle when it has drifted too much
  if (gxangle < -180 || gxangle > 180)
    gxangle = kalAngleX;
  if (gyangle < -180 || gyangle > 180)
    gyangle = kalAngleY;

  Kalmanroll = kalAngleX;
  Kalmanpitch = kalAngleY;

  if (myFile) {
    indexdata++;
    myFile.print(Kalmanroll, 3);
    myFile.print("\t");
    myFile.print(reference);
    myFile.print("\t");
    myFile.println(digit);
    digit = '0';
  }
}
```

ET LUX IN TENEBRIS LUCET
MCMXVII

C.2) Módulo de Interfaz Gráfica

```
#*****************************************************************************************
******************
#
# Título: Diseño e Implementación de un Sistema de Adquisición para la Evaluación del
Balance Corporal
# Tesista: Juan Marcos Luna Jaramillo
# Asesora: Rocio Callupe Pérez
# Co-Asesor: Damián Sal y Rosas Celi
#
# Código: Módulo de Adquisición y Almacenamiento aplicando Filtro Kalman
#
#*****************************************************************************************
*****************

import pygame, sys, math
from datetime import datetime
import time
import random
import serial
import tkinter as tk
import RPi.GPIO as GPIO
from pygame.locals import *
from pygame_functions import *

root = tk.Tk()
root.withdraw()

WINWIDTH = root.winfo_screenwidth()
WINHEIGTH = root.winfo_screenheight()

FPS=60
HALF_WINWIDTH = int(WINWIDTH/2)
HALF_WINHEIGTH = int(WINHEIGTH/2)


BLACK =          (  0,   0,   0)
WHITE =          (255, 255, 255)
BRIGHTBLUE =     (  0,  50, 255)
RED=             (200,   0,   0)
RED_BRIGTH =     (255,   0,   0)
DARKTURQUOISE =  (  3,  54,  73)
GREEN =          (  0, 204,   0)
GRAY =           (128, 128, 128)
LIGTHGRAY =      (211, 211, 211)
BLUEGRAY =       (119,136,153)
DIMGRAY=         (105,105,105)
BGCOLOR=DARKTURQUOISE
TEXTCOLOR=WHITE
BUTTONCOLOR_OFF=DIMGRAY
BUTTONCOLOR_ON=GRAY
TEXTBUTTON=WHITE

initialize = 0
GAME=0

nameDemoFile=""
nameTrialDirec=""
nameTrialFile=""
```

```
myangles=[]
TRIAL_SURFS=[]
TRIAL_RECTS=[]
DEMO_SURFS=[]
DEMO_RECTS=[]
SEQ_SURFS=[]
SEQ_RECTS=[]


SEC1 = [0,3,0,-6,0,-3,0,6,0,-3,0,-6,0,3,0,6]
SEC2 = [0,6,0,-3,0,3,0,-6,0,6,0,-3,0,-6,0,3]
SEC3 = [0,-3,0,-6,0,6,0,-3,0,3,0,-6,0,6,0,3]
SEC4 = [0,3,0,6,0,-6,0,-3,0,3,0,6,0,-6,0,-3]

arduino=serial.Serial('/dev/ttyUSB0',baudrate=115200)#ACM0230400

if (arduino.isOpen()==False):
    arduino.open()
time.sleep(0.1)

def main():
    global
myangles,FPSCLOCK,DISPLAYSURF,inst,GAME,BASICFONT,ANGLES,INSTRUCTIONS_DEMO,INSTRUCTIONS_T
RIAL
    global trial,demo,numtest
    global TRIAL_SURFS, TRIAL_RECTS, DEMO_SURFS,DEMO_RECTS,SEQ_SURFS,SEQ_RECTS
    global SECA,SECB,SECC,SECD
    global PRUEBA_A,PRUEBA_B,PRUEBA_C,PRUEBA_D
    global mouse,click
    global sec
    global nameDemoFile,nameTrialDirec,nameTrialFile
    global dateStr

    global COUNT,info_count,count_full,READY
    global index_ok

    COUNT=0
    count_full=0
    info_count=0
    READY=0

    index_ok=0

    ##Agrego dos varioables globales : Prueba(T) y Demo(D)
    mouse=0
    click=0
    sec=0
    SECA=0
    SECB=0
    SECC=0
    SECD=0
    numtest=0
    trial=0
    demo=0
    inst=0
    GAME=0

    nameDemoFile=''
    nameTrialDirec=''
    nameTrialFile=''
```

```
    pygame.init()
    pygame.time.set_timer(pygame.USEREVENT,4000)
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINWIDTH,WINHEIGTH))

    pygame.display.set_caption('BALANCE TEST - CAP2018')
    BASICFONT = pygame.font.Font('freesansbold.ttf',30)

    INSTRUCTIONS_TRIAL = ['Welcome to the BALANCE TEST...!',
                      'To start the Evaluation you must choose the sequence you want to
follow: A,B,C,D',
                      'Only press the correpond button on the screen',
                      'How to Play?',
                      'You must follow the target position only moving ',
                      'your trunk using the wearable']

    INSTRUCTIONS_DEMO = ['Welcome to the DEMO BALANCE TEST...!',
                      'When ready you must press the Ready Button to start the DEMO ',
                      'How to Play?',
                      'You must follow the target position only moving ',
                      'your trunk using the wearable']

    TEST_SEQ=[' TEST 1 ',' TEST 2 ',' TEST 3 ',' TEST 4 ',' TEST 5 ',' TEST 6 ',' TEST 7
',' TEST 8',' TEST 9 ',' TEST 10 ',' TEST 11 ',' TEST 12 ',' TEST 13',' TEST 14 ',' TEST
15 ']

    topCoord=80
    for i in range(len(INSTRUCTIONS_TRIAL)):
        instSurf = BASICFONT.render(INSTRUCTIONS_TRIAL[i],1,TEXTCOLOR)
        instRect = instSurf.get_rect()
        instRect.centerx = HALF_WINWIDTH
        instRect.centery = topCoord
        topCoord += 50
        TRIAL_SURFS.append(instSurf)
        TRIAL_RECTS.append(instRect)


    topCoord=80
    for i in range(len(INSTRUCTIONS_DEMO)):
        instSurf = BASICFONT.render(INSTRUCTIONS_DEMO[i],1,TEXTCOLOR)
        instRect = instSurf.get_rect()
        instRect.centerx = HALF_WINWIDTH
        instRect.centery = topCoord
        topCoord += 50
        DEMO_SURFS.append(instSurf)
        DEMO_RECTS.append(instRect)


    TITLESEQFONT = pygame.font.Font('freesansbold.ttf',120)
    topCoord=200
    for i in range(len(TEST_SEQ)):
        instSurf = TITLESEQFONT.render(TEST_SEQ[i],1,TEXTCOLOR)
        instRect = instSurf.get_rect()
        instRect.centerx = HALF_WINWIDTH
        instRect.centery = topCoord
        SEQ_SURFS.append(instSurf)
        SEQ_RECTS.append(instRect)

    PRUEBA_A=[SEC1,SEC2,SEC3,SEC4]
    PRUEBA_B=[SEC3,SEC4,SEC1,SEC2]
```

```
    PRUEBA_C=[SEC2,SEC1,SEC4,SEC3]
    PRUEBA_D=[SEC4,SEC2,SEC1,SEC3]

    while True:
        if inst == 0 and GAME == 0:
            InitializeGame()
        elif inst == 1 and GAME == 0:
            dateObject=datetime.now()
            dateStr=dateObject.strftime("%m%d")
            if trial==1 and sec==0:
                trialScreen()
                numtest=1
            elif trial==1 and sec==1:
                ReadyTestScreen()
            elif demo==1:
                demoScreen()
        elif inst == 1 and GAME == 1:
            PlayGame()


def InitializeGame():
    global inst,trial,demo
    global nameDemoFile,nameTrialDirec,nameTrialFile

    topCoord=0
    TITLEFONT = pygame.font.Font('freesansbold.ttf',120)
    titleText = ' BALANCE TEST'

    #Instruction to Start
    instructionText = 'Press the START Button to start the test'

    #Drawing a blank color to the entire window
    DISPLAYSURF.fill(BGCOLOR)

    #TITLE
    titleSurf, titleRect = TextRect(titleText, TITLEFONT,WHITE)
    titleRect.centerx=HALF_WINWIDTH
    titleRect.centery=HALF_WINHEIGTH-200
    DISPLAYSURF.blit(titleSurf,titleRect)
    checkForQuit()


    mouse=0
    click=0

    trial=button(' TRIAL ', HALF_WINWIDTH-300, HALF_WINHEIGTH-50, 250,90,
BUTTONCOLOR_OFF, BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")
    demo=button(' DEMO ', HALF_WINWIDTH+50, HALF_WINHEIGTH-50, 250,90, BUTTONCOLOR_OFF,
BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")
    exit_button=button(' EXIT ', HALF_WINWIDTH-60, HALF_WINHEIGTH+100, 120,60,
BUTTONCOLOR_OFF, BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")

    if exit_button==1:
        pygame.quit()
        sys.exit()

    if trial==1 or demo==1:
        inst=1
        if trial==1:
            nameTrialDirec='T'
        elif demo==1:
```

```
            nameDemoFile='D'

    pygame.display.update()
    FPSCLOCK.tick(FPS)


def trialScreen():
    global TRIAL_SURFS,TRIAL_RECTS
    global GAME
    global SECA,SECB,SECC,SECD
    global PRUEBA
    global inst,sec
    global nameDemoFile,nameTrialDirec,nameTrialFile
    global dateStr
    global info_count,count_full,COUNT,READY,index_ok
    global code_sec
    global NumSec

    msg='Enter Trial index'

    DISPLAYSURF.fill(BGCOLOR)

    for i in range(len(TRIAL_SURFS)):
        DISPLAYSURF.blit(TRIAL_SURFS[i], TRIAL_RECTS[i])

    mouse=0
    click=0

    NUMBERFONT=pygame.font.Font('freesansbold.ttf',25)

    INSTRUCCION_COUNT = 'Enter the Trial index: '
    countSurf = BASICFONT.render(INSTRUCCION_COUNT,1,TEXTCOLOR)
    countRect = countSurf.get_rect()
    countRect.centerx = HALF_WINWIDTH-200
    countRect.centery = HALF_WINHEIGTH+20

    DISPLAYSURF.blit(countSurf, countRect)

    if index_ok==0:
        index_ok=CountTrialScreen(msg,0)
    elif index_ok==1:

buttontext(str(info_count),HALF_WINWIDTH,HALF_WINHEIGTH,300,40,WHITE,NUMBERFONT,"info")
        code_sec=0

        SECA=button(' SEQUENCE A ', HALF_WINWIDTH-330, HALF_WINHEIGTH+100, 250,60,
BUTTONCOLOR_OFF, BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")
        SECB=button(' SEQUENCE B ', HALF_WINWIDTH+80, HALF_WINHEIGTH+100, 250,60,
BUTTONCOLOR_OFF, BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")
        SECC=button(' SEQUENCE C ', HALF_WINWIDTH-330, HALF_WINHEIGTH+250, 250,60,
BUTTONCOLOR_OFF, BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")
        SECD=button(' SEQUENCE D ', HALF_WINWIDTH+80, HALF_WINHEIGTH+250, 250,60,
BUTTONCOLOR_OFF, BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")


        if SECA==1:
            PRUEBA=PRUEBA_A
            code_sec='A'
            sec=1
            NumSec=['1','2','3','4']
        elif SECB==1:
```

```
            PRUEBA=PRUEBA_B
            code_sec='B'
            sec=1
            NumSec=['3','4','1','2']
        elif SECC==1:
            PRUEBA=PRUEBA_C
            code_sec='C'
            sec=1
            NumSec=['2','1','4','3']
        elif SECD==1:
            PRUEBA=PRUEBA_D
            code_sec='D'
            sec=1
            NumSec=['4','2','1','3']

        if code_sec != 0:
            sec=1
            if info_count>10:
                nameTrialDirec=nameTrialDirec+str(info_count)+code_sec+dateStr+'/'
            else:
               nameTrialDirec=nameTrialDirec+'0'+str(info_count)+code_sec+dateStr+'/'

            print(nameTrialDirec)
            arduino.reset_output_buffer()
            arduino.write(str(nameTrialDirec).encode('ascii'))

    Menu=button(' MENU ', HALF_WINWIDTH+450, HALF_WINHEIGTH+200, 150,60, BUTTONCOLOR_OFF,
BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")

    if Menu==1:
        READY=0
        info_count=0
        count_full=0
        COUNT=0
        index_ok=0
        inst=0

    pygame.display.update()
    FPSCLOCK.tick(15)
    checkForQuit()


def CountTrialScreen(msg,y):
    global nameDemoFile,nameTrialDirec,nameTrialFile
    global info_count,count_full
    global COUNT,READY

    var_ok=0

    BUTTONFONT=pygame.font.Font('freesansbold.ttf',15)
    NUMBERFONT=pygame.font.Font('freesansbold.ttf',25)
    TEXTFONT=pygame.font.Font('freesansbold.ttf',35)

    if COUNT==0:

COUNT=button(msg,HALF_WINWIDTH,HALF_WINHEIGTH+y,300,40,WHITE,WHITE,BLACK,BUTTONFONT,"info
")
    else:
        if count_full==0:

buttontext(str(info_count),HALF_WINWIDTH,HALF_WINHEIGTH+y,300,40,WHITE,NUMBERFONT,"info")
```

```
            c=Keyboard(y)
            if c==11:
                count_full=1
                READY=1
                time.sleep(0.2)
            elif c==10:
                info_count=int(info_count/10)
            elif c!= '-':
                info_count=info_count*10+c
                if info_count>99:
                    info_count=int(info_count/10)
                    buttontext('Range exceeded!',HALF_WINWIDTH-350, HALF_WINHEIGTH-
50,700,100,WHITE,TEXTFONT,"info")
                    pygame.display.update()
                    time.sleep(0.5)
        else:

buttontext(str(info_count),HALF_WINWIDTH,HALF_WINHEIGTH+y,300,40,WHITE,NUMBERFONT,"info")

    if READY==1:
        if count_full==1:
            var_ok=1
        else:
            buttontext('Please assign an index!',HALF_WINWIDTH-350, HALF_WINHEIGTH-
50,700,100,WHITE,TEXTFONT,"info")

    return var_ok


def Keyboard(y):
    var='-'
    topCoordK=630

    BUTTONFONT=pygame.font.Font('freesansbold.ttf',20)
    #80x80

    cero=button('0',HALF_WINWIDTH-240,HALF_WINHEIGTH+y+90,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    uno=button('1',HALF_WINWIDTH-160,HALF_WINHEIGTH+y+90,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    dos=button('2',HALF_WINWIDTH-80,HALF_WINHEIGTH+y+90,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    tres=button('3',HALF_WINWIDTH,HALF_WINHEIGTH+y+90,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    cuatro=button('4',HALF_WINWIDTH+80,HALF_WINHEIGTH+y+90,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    back=button('<-',HALF_WINWIDTH+160,HALF_WINHEIGTH+y+90,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"back")
    cinco=button('5',HALF_WINWIDTH-240,HALF_WINHEIGTH+y+170,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    seis=button('6',HALF_WINWIDTH-160,HALF_WINHEIGTH+y+170,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    siete=button('7',HALF_WINWIDTH-80,HALF_WINHEIGTH+y+170,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    ocho=button('8',HALF_WINWIDTH,HALF_WINHEIGTH+y+170,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    nueve=button('9',HALF_WINWIDTH+80,HALF_WINHEIGTH+y+170,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"number")
    ok=button('Ok',HALF_WINWIDTH+160,HALF_WINHEIGTH+y+170,80,80,BUTTONCOLOR_OFF,
BUTTONCOLOR_ON,WHITE,BUTTONFONT,"complete")
```

```
    if cero==1:
        var=0
        time.sleep(0.15)
    elif uno==1:
        var=1
        time.sleep(0.15)
    elif dos==1:
        var=2
        time.sleep(0.15)
    elif tres==1:
        var=3
        time.sleep(0.15)
    elif cuatro==1:
        var=4
        time.sleep(0.15)
    elif cinco==1:
        var=5
        time.sleep(0.15)
    elif seis==1:
        var=6
        time.sleep(0.15)
    elif siete==1:
        var=7
        time.sleep(0.15)
    elif ocho==1:
        var=8
        time.sleep(0.15)
    elif nueve==1:
        var=9
        time.sleep(0.15)
    elif back==1:
        var=10
        time.sleep(0.15)
    elif ok==1:
        var=11

    return var


def buttontext(msg,x,y,w,h,color,font,action=None):
    pygame.draw.rect(DISPLAYSURF,color, (x,y,w,h))
    textSurf, textRect = TextRect(msg, font,BLACK)
    textRect.center = (x+(w/2),y+(h/2))
    DISPLAYSURF.blit(textSurf,textRect)


def ReadyTestScreen():
    global inst,GAME,sec
    global nameDemoFile,nameTrialDirec,nameTrialFile
    global info_count,count_full,COUNT,READY,index_ok
    global code_sec
    global NumSec
    global nameTrialFile
    global dateStr

    #Drawing a blank color to the entire window
    DISPLAYSURF.fill(BGCOLOR)

    #Instruction to Start
    instructionText = 'Press the START Button to start the test'
    instSurf, instRect = TextRect(instructionText,BASICFONT,WHITE)
```

```
    instRect.centerx=HALF_WINWIDTH
    instRect.centery=HALF_WINHEIGTH-80
    DISPLAYSURF.blit(instSurf,instRect)
    DISPLAYSURF.blit(SEQ_SURFS[numtest-1],SEQ_RECTS[numtest-1])
    index_sec=NumSec[numtest-1]

    GAME=button(' START ', HALF_WINWIDTH-100, HALF_WINHEIGTH, 200,100, BUTTONCOLOR_OFF,
BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")
    ENDTRIAL=button(' TRIAL END ', HALF_WINWIDTH+400, HALF_WINHEIGTH+100, 200,100,
BUTTONCOLOR_OFF, BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")

    if GAME==1:
        if int(index_sec)>10:
            nameTrialFile='f'+nameTrialDirec+'T'+code_sec+index_sec+dateStr+'.txt'
        else:
            nameTrialFile='f'+nameTrialDirec+'T'+code_sec+'0'+index_sec+dateStr+'.txt'
        print(nameTrialFile)
        arduino.reset_output_buffer()
        arduino.write(str(nameTrialFile).encode('ascii'))

    if ENDTRIAL==1:
        inst=0
        GAME=0
        sec=0
        READY=0
        info_count=0
        count_full=0
        index_ok=0
        COUNT=0

    pygame.display.update()
    FPSCLOCK.tick(15)
    checkForQuit()


def demoScreen():
    global DEMO_SURFS,DEMO_RECTS
    global GAME
    global inst
    global nameDemoFile,nameTrialDirec,nameTrialFile
    global dateStr
    global info_count,count_full,COUNT,READY,index_ok

    msg='Enter Demo index'

    NUMBERFONT=pygame.font.Font('freesansbold.ttf',25)

    DISPLAYSURF.fill(BGCOLOR)

    INSTRUCCION_COUNT = 'Enter the Demo index: '
    countSurf = BASICFONT.render(INSTRUCCION_COUNT,1,TEXTCOLOR)
    countRect = countSurf.get_rect()
    countRect.centerx = HALF_WINWIDTH-200
    countRect.centery = HALF_WINHEIGTH+60

    DISPLAYSURF.blit(countSurf, countRect)

    for i in range(len(DEMO_SURFS)):
        DISPLAYSURF.blit(DEMO_SURFS[i], DEMO_RECTS[i])

    if index_ok==0:
```

```
        index_ok=CountTrialScreen(msg,40)
    elif index_ok==1:

buttontext(str(info_count),HALF_WINWIDTH,HALF_WINHEIGTH+40,300,40,WHITE,NUMBERFONT,"info"
)
        GAME=button(' READY ', HALF_WINWIDTH-100, HALF_WINHEIGTH+120, 200,100,
BUTTONCOLOR_OFF, BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")

    if GAME==1:
        if info_count>10:
            nameDemoFile=nameDemoFile+str(info_count)+dateStr+'.txt'
        else:
             nameDemoFile=nameDemoFile+'0'+str(info_count)+dateStr+'.txt'
        print(nameDemoFile)
        arduino.reset_output_buffer()
        arduino.write(str(nameDemoFile).encode('ascii'))
        time.sleep(0.1)

    Menu=button(' MENU', HALF_WINWIDTH+450, HALF_WINHEIGTH+150, 150,60, BUTTONCOLOR_OFF,
BUTTONCOLOR_ON, WHITE, BASICFONT,"instructions")

    if Menu==1:
        READY=0
        info_count=0
        count_full=0
        COUNT=0
        index_ok=0
        inst=0

    pygame.display.update()
    FPSCLOCK.tick(15)
    checkForQuit()


def getDistanceAngle(angle):
    L1 = 90#distancia del punto medio de la pantalla al piso
    L2 = 60#distancia de los gluteos de la persona al piso

    weight_cm=43.6#ancho de la pantalla en cm
    height_cm=23.5#alto de la pantalla en cm
    R = L1-L2
    distx = R*math.sin(angle*(math.pi/180))*(WINWIDTH/weight_cm)
    #disty = R*(1-math.cos(angle*(math.pi/180)))*(WINHEIGTH/height_cm)

    return distx


def PlayGame():
    global inst,GAME,myangles
    global trial,demo,numtest,sec
    global info_count,count_full,COUNT,READY,index_ok
    global nameDemoFile,nameTrialDirec,nameTrialFile

    TESTFONT = pygame.font.Font('freesansbold.ttf',20)

    #Bloques para actualizar secciones de pantalla
    bargoal_rect=pygame.Rect(0,0,WINWIDTH,30)
    object_rect=pygame.Rect(0,250,WINWIDTH,270)
    boton_rect=pygame.Rect(WINWIDTH-250,WINHEIGTH-150,125,50)

    #Inicializa variables
```

```
    posx=0
    posy=0
    neg=-1
    #offset=1.35

    test_bt=0

    color_bar=RED

    despx=0
    despy=0
    angle_imu=10

    time_limit_bar=WINWIDTH

    arduino.reset_input_buffer()

    NUMBERFONT = pygame.font.Font('freesansbold.ttf',200)

    pygame.time.set_timer(pygame.USEREVENT,1000)

    for i in range(3):
        second = 0

        if i == 0:
            number="3"
        elif i== 1:
            number="2"
        else:
            number="1"

        numberSurf = NUMBERFONT.render(number,1,TEXTCOLOR)
        numberRect = numberSurf.get_rect()
        numberRect.centerx = HALF_WINWIDTH
        numberRect.centery = HALF_WINHEIGTH

        DISPLAYSURF.fill(BGCOLOR)
        DISPLAYSURF.blit(numberSurf, numberRect)
        pygame.display.update()
        FPSCLOCK.tick(15)

        while second == 0:
            for event in pygame.event.get():
                if event.type == pygame.USEREVENT:
                    second=1
                elif event.type == pygame.QUIT:
                    terminate()

    #Modificando para actualizar barra cada 10ms
    time_timer=10#ms
    time_target=5000#ms
    count_timer=time_target/time_timer
    rate=(WINWIDTH*time_timer)/time_target
    pygame.time.set_timer(pygame.USEREVENT,time_timer)#time_target(time_timer*count)
    time.sleep(0.001)

    code = 'Y'#Enable interrupts
    arduino.reset_output_buffer()
    arduino.write(str(code).encode('ascii'))

    if demo==1:
```

```
        i=random.randint(1,len(PRUEBA_A))
        print(i)
        ANGLES=PRUEBA_A[i-1]
    elif trial==1:
        ANGLES=PRUEBA[numtest-1]

    ang='0'

    width_target=250
    heigth_target=250
    width_imu=200
    heigth_imu=200

    targetRect=pygame.Rect(0,0,width_target,heigth_target)#130,250
    imuRect=pygame.Rect(0,0,width_imu,heigth_imu)#120,200

    targetRect.centery=HALF_WINHEIGTH
    imuRect.centery=HALF_WINHEIGTH

    color_bg=BLACK

    DISPLAYSURF.fill(color_bg)
    pygame.display.flip()

    for i in range(len(ANGLES)):
        contpos=1

        cont=1
        cinco=0
        read=0
        angle = ANGLES[i]

        if angle == 0:
            targetRect.centerx = HALF_WINWIDTH
        else:
            posx = getDistanceAngle(angle)
            targetRect.centerx = HALF_WINWIDTH+posx

        imuRect.centerx = HALF_WINWIDTH+despx

        if angle == 0:
            code='0'
        elif angle < 0:
            ang=str(angle*(-1))
            code='N'+ang
        elif angle > 0:
            ang=str(angle)
            code='P'+ang


        arduino.reset_output_buffer()
        arduino.write(str(code).encode('ascii'))

        posangle=0
        while cinco == 0 and test_bt==0:
            #Lectura del dato
            if (arduino.inWaiting()>0):
                texto=arduino.readline()
                texto=texto.split()
                angle_imu=neg*(float(texto[0])-offset)
                despx=getDistanceAngle(angle_imu)
```

```
                imuRect.centerx = HALF_WINWIDTH+despx#Show only the projection in x axis

            #Anàlisis logro de posicòn
            if angle==0:
                if (angle_imu<(angle+0.075)) and (angle_imu>(angle-0.075)):#0.1
                    posangle=1
                else:
                    posangle=0
            elif angle>0:
                if (angle_imu<(angle+angle*0.05)) and (angle_imu>(angle-angle*0.05)):#0.1
                    posangle=1
                else:
                    posangle=0
            elif angle<0:
                if (angle_imu<(angle-angle*0.05)) and (angle_imu>(angle+angle*0.05)):#0.1
                    posangle=1
                else:
                    posangle=0

            test_bt=button('TRIAL END', WINWIDTH-250,WINHEIGTH-
150,125,50,BUTTONCOLOR_OFF,BUTTONCOLOR_ON,WHITE,TESTFONT,"instructions")

            if test_bt==1:
                arduino.reset_input_buffer()
                code='Z'#Disable timer
                arduino.reset_output_buffer()
                arduino.write(str(code).encode('ascii'))
                time.sleep(0.005)
                i=len(ANGLES)+1
                READY=0
                info_count=0
                count_full=0
                COUNT=0
                index_ok=0
                nameTrialFile='T'

            pygame.draw.rect(DISPLAYSURF,color_bar, (0,0,time_limit_bar,30))#GREEN
            pygame.draw.ellipse(DISPLAYSURF,GREEN,targetRect,5)
            pygame.draw.ellipse(DISPLAYSURF,RED_BRIGTH,imuRect,0)

            pygame.display.update(bargoal_rect)
            pygame.display.update(object_rect)
            pygame.display.update(boton_rect)

            arduino.reset_output_buffer()
            arduino.write(str('Q').encode('ascii'))


            pygame.draw.rect(DISPLAYSURF,BLACK, (0,0,time_limit_bar,30))#GREEN
            pygame.draw.ellipse(DISPLAYSURF,color_bg,targetRect,5)
            pygame.draw.ellipse(DISPLAYSURF,color_bg,imuRect,0)
            pygame.draw.rect(DISPLAYSURF,color_bg,(WINWIDTH-250,WINHEIGTH-150,125,50))

            for event in pygame.event.get():
                if event.type == pygame.USEREVENT:
                    time_limit_bar=time_limit_bar-rate
                    if posangle==1:
                        color_bar=GREEN
                    else:
                        color_bar=RED
```

```
                    if cont==count_timer:
                        cinco=1
                        time_limit_bar=WINWIDTH
                    else:
                        cont=cont+1
                elif event.type == pygame.QUIT:
                    terminate()

    if test_bt == 0:
        time.sleep(0.1)
        arduino.reset_input_buffer()
        code='Z'#Disable timer
        arduino.reset_output_buffer()
        arduino.write(str(code).encode('ascii'))
        time.sleep(0.005)
        READY=0
        info_count=0
        count_full=0
        COUNT=0
        index_ok=0
        nameTrialFile='T'

    if demo==1 or numtest==len(PRUEBA):
        inst = 0
        GAME = 0
        trial=0
        demo=0
        sec=0
        numtest=1
        READY=0
        info_count=0
        count_full=0
        COUNT=0
        index_ok=0
        nameDemoFile=''
        nameTrialDirec=''
        nameTrialFile=''
    else:
        numtest=numtest+1
        GAME=0


def TextRect(text,font,color):
    textSurf = font.render(text,True,color)
    textRect = textSurf.get_rect()
    return (textSurf,textRect)


def button(msg,x,y,w,h,ic,ac,tc,font,action=None):
    var = 0

    mouse=0
    click=0

    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()

    if x+w > mouse[0] > x and y+h > mouse[1] > y:
        pygame.draw.rect(DISPLAYSURF,ac, (x,y,w,h))
        if click[0] == 1 and action != None:
```

```
            if action == "instructions" or action == "start" or action=="number" or
action=="back" or action=="info" or action=="complete":
                var = 1
    else:
        pygame.draw.rect(DISPLAYSURF,ic, (x,y,w,h))

    textSurf, textRect = TextRect(msg, font,tc)
    textRect.center = (x+(w/2),y+(h/2))
    DISPLAYSURF.blit(textSurf,textRect)
    return var


def checkForQuit():
    for event in pygame.event.get():
        if event.type == QUIT:
            terminate()

def terminate():
    pygame.quit()
    sys.exit()

if __name__ == '__main__':
    main()
```

BIBLIOGRAFÍA

[1] "smaffer_vgax_ VGA library for Arduino UNO". [En línea]. Disponible en: https://github.com/smaffer/vgax. [Consultado: 17-jul-2018].

[2] "RPi-Distro_RTIMULib_ RTIMULib is a C++ and Python library that makes it easy to use 9-dof and 10-dof IMUs with embedded Linux systems (especially the Raspberry PI and Intel Edison!)". [En línea]. Disponible en: https://github.com/RPi-Distro/RTIMULib. [Consultado: 12-jul-2018].

[3] "pygame". [En línea]. Disponible en: https://www.pygame.org/news. [Consultado: 20-sep-2018].