Proyecto Final de Carrera

Bioingeniería

Instituto Tecnológico de Buenos Aires

Sistema de captura de parámetros espacio-temporales para laboratorio de marcha basado en sensores MEMS

Autor

Fernando Kabas

Tutor

Ing. Alberto Tablon

11 de noviembre de 2020

Resumen

Las evaluaciones de marcha son un elemento relevante en la rehabilitación de pacientes. Hoy en día, los sistemas de captura de movimiento con cámaras optoelectrónicas son el *gold standard* tanto para el análisis cinemático como para el cálculo de parámetros espacio-temporales. Por otro lado, estos sistemas representan un costo elevado y la necesidad de una infraestructura que los aloje. Frente a esta problemática, los sistemas basados en sensores inerciales son una posible solución. En este trabajo se presenta el desarrollo de hardware y software del prototipo de un sistema de captura de parámetros espacio-temporales basado en sensores inerciales MEMS. El dispositivo consiste en una IMU que se coloca en la zona lumbar del sujeto a evaluar y obtiene la cadencia, la velocidad, el largo de paso, la duración del ciclo de marcha, la duración de la fase de balanceo y sustentación.

Agradecimientos

Gracias a todas las personas que cuando me ven, desean lo mejor para mí. Mi familia, porque que es eterna de verdad. Mis amigos del secundario, porque crecimos juntos. Mis amigos de la facultad, porque nos une este camino. Los profesores que entregaron lo mejor de sí, porque hacen que la carrera tenga un propósito. No me quiero olvidar de mencionar especialmente a mi mama, mi papa, Flor, Pocho, Ceci, Nico, Huili y Fran Chaves. Por último quiero dar las gracias a Beto, que me levantó cuando estaba caído en serio. Sin él, este trabajo no hubiese terminado.

Índice general

1. Introducción **7**

2. Marco teórico **9**

2.1. Marcha 9

2.1.1. Aspectos generales 9

2.1.2. Caracterización de la marcha 11

2.1.3. Planos y ejes anatómicos 16

2.2. Laboratorio de marcha 18

2.2.1. Sistemas Optoelectrónicos 19

2.2.2. Sistemas inerciales 20

2.3. IMU: Inertial Meassurement Unit 24

2.3.1. Aceleración lineal y velocidad angular 25

2.3.2. Acelerómetro 26

2.3.3. Giróscopo 27

2.3.4. Configuraciones de uso de las IMU y principios de funcionamiento . . 28

2.3.5. Representación de orientaciones 30

2.3.6. Procesamiento de señales 37

2.3.7. Ruido en sensores inerciales MEMS 41

3. Diseño e implementación **45**

3.1. Adquisición 47

3.2. Control 52

3.3. Comunicación 57

3.4. Alimentación eléctrica 60

3.5. Procesamiento 63

3.5.1. Caracterización de ruido: Varianza Allan 64

3.5.2. Calibración 68

3.5.3. Estimación de la orientación 70

3.5.4. Cálculo de parámetros espacio-temporales 74

4. Experimentos **77**

4.1. Orientación 78

4.1.1. Análisis de base de datos (RepoIMU) 78

4.1.2. Experimento sobre el prototipo 79

4.2. Parámetros espacio-temporales 80

5. Discusión y conclusiones **88**

5.1. Discusión 88

5.2. Conclusiones 90

5.2.1. Trabajo a futuro 91

Bibliografía 92

A. Código de Arduino 98

B. Varianza Allan 100

C. Calibración 109

D. Madgwick 114

E. Scripts 117

Abreviaciones

IMU	Siglas en ingles para Inertial Meassurement Unit.
MEMS	Siglas en ingles para Micro Electro Mechanical System.
PSD	Siglas en ingles para Power Spectral Density. En castellano, Densidad Espectral de Potencia.
VA	Varianza Allan.

Introducción

Las evaluaciones de marcha son un elemento relevante en la rehabilitación de pacientes y pueden servir para hacer un diagnóstico temprano de enfermedades como el Parkinson o el Alzheimer. Es una herramienta que facilita el monitoreo de la evolución de un paciente y la confección de un plan de acción. El principal beneficiado es el individuo y su familia, ya que el paciente recibe un mejor tratamiento y alcanza los objetivos planificados más rápido. Esto es debido a que el estudio de la marcha le ahorra tiempo y trabajo a los kinesiólogos y fisiatras, quienes pueden enfocarse en el análisis de resultados y la planificación. En consecuencia, aumenta la rentabilidad para la institución de salud.

Hoy en día, las evaluaciones de marcha se llevan a cabo de forma artesanal con elementos básicos (cámara de video, cinta adhesiva, papel de color, talco, cinta métrica) o en un laboratorio con tecnología de punta. El primer enfoque resulta ser práctico y realizable en cualquier entorno, aunque por otro lado, el error humano está presente en cada paso del método. La ventaja de realizar mediciones con tecnología más avanzada, es que se disminuye la probabilidad del error, haciendo que los resultados sean consistentes evaluación tras evaluación.

La tecnología gold standard hoy en día, tanto para el análisis cinemático como para el cálculo de parámetros espacio-temporales, es el sistema de captura de movimiento basado en cámaras optoelectrónicas. Estos sistemas representan un costo económico elevado que sólo algunas instituciones pueden pagar. Además, son voluminosos y requieren un amplio

espacio, lo cual es un recurso que suele ser más escaso que el dinero. Además no son sistemas fácilmente portables, lo que elimina la posibilidad de realizar estudios ambulatorios. Frente a estas problemáticas, los sistemas basados en sensores inerciales MEMS son una posible solución, ya que sus componentes son menos costosos, fáciles de transportar, pequeños y de poco peso. Hoy en día existen productos comercializados y validados para el uso asistencial. De todos modos, contínuamente se publican artículos con nuevas implementaciones y algoritmos para obtener parámetros espacio-temporales y cinemáticos de la marcha.

En este trabajo se presenta el desarrollo del prototipo de un sistema de captura de parámetros espacio-temporales basado en sensores inerciales MEMS. El dispositivo consiste en una IMU que se coloca en la zona lumbar del sujeto a evaluar y obtiene la cadencia, la velocidad, el largo de paso, la duración del ciclo de marcha, la duración de la fase de balanceo y sustentación. El proyecto brinda un marco teórico sobre el estudio de la marcha y sobre el procesamiento requerido para acondicionar las señales de la IMU. Luego, construyendo sobre el marco teórico, el cuerpo del informe abarca la implementación del hardware y software necesarios para obtener la orientación de la IMU y calcular los parámetros espacio-temporales. Finalmente, se exponen resultados obtenidos, una discusión, conclusiones y trabajos a futuro.

Marco teórico

2.1. Marcha

2.1.1. Aspectos generales

Cada persona tiene un andar único que se ve influenciado por su estado de salud, de ánimo, su personalidad y su edad, independientemente de la condición en la que se encuentre, ya sea que utilice un bastón, muletas o aún cuando no requiere soporte ortopédico. Lo que el desplazamiento tiene en común entre todos, la esencia, es lo que se denomina marcha.

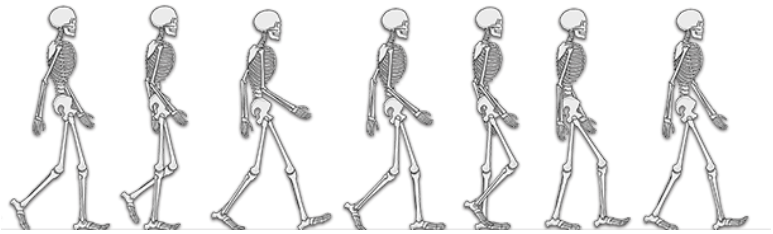


Figura 2.1: Imagen tomada del sitio de Gait & Clinical Movement Analysis Society (http://www.gcmas.org/).

Es un movimiento complejo que depende de la condición de los huesos, músculos, articulaciones y del sistema nervioso. Se requiere balance, fuerza, propiocepción, coordinación y la energía suficiente para movilizar el cuerpo. Los huesos proporcionan estructura y brazos de palanca, los músculos se contraen y extienden, las articulaciones permiten movimiento

y el sistema nervioso coordina la secuencia y se retroalimenta ajustando cada paso [1]. En consecuencia, el estado de salud de la persona se ve plasmado en su respectiva marcha. Una marcha defectuosa implica al cuerpo un costo energético mayor [2, 3] y por ejemplo, si genera incapacidad para bipedestar, causa disminución de la densidad ósea [4, 5]. Además, la habilidad de caminar es una condición necesaria para realizar gran parte de las actividades cotidianas. Es por eso que, sufrir un trastorno en la marcha condiciona todo aspecto en la vida de la persona [6] (Fig. 2.2). Adicionalmente, todo el entorno familiar se vería afectado por la problemática.

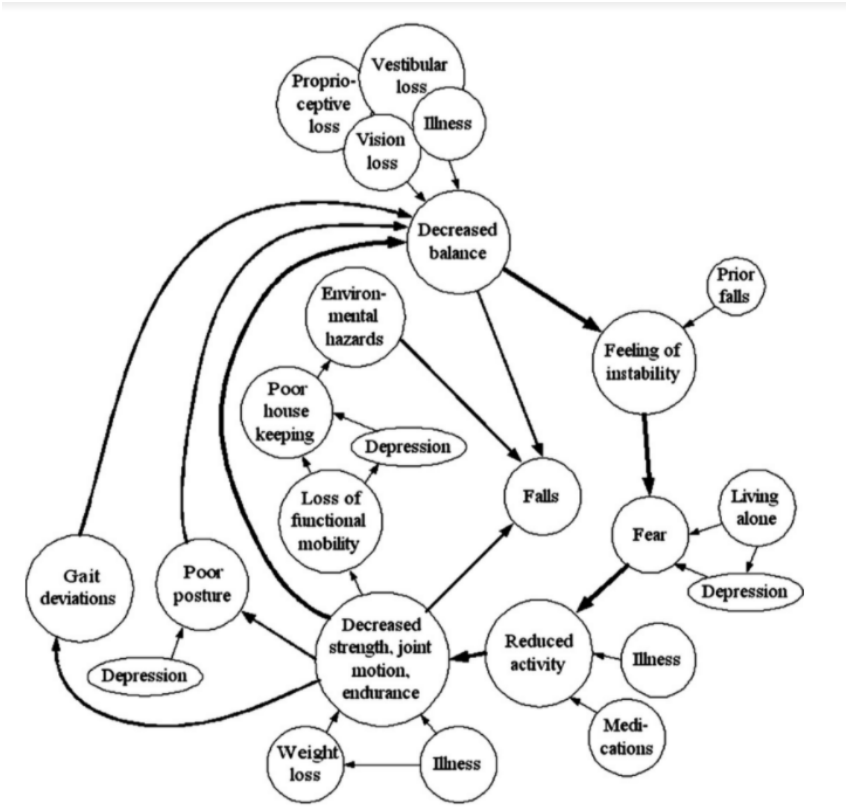


Figura 2.2: Diagrama de los efectos que traen las dificultades para caminar y cómo se retroalimentan entre sí y afectan otros ámbitos de la vida diaria del paciente [7].

El estudio de la marcha puede realizarse con un propósito académico u otro práctico. Académicamente se busca describir y discriminar las marchas normales de las patológicas obteniendo rangos normales para distintos grupos poblacionales. En el ámbito clínico, la evaluación se utiliza para describir los casos en el marco de la afección diagnosticada y de esta forma poder definir un plan de rehabilitación acorde al paciente [8]. También se pueden obtener indicadores de diagnóstico temprano, particularmente en enfermedad de Parkinson o Alzheimer [9, 10, 11, 12].

Las cualidades distintivas de una patología se manifiestan en mayor o menor medida en cada persona (Fig. 2.6). Algunas no requieren auxilio de una prótesis u órtesis pero de todas formas deben pasar por algún período de rehabilitación para recuperarse. Otros necesitan reaprender a caminar, como es el caso de una paraplejía y por ende los procesos y objetivos de rehabilitación serán distintos.

En un centro de rehabilitación se pretende realizar una evaluación inicial para definir tales procesos y objetivos en función del paciente [13]. Luego esta se repite periódicamente para llevar a cabo el control y seguimiento del tratamiento. Comparando los resultados evolutivos con los objetivos estimados, se pueden hacer ajustes sobre la planificación y obtener conclusiones del trabajo realizado.

Para que cada evaluación sea comparable con el resto, se requiere que por un lado sean consistentes en su forma y metodología y por el otro objetivas en su análisis e interpretación. Naturalmente, la calidad de su desarrollo y de su análisis depende de la experiencia de los kinesiólogos y fisiatras, aunque de todas formas no se puede garantizar consistencia y homogeneidad. Por ello, la evaluación de marcha se apoya en distintas tecnologías para "medir" el movimiento y destilar su complejidad en indicadores útiles para el profesional asistencial. Los sistemas de captura de movimiento, las plataformas de fuerza y la electromiografía ambulatoria son las tres tecnologías más importantes que permiten realizar una evaluación clínica o una descripción de carácter académico [14, 15, 16].

2.1.2. Caracterización de la marcha

La marcha sigue un patrón cíclico (Fig. 2.4) el cual se emplea para normalizar los datos del paciente y comparar con otras evaluaciones. Se suelen ubicar los parámetros medidos a lo largo de un eje temporal representado en porcentajes de 0 % a 100 % del ciclo de marcha (Fig. 2.6). La marcha normal se divide claramente en 7 etapas (Fig. 2.3), las cuales inician con los siguientes eventos:

- Contacto inicial (*initial contact*)

- Despegue de dedos del pie opuesto (*opposite toe off*)
- Despegue de talón (*heel rise*)
- Contacto inicial opuesto (*opposite initial contact*)
- Despegue de dedos del pie (*toe off*)
- Pies adyacentes (*feet adjacent*)
- Tibias paralelas (*tibia vertical*)

Los primeros cuatro ocurren durante la fase de sustentación (*stance phase*) que dura el 60 % del ciclo y se dividen en:

- Respuesta a la carga (*loading response*)
- Sustentación media (*mid stance*)
- Sustentación terminal (*terminal stance*)
- Pre balanceo (*pre swing*)

Los últimos tres ocurren durante la fase de balanceo (*swing phase*) que dura el restante 40 % del ciclo y la dividen en:

- Balanceo inicial (*initial swing*)
- Balanceo medio (*mid swing*)
- Balanceo terminal (*terminal swing*)

Durante un ciclo de marcha se ejecuta una zancada (*stride*) compuesta por un paso (*step*) izquierdo y otro derecho.

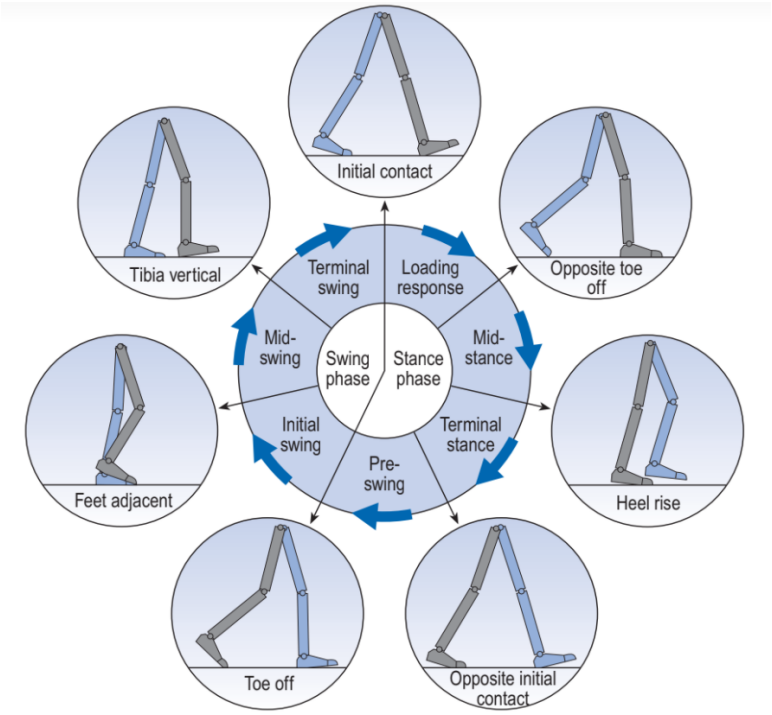


Figura 2.3: Fases y eventos del ciclo de marcha. Imagen de [17].

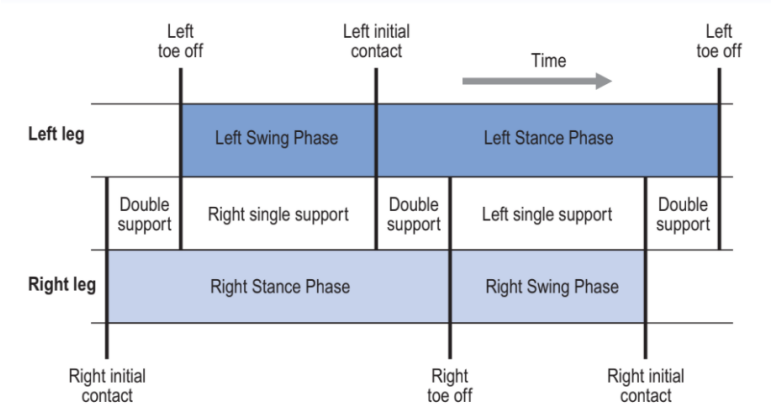


Figura 2.4: Ciclo de marcha. Imagen de [17].

En el tren inferior los miembros se representan como segmentos y las articulaciones como rótulas. Los miembros son 7: una pelvis, dos fémures, dos tibias y dos pies. Las articulaciones son 6: dos caderas, dos rodillas y dos tobillos (Fig. 2.5).

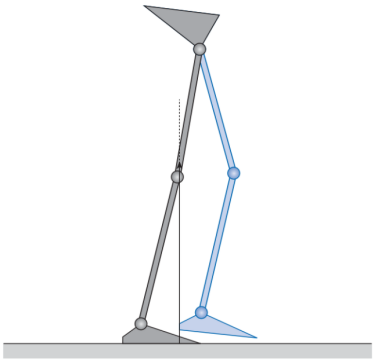
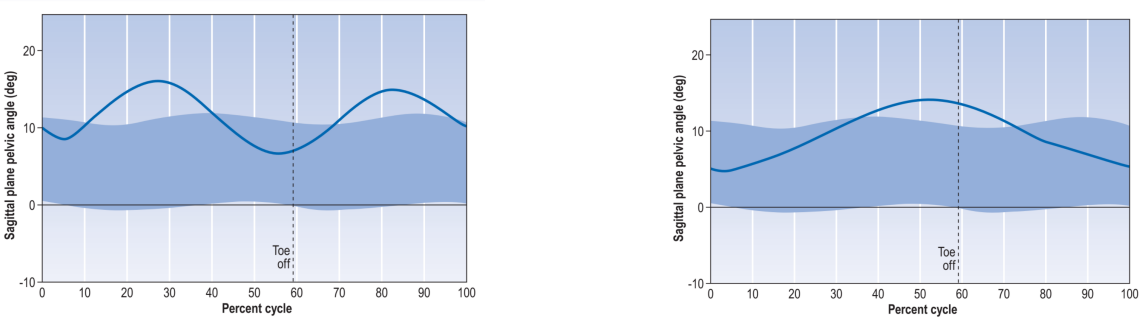


Figura 2.5: Ilustración de los 7 segmentos y 6 articulaciones durante el despegue de talón de una pierna. Imagen de [17].

Para un ciclo de marcha se puede realizar un análisis cinemático de cada articulación en el que se miden los ángulos de flexión y extensión para luego graficarlos (Fig. 2.6).



a b

Figura 2.6: Ejemplo del ángulo de inclinación de la pelvis en el plano sagital (color azul) para casos de parálisis cerebral. En azul pálido se muestra el rango normal para un adulto. En línea negra punteada se indica el momento del despegue de los dedos del pié. (a) Esta forma de doble pico suele darse cuando los flexores de las caderas son espásticos y los isquiotibiales débiles. (b) Esta forma pico único suele darse cuando un lado del cuerpo es más afectado que otro. Imágenes de [17].

También es posible hacer una reconstrucción por computadora del desplazamiento si se posee un software que, en base a un modelo matemático y los parámetros medidos, pueda generar un renderizado 3D del cuerpo en movimiento (Fig. 2.7). La metodología estándar hoy en día es ubicar cámaras infrarrojas alrededor del paciente y colocarle marcadores en puntos anatómicos que se correspondan con el modelo utilizado. Esta tecnología de captura de movimiento se denomina sistema optoelectrónico, el cual será detallado en la sección 2.2.1.

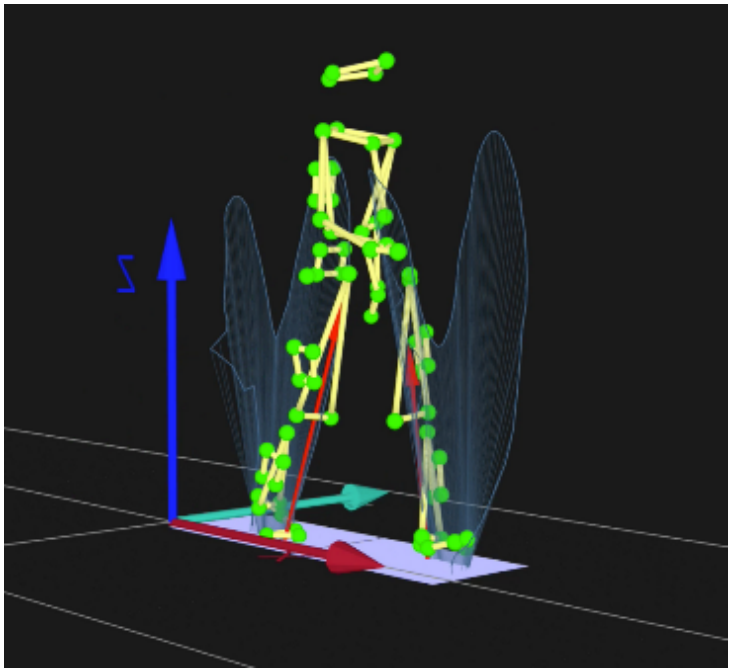


Figura 2.7: Reconstrucción del movimiento capturado de un individuo. Se puede ver la ubicación de los marcadores (esferas verdes) y también la dirección y magnitud de la fuerza de reacción del suelo (líneas celestes). Imágen de la Universidad de Livepool (https://www.liverpool.ac.uk/ageing-and-chronic-disease/research-groups/evolutionary-biomechanics/about/).

Paralelamente se miden parámetros espacio-temporales que, como su nombre indica, son relaciones de espacio y tiempo que sirven como indicadores para una descripción general del patrón de marcha. De hecho, la evaluación de marcha objetiva más simple sólo requiere de 4 parámetros espacio-temporales: largos de zancada, largos de paso, cadencia y velocidad [18]. A estos se suman: tiempo de ciclo, base de sustentación y ángulo de pronación de pies (Fig. 2.8).

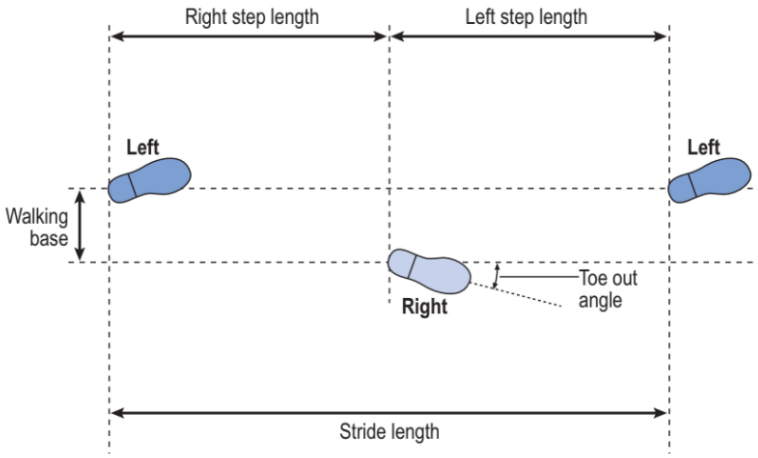


Figura 2.8: Esquema de algunos parámetros espacio-temporales: base de sustentación, largos de paso, largo de zancada y ángulo de pronación de pies. Imagen de [17].

También se suele medir la frecuencia cardíaca en reposo y en actividad para cruzarlas con la velocidad y calcular el índice de costo fisiológico que es un indicador de la energía que debe utilizar el paciente [19, 20].

El análisis visual y cinemático de la marcha tiene el propósito de realizar una descripción detallada del movimiento pero no implica contemplar las causas del mismo [15]. Es por eso que se complementa con un análisis cinético, en el cual se utiliza una plataforma de fuerza que mide la reacción del suelo contra los pies en 3 dimensiones [16]. En base a tal reacción, se calculan los momentos de fuerza que soporta cada segmento y articulación. A su vez, esta información se suma a un estudio de la electromiografía de los músculos [14]. Es usual que el paciente requiera compensar sus esfuerzos entre distintos músculos para sobreponerse a los impedimentos que la patología le impone para poder caminar.

El estudio completo (cinemático, cinético y electromiográfico) entrega una gran cantidad de información, pero interpretar y relacionar cada indicador puede ser innecesario. Existen protocolos que reducen el resultado de una evaluación de marcha a un único índice, aunque hasta el momento ninguno es considerado el estándar. Cimolin *et. al.* [21] los describe.

2.1.3. Planos y ejes anatómicos

Para describir movimientos del cuerpo humano se utilizan cierta terminología, la cual es importante a la hora de presentar resultados. Se introducen 3 planos y 3 ejes:

- plano transversal
- plano frontal
- plano sagital
- eje antero-posterior (con sentido positivo hacia adelante)
- eje latero-lateral (con sentido positivo hacia la izquierda)
- eje céfalo-caudal (con sentido positivo hacia la cabeza)

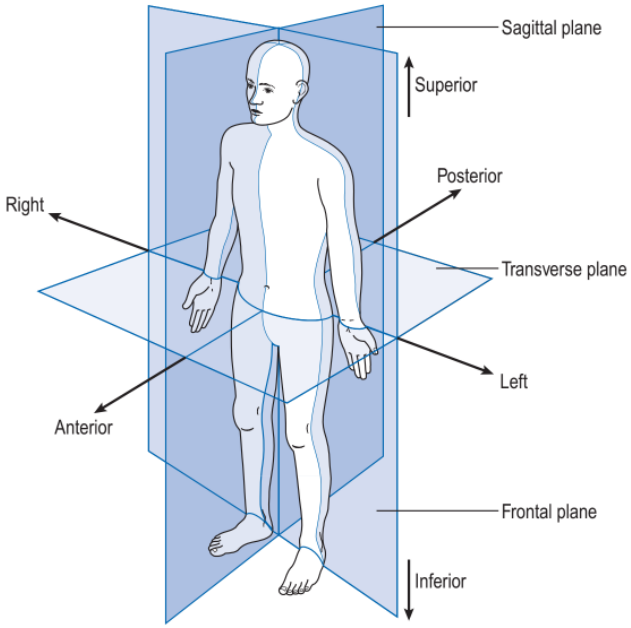


Figura 2.9: *Transverse plane* es el plano transversal. *Frontal plane* es el plano frontal. *Saggital plane* es el plano sagital. El eje antero-posterior es el indicado por las flechas *Anterior* (hacia adelante) y *Posterior* (hacia atrás). El eje latero-lateral es el indicado por las flechas *Right* (hacia la derecha) y *Left* (hacia la izquierda). El eje céfalo-caudal es el indicado por las flechas *Superior* (hacia la cabeza) e *Inferior* (hacia los pies). Imágen de [17]

Durante este trabajo se hará uso de terminología en referencia a la pelvis. Los ángulos de rotación de la misma a lo largo de los ejes antero-posterior, latero-lateral y céfalo-caudal se conocen de distintas formas. Aquí se los nombrará como:

- Abducción (a lo largo del eje antero-posterior)
- Inclinación (a lo largo del eje latero-lateral)
- Rotación (a lo largo del eje céfalo-caudal)

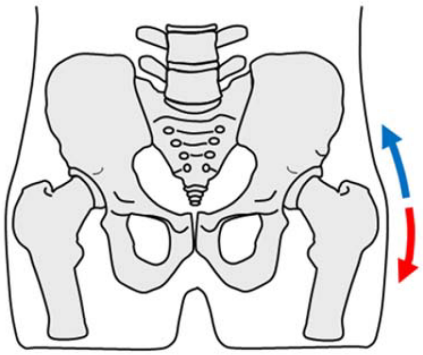


Figura 2.10: Vista de la pelvis por el plano frontal. La flecha azul representa una rotación positiva, la flecha roja, una negativa. Imagen de [22]

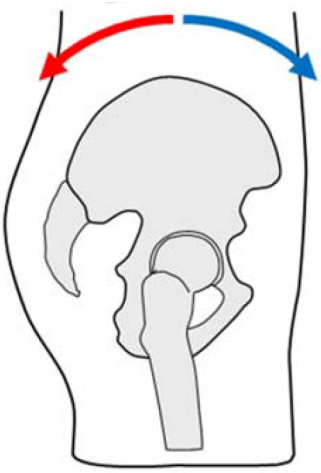


Figura 2.11: Vista de la pelvis por el plano sagital. La flecha azul representa una rotación positiva, la flecha roja, una negativa. Imagen de [22]

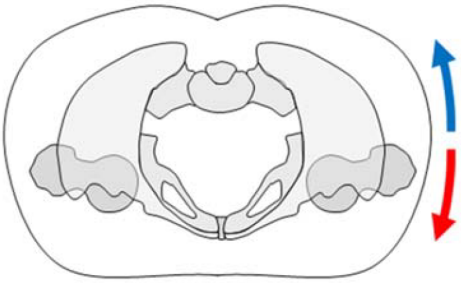


Figura 2.12: Vista de la pelvis por el plano transversal. La flecha azul representa una rotación positiva, la flecha roja, una negativa. Imagen de [22]

2.2. Laboratorio de marcha

Un laboratorio de marcha es el espacio en donde se realizan evaluaciones de carácter académico o evaluaciones clínicas para el relevamiento de un paciente atravesando una rehabilitación. Según lo visto en la sección 2.1.2, sus tres componentes principales son plataformas de fuerza, electromiógrafos portátiles y un sistema de captura de movimiento.

De todas las facetas del estudio del laboratorio de marcha, este trabajo se enfoca en el análisis cinemático. El mismo puede ser llevado a cabo ya sea a simple vista, con filmaciones o con un sistema optoelectrónico de captura de movimiento.

La tecnología utilizada depende de los recursos económicos, la infraestructura y el per-

sonal capacitado que posea la institución sanitaria. Los sistemas optoelectrónicos son muy precisos pero requieren amplio espacio para ser instalados y son más costosos que los sistemas inerciales. Estos últimos son portátiles y no necesitan un recinto cubierto ni espacioso para utilizarse. Adicionalmente permiten realizar estudios ambulatorios [23, 24]. Hasta el momento han probado ser útiles, especialmente para medir parámetros espacio-temporales [25], predecir riesgo de caídas [26] y realizar monitoreo de actividades [27]. Los sistemas inerciales ya se encuentran en el mercado y paralelamente continúa la investigación y desarrollo de los mismos.

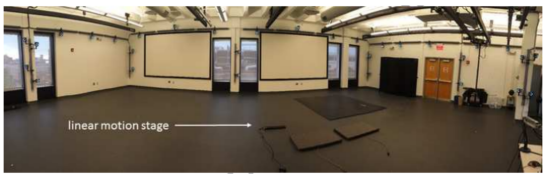
2.2.1. Sistemas Optoelectrónicos

El principio de funcionamiento de un sistema optoelectrónico es la triangulación de la posición de objetos en el espacio. Cámaras optoelectrónicas capturan frecuencias determinadas de luz infrarroja y triangulan la posición de marcadores ubicados en las superficies a rastrear. Estos marcadores pueden ser pasivos o activos, es decir que pueden reflejar la luz infrarroja del ambiente o bien producirla. Para realizar la triangulación de un marcador, el mismo debe estar presente en el campo de visión de al menos dos cámaras [28]. En la práctica se utilizan más de dos para generar redundancia y aumentar la precisión del sistema, además de prevenir que ocurra una obstrucción de los marcadores durante la grabación y se pierda la información de esos instantes [29].

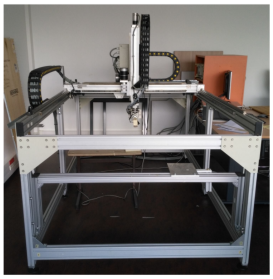
Los sistemas optoelectrónicos son el gold standard entre las tecnologías de captura de movimiento. Alcanzan una gran precisión, tanto es así que se utiliza para validar otros sistemas [30]. También son empleados en campos donde se requiere capturar un alto detalle de los objetos de interés, como por ejemplo en sistemas de realidad virtual y animaciones 3D cinematográficas.

Dependiendo del sistema, la cantidad de cámaras y el volúmen de captura, se alcanzan distintos niveles de precisión. Aurand et. al. [31] reporta un error estático menor a 0,2mm en el $97\,\%$ de un volumen de captura de 135m^3 (6,6m x 10,4m x 1,97m) utilizando 42 cámaras OptiTrack (TM) y en el $91\,\%$ del volumen utilizando 21 cámaras. Merriaux et. al. [32] concluye que un sistema Vicon (TM) de 8 cámaras en un volumen de 3m^3 (2m x 1,5m x 1m) asegura

un error menor a 2mm para movimientos de hasta 7m/s. McGinley et. al. [33] concluye que el error para ángulos de rotación en los planos sagital y coronal son menores a 5° en la mayoría de los casos.



a



b

Figura 2.13: (a) Laboratorio donde Aurand et. at. realizó su experimento en un volumen de 135m^3. Imagen de [31]. (b) Robot utilizado por Merriaux et. al. Su volumen interno (3m^3) fue capturado por cámaras optoelectrónicas siguiendo un marcador infrarrojo. Imagen de [32].

Sumado a los errores implícitos en la tecnología, también existen errores metodológicos. Debido a que es un sistema no invasivo, la ubicación de los marcadores no es la misma que la de los huesos. Entonces, es necesario definir una serie de transformaciones y calibraciones. Cómo y dónde colocar los marcadores se determina en base a modelos anatómicos establecidos [34, 35, 36, 37] aunque ninguno es considerado el estándar de forma unánime [38].

2.2.2. Sistemas inerciales

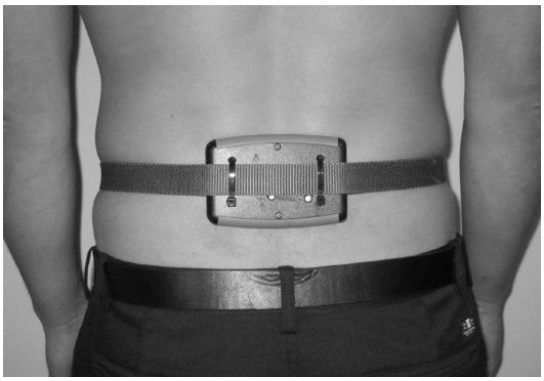
El sistema inercial con el que se evalúa la marcha se conforma por un conjunto de IMUs (Inertial Meassurement Units). A su vez, cada IMU se compone de un acelerómetro y un giróscopo, los cuales sensan su propia aceleración lineal y velocidad angular respectivamente. También existe la configuración de MIMU (Magnetic and Inertial Meassurement Unit) que incluye un magnetómetro y permite ubicar la IMU en el espacio de forma absoluta.

Los sistemas inerciales prometen ser una alternativa a los optoelectrónicos [24, 7] para el análisis de marcha y para estudios más cortos y específicos: cuantificación de temblores, estabilidad, tests instrumentados de movilidad [23] y además suman la prestación de estudios ambulatorios como monitoreo de actividad física [23, 27] y predicción de caídas [26].

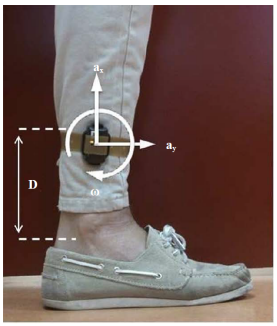
Jarchi et. al. [39] realiza una revisión de la literatura con respecto al uso de acelerómetros en el campo del análisis de marcha y concluye que la tendencia es el desarrollo de sistemas con un sólo sensor, abriéndose la posibilidad a utilizar dispositivos vestibles o comunes (ej.: celulares, o relojes).

El uso de los IMUs para obtener parámetros espaciotemporales en personas sanas muestra buena evidencia [40, 41, 25] y repetibilidad [42, 43, 44]. Yeo et. al. [45] concluye que un sistema de captura inercial es apto para determinar parámetros espacio-temporales y el rango de movimiento de las rodillas en personas sanas. Con un acelerómetro tri-axial en la zona lumbar, Zijlstra et. at. [46] obtiene parámetros espacio-temporales en personas sanas, entre ellos la velocidad con un error de 0,05 m/s^2. Esser et. al. [47] muestra que esa estimación de la velocidad es útil en pacientes con Parkinson y distrofia muscular pero debe ser modificada para casos de enfermedades de neuronas motoras y ACVs.

La precisión que se alcanza a la hora de calcular parámetros espacio-temporales depende de la señal utilizada y también del lugar del cuerpo de donde se hace la captura. Por ejemplo, la velocidad angular a lo largo del eje latero-lateral de la pantorrilla presenta los mejores resultados para identificar el contacto inicial y final. Por su parte la aceleración vertical de la zona lumbar o el tronco son las señales predilectas para calcular el largo de zancada.



a



b

Figura 2.14: (a) Imagen de una IMU colocada en la zona lumbar [48]. (b) Imagen de una IMU colocada en la pantorrilla. [49].

Sessa et. al. [50] determina que el mejor lugar para ubicar una IMU y segmentar el ciclo de marcha es en la pantorrilla. Kuo et. al. [51] obtiene mejores resultados para la estimación de largo de paso con IMUs en el tronco y mayor precisión en el conteo de pasos con giróscopos

en las pantorrillas.

Calibración de sistemas inerciales con sensores MEMS

La tecnología MEMS presenta errores intrínsecos que varían aleatoriamente. Existe una extensa bibliografía caracterizándolos y proponiendo calibraciones para mitigarlos. La mayoría de las metodologías de calibración implican utilizar equipamiento específico, como cámaras de temperatura regulada y mesas de rotación. Para una aplicación clínica esto no resulta práctico. Afortunadamente, existen metodologías que no requieren equipamiento extra y pueden ser realizadas previas a una evaluación de marcha [52, 53, 54]. Detalles sobre estos procesos de calibración serán descritos en la sección 3.5.2, ya que la fuente de error que los motiva es propia de los sensores y no de la naturaleza de la aplicación de captura de movimiento.

Posicionamiento y calibración sensor-segmento

A la hora de posicionar la IMU sobre el cuerpo del paciente, no existe una convención definida formalmente. No obstante, los productos comerciales y trabajos de investigación ubican una IMU por segmento.

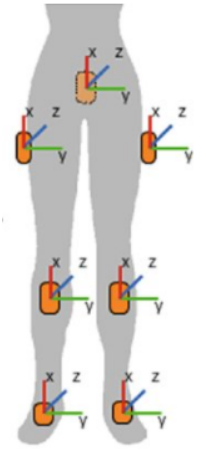
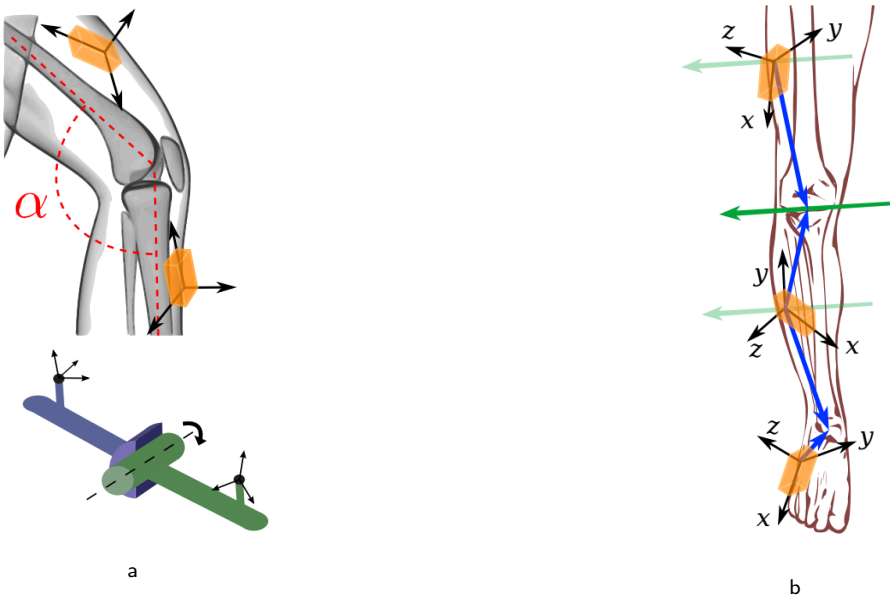


Figura 2.15: Ilustración del posicionamiento de IMUs, cada una correspondiéndose con un miembro del tren inferior. Imagen de [55].

Idealmente, un eje de la IMU debería coincidir con el eje longitudinal del hueso y otro debería coincidir con el eje de rotación de la articulación [56]. De esta forma, la pose de la IMU sería análoga a la del segmento. En muchos casos es difícil que esto ocurra, ya que al ser un método no invasivo, las IMU se ubican sobre la piel y separadas del hueso por los tejidos del cuerpo.

Algunos trabajos ignoran la desalineación de las IMUs, asumiendo que se las puede ubicar suficientemente bien [57, 58]. No obstante, es una aproximación que no siempre se justifica si se quiere reconstruir el movimiento en 3D contemplando más de un segmento del cuerpo [59].

Para corregir tal error, la posición y orientación sensor-segmento puede ser medida manualmente, aunque resulta complicado y no genera buenos resultados [58, 60].



a b

Figura 2.16: Sistemas de referencia locales de cada IMU. (a) Ningún eje de los sistemas locales de referencia de las IMU coincide con los ejes que definen el ángulo de flexión. (b) Su distanciamiento con las articulaciones (flechas azules) y con los ejes de rotación (flechas verdes). Imagen de [59].

Otra solución es realizar una calibración. Algunas requieren de posturas estáticas [61], otras utilizan movimientos definidos para relacionar las rotaciones entre IMUs [62]. Por su parte, Seel et. al. [59] utiliza el conocimiento a priori de los grados de libertad de las articulaciones. Así puede definir una relación entre las señales de los sensores a ambos lados

de cada articulación sin importar la posición ni la orientación de los mismos. En base a esa relación define una función de costo que puede ser minimizada. Entonces, no se requiere que el paciente realice movimientos pre definidos, sino que pueden ser aleatorios.

Figura 2.17: Imágenes de los movimientos necesarios para distintas calibraciones [59].

2.3. IMU: Inertial Meassurement Unit

Una IMU se compone de acelerómetros y giróscopos. Su nombre se debe a que el principio de funcionamiento de estos sensores implica medir el comportamiento inercial de una masa conocida. Gracias a ellos, el dispositivo puede determinar su aceleración lineal y velocidad angular instantáneas en 3 dimensiones.

La aceleración lineal y velocidad angular son propiedades físicas vectoriales. Hoy en día, cada sensor se compone de 3 ejes ortogonales (Fig. 2.19). Así se puede representar la magnitud, orientación y sentido en el espacio tridimensional (Fig. 2.18).

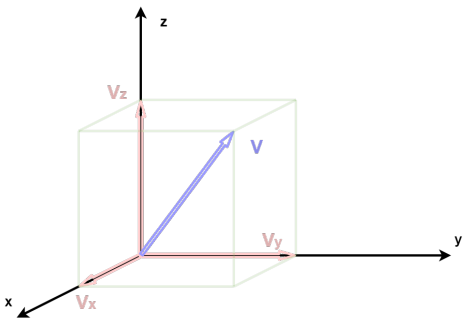


Figura 2.18: Representación de un vector V en un sistema de coordenadas ortogonal y sus proyecciones a los ejes X, Y y Z.

En la actualidad se utilizan sensores MEMS contenidos en encapsulados de tamaño del orden de milímetros (ej.: las medidas del chip LSM9DS1 utilizado en este trabajo son 3,5x3x1,0 mm), lo que permite integrarlos fácilmente en cualquier dispositivo móvil [63].

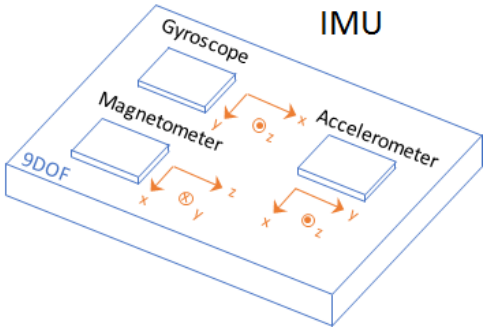


Figura 2.19: Diagrama de un acelerómetro, giróscopo y magnetómetros ubicados en un mismo plano con sus ejes paralelos y perpendiculares respectivamente, pero no colineales. Imagen de Mathworks (https://es.mathworks.com/help/nav/ug/model-imu-gps-and-insgps.html).

2.3.1. Aceleración lineal y velocidad angular

La aceleración es la rapidez de cambio de la velocidad y esta a su vez es la rapidez de cambio de la posición. Se definen como la segunda y primera derivada de la posición con respecto al tiempo:

$$\boldsymbol{a} = \frac{\delta \boldsymbol{v}}{\delta t} = \frac{\delta}{\delta t}\left(\frac{\delta \boldsymbol{r}}{\delta t}\right) \tag{2.1}$$

Donde $\boldsymbol{r}$ es el vector posición, $\boldsymbol{v}$ es el vector velocidad lineal y $\boldsymbol{a}$ es el vector aceleración.

Similarmente a la velocidad lineal, la velocidad angular representa la rapidez de cambio de la posición angular y se define como:

$$\boldsymbol{w} = \frac{\delta \boldsymbol{\theta}}{\delta t} \tag{2.2}$$

Donde $\boldsymbol{\theta}$ es el vector posición angular y $\boldsymbol{w}$ es el vector velocidad angular.

2.3.2. Acelerómetro

La aceleración sensada por los acelerómetros es producto de la fuerza de gravedad y otras fuerzas externas. Para explicar su funcionamiento, una forma de representar la estructura del acelerómetro es la de una masa suspendida entre dos resortes Fig. (2.20).

Cuando el sensor se encuentra en reposo sobre una superficie horizontal, la suma de fuerzas externas es cero. Una es la fuerza de gravedad y otra es la fuerza de reacción que la cancela. La masa del eje que esté alineado con la gravedad, se desplaza, estirando un resorte hasta volver al estado de reposo. En ese caso la señal de salida representa que en esa dirección hay una aceleración de magnitud 1g o 9,8m/s^2. Para un eje que se encuentre perpendicular, la masa se mantiene en el centro, indicando que no hay aceleración.

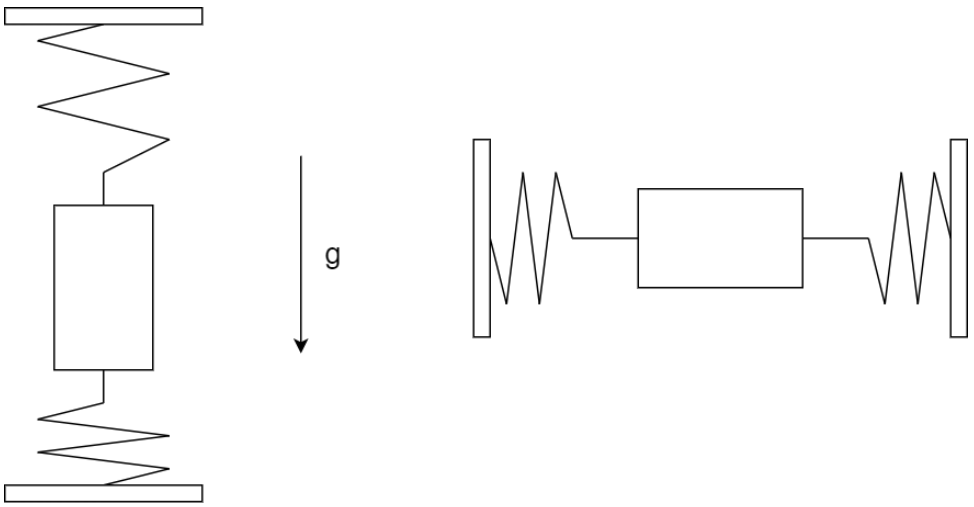


Figura 2.20: Diagrama de los ejes de un acelerómetro. Una masa se encuentra suspendida entre dos resortes. Al aplicarse una fuerza sobre el sistema, la masa se desplaza en sentido opuesto de forma proporcional a la aceleración provocada. Cuando el acelerómetro se encuentra estático, sólo hay una fuerza externa aplicada que se opone al peso del sistema (o fuerza de gravedad). En ese caso, cualquier eje que se encuentre alineado con la fuerza de gravedad, indicará una aceleración de 1g. Por el mismo motivo, cualquier eje perpendicular a la fuerza de gravedad, no indica aceleración.

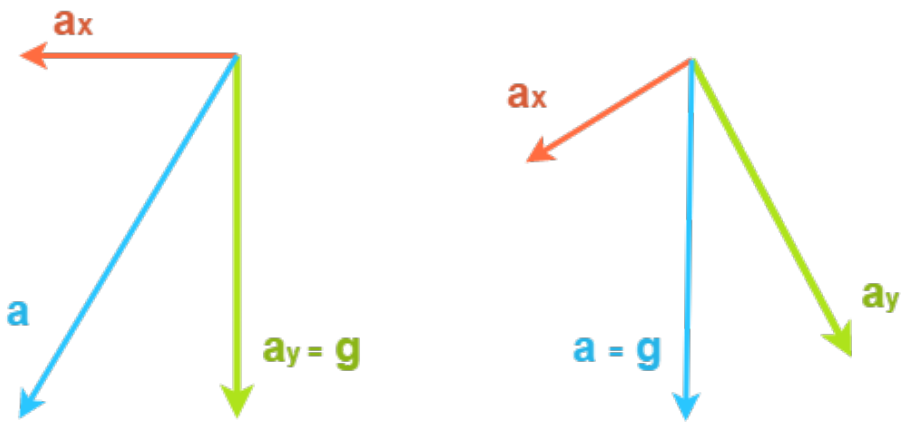


Figura 2.21: Diagrama de la aceleración sensada por los ejes x (vector rojo) e y (vector verde) de un acelerómetro. El vector azul es el resultante. A la izquierda, se muestra el caso en que el eje y se encuentra paralelo a la dirección de la gravedad mientras el acelerómetro se encuentra afectado por una aceleración en la dirección del eje x. En este caso, la magnitud del vector resultante es mayor al de la gravedad. A la derecha, se muestra el caso en que el acelerómetro se encuentra estático y con ningún eje paralelo a la dirección de la gravedad.

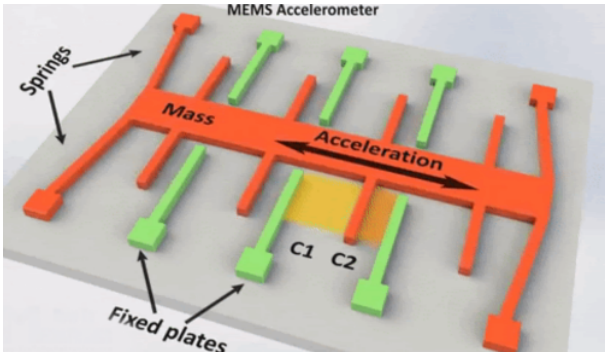


Figura 2.22: Ilustración de la estructura de un eje de un acelerómetro MEMS. La masa se mueve sostenida por los resortes y ese movimiento es capturado como la variación de la capacidad entre la masa y otras placas fijas.

2.3.3. Giróscopo

El giróscopo sensa la velocidad angular. Un sensor lo suficientemente sensible podría medir la velocidad de rotación de la tierra, pero en este caso se considera que un giróscopo estático o en traslación pura, debe sensar una velocidad angular nula.

El giróscopo mems uni-axial se compone de dos marcos coplanares concéntricos, uno interno y otro externo (Fig. 2.23). Sólo pueden desplazarse en un plano perpendicular al eje sensitivo del sensor. El marco externo es solidario al sensor en una sola dirección y libre en la perpendicular. El marco interno es solidario al externo, de forma similar a la que el externo es

solidario al sensor. La diferencia es que el marco interno es libre de desplazarse con respecto al marco externo en la dirección que el marco externo es solidario al sensor.

Al marco externo se lo hace oscilar con una frecuencia y amplitud conocidas. Cuando el sensor rota a lo largo del eje sensitivo, el marco interno comienza a oscilar en la dirección de su único grado de libertad disponible. Esto se debe a que la oscilación del marco externo es forzada y el sistema debe mantener constante su momento angular según la ley de conservación del momento angular. La magnitud de la velocidad de rotación aplicada sobre el sensor se refleja en la amplitud de oscilación del marco interno.

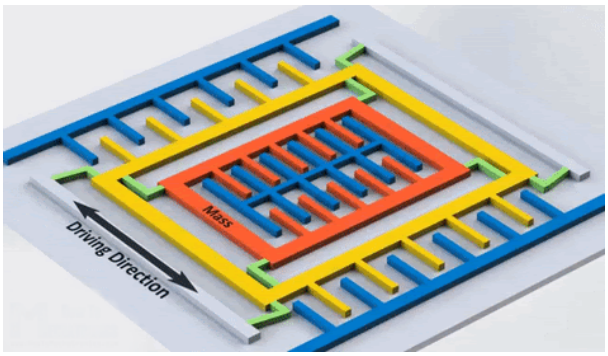


Figura 2.23: Ilustración de la estructura de un eje de un giroscopio MEMS. Se compone de dos marcos coplanares. El marco externo (amarillo) se encuentra sujeto al sensor excepto en una dirección, la cual es perpendicular a la del eje sensitivo y el marco puede desplazarse a lo largo de ella. El marco interno (naranja) es solidario al externo, con el agregado de que la dirección en que se puede desplazar también es perpendicular a la dirección de desplazamiento del marco externo.

2.3.4. Configuraciones de uso de las IMU y principios de funcionamiento

Las IMU se utilizan en dos configuraciones distintas. La primera se conoce como "plataforma estable" [63] y consiste en disociar la IMU del objeto al cual está adosada para poder sensar desviaciones en relación al sistema de coordenadas de referencia externo al objeto o sistema de referencia global. Luego se retroalimenta la información al sistema y se corrige la orientación de la plataforma para mantenerla alineada con el estado instantáneamente previo. El método más usual es ubicar la plataforma suspendida mediante tres marcos concéntricos articulados como juntas universales o cardanes – conocidos en inglés como "gimbals" [63].

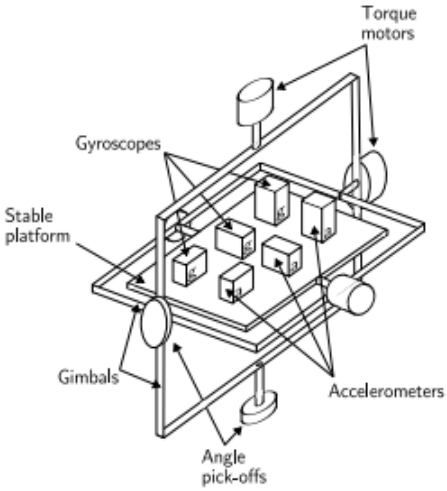


Figura 2.24: IMU en configuración de plataforma estable. La plataforma se ubica articulada mediante cardanes ("gimbals"). Mientras el sistema se mueve, los cardanes permiten que la plataforma mantenga su pose. Imagen de [63].

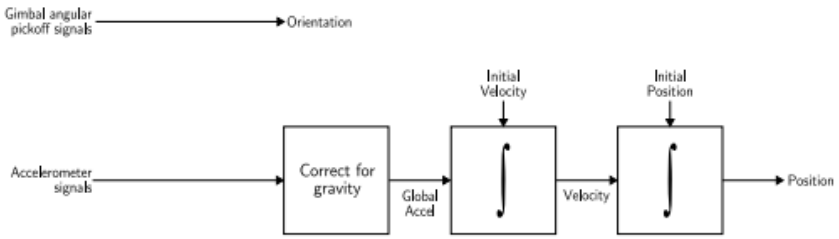


Figura 2.25: Algoritmo básico para una IMU de plataforma estable. La orientación se obtiene de la rotación de los cardanes. Cualquier rotación sensada desde los giróscopos se utiliza para corregir la pose de la plataforma. El desplazamiento es calculado en base a los acelerómetros asumiendo que la plataforma se mantiene alineada con el sistema global de referencia y por ende conociendo la dirección de la aceleración de gravedad. Imagen de [63].

Este paradigma se emplea, por ejemplo, en helicópteros quadrotor para estabilizarlos de forma horizontal y compensar disrupciones causadas por el entorno. Otro uso común es en estabilizadores de cámaras de video utilizados para cinematografía.

Por su parte, la configuración "strapdown" mantiene la IMU solidaria al objeto de interés, por lo que el movimiento sensado es en relación a su propio sistema de referencia. Son sistemas menos voluminosos. Sumado a esto, los algoritmos de procesamiento utilizados han avanzado y actualmente son la configuración más utilizada de las dos [63].

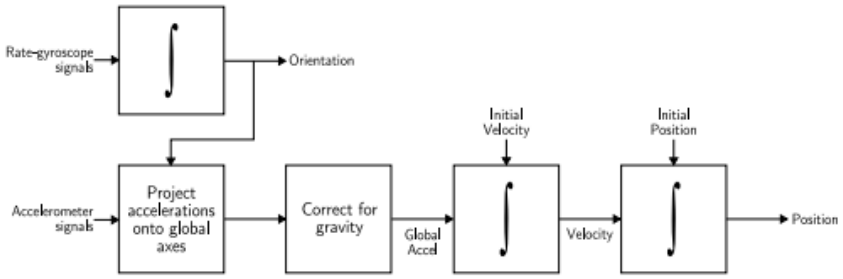


Figura 2.26: Algoritmo básico para una IMU en configuración strapdown. La orientación del objeto de interés se calcula integrando la velocidad angular. La dirección de la gravedad es definida en base a la orientación calculada. Luego se corrige la aceleración y se calcula la velocidad y posición. Imagen de [63].

2.3.5. Representación de orientaciones

Para entender cómo expresar la orientación de una IMU, se debe introducir la matemática pertinente. No sólo para describir una orientación, sino también para definir cómo se aplica una rotación a un vector, transformándolo. Eso es necesario para propagar la orientación de la IMU a lo largo del tiempo e implementar algoritmos.

La orientación es un concepto que se define en relación a un sistema de referencia ortogonal y existe más de una manera de representarla. En sí mismo, un sistema de referencia puede definirse en base a su orientación con respecto a otro sistema. Para la aplicación actual, es conveniente establecer dos: uno local a la IMU y otro global fijado a la tierra.

A lo largo del trabajo se hará referencia al sistema global con una E y al local con una S. Además, los vectores y matrices se representarán con letras de mayor peso, minúscula y mayúscula repectivamente. Por ejemplo: x es un escalar y $\boldsymbol{x}$ un vector y $\boldsymbol{X}$ una matriz.

También se utilizarán nombres conocidos dentro del campo de la navegación aeronáutica. Los ejes del sistema de referencia local se conocen como Roll (eje x), Pitch (eje y) y Yaw (eje z) que, en relación a lo mencionado en la sección 2.1.3, son el eje antero-posterior, latero-lateral y céfalo-caudal respectivamente.

A continuación se introducen las 3 representaciones de orientación más utilizadas: matriz

de rotación, ángulos de euler y quaterniones. La sección termina con una descripción de las ventajas y desventajas de cada representación.

Matriz de rotación

La matriz de rotación[1] es una matriz de 3x3, cuyas columnas y filas son ortonormales y representan la transformación de las coordenadas de un punto luego de una rotación.

Soponiendo un vector $\boldsymbol{x}$ colineal al eje $\hat{x}$ de un sistema de referencia ortogonal cualquiera, una rotación que alinee al vector $\boldsymbol{x}$ con el eje $\hat{y}$, se expresa de la siguiente forma:

$$\boldsymbol{x}^{rot} = \boldsymbol{C}\ \boldsymbol{x} \tag{2.3}$$

$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.4}$$

La matriz de rotación $\boldsymbol{C}$ expresa la orientación entre un sistema de referencia y otro. El elemento de la fila i y la columna j es el coseno del ángulo entre el eje i del sistema global y el eje j del local.

$$\boldsymbol{r}^E = \boldsymbol{C}_S^E\ \boldsymbol{r}^S \tag{2.5}$$

$$\boldsymbol{C}_S^E = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{2.6}$$

Donde $\boldsymbol{r}$ representa un vector cualquiera expresado en el sistema de referencia local ($\boldsymbol{r}^S$) o en el sistema global ($\boldsymbol{r}^E$)

La inversa de la matriz de rotación es igual a su transpuesta. Por lo tanto, conociendo

[1] Conocida también como matriz de cosenos directores.

la transformación de un sistema al otro, el camino inverso es trivial.

$$\boldsymbol{r}^S = \boldsymbol{C}_E^S \; \boldsymbol{r}^E = (\boldsymbol{C}_S^E)^T \; \boldsymbol{r}^E \tag{2.7}$$

La propagación de un movimiento de rotación consiste en aplicar una secuencia de rotaciones, cada una asociada a instantes de tiempo contiguos. Tales rotaciones, si pueden aproximarse por pequeños ángulos, permiten ser expresadas de la siguiente forma:

$$\boldsymbol{C}_S^E(t+\delta t) = \boldsymbol{C}_S^E(t) \; \boldsymbol{A}(t) \tag{2.8}$$

$$\boldsymbol{A}(t) = [\boldsymbol{I} + \delta\boldsymbol{\Psi}] \tag{2.9}$$

$$\delta\boldsymbol{\Psi} = \begin{bmatrix} 0 & -\delta\psi & \delta\theta \\ \delta\psi & 0 & -\delta\phi \\ -\delta\theta & \delta\phi & 0 \end{bmatrix} \tag{2.10}$$

Donde ψ es el Yaw, θ el Pitch y ϕ el Roll.

A medida que el intervalo de tiempo se acerca a cero, el orden de las rotaciones resulta despreciable. Aplicando el límite:

$$\lim_{\delta t \to 0} \frac{\delta\boldsymbol{\Psi}}{\delta t} = \boldsymbol{\Omega}_{ES}^S \tag{2.11}$$

Y reemplazando (2.9) en la ecuación (2.8) y aplicando el límite de $\delta t \to 0$, resulta:

$$\dot{\boldsymbol{C}}_S^E = \boldsymbol{C}_S^E \; \lim_{\delta t \to 0} \frac{\delta\boldsymbol{\Psi}}{\delta t} \tag{2.12}$$

$$\dot{\boldsymbol{C}}_S^E = \boldsymbol{C}_S^E \; \boldsymbol{\Omega}_{ES}^S \tag{2.13}$$

Donde:

$$\boldsymbol{\Omega}_{ES}^S = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.14}$$

Siendo ω_x, ω_y y ω_z las velocidades angulares de Roll, Pitch y Yaw respectivamente y medidas en el sistema de referencia local.

Ángulos de Euler

Por su parte, la representación de ángulos de Euler emplea una matriz de cosenos directores para cada ángulo. El orden de las rotaciones que se opta usualmente se conoce como 3-2-1 siendo roll, pitch y yaw respectivamente. Entonces la transformación entre sistemas de referencia queda:

$$\boldsymbol{C}_E^S = \boldsymbol{C}_3\boldsymbol{C}_2\boldsymbol{C}_1 \tag{2.15}$$

$$\boldsymbol{C}_S^E = (\boldsymbol{C}_E^S)^T = \boldsymbol{C}_3^T\boldsymbol{C}_2^T\boldsymbol{C}_1^T \tag{2.16}$$

$$\boldsymbol{C}_S^E = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.17}$$

$$\boldsymbol{C}_S^E = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{2.18}$$

Para ángulos pequeños, se reduce a:

$$\boldsymbol{C}_S^E \approx \begin{bmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{bmatrix} \tag{2.19}$$

La propagación a lo largo del tiempo en relación a las velocidades angulares de cada eje

queda:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{C}_3 \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{C}_3\boldsymbol{C}_2 \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.20}$$

Quaterniones

Los quaterniones son una extensión de los números complejos a 4 dimensiones:

$$\boldsymbol{q} = a + \boldsymbol{i}b + \boldsymbol{j}c + \boldsymbol{k}d \tag{2.21}$$

El complejo conjugado:

$$\boldsymbol{q}^* = a - \boldsymbol{i}b - \boldsymbol{j}c - \boldsymbol{k}d \tag{2.22}$$

Valen las reglas conocidas:

$$\boldsymbol{i} \cdot \boldsymbol{i} = -1 \quad ; \quad \boldsymbol{i} \cdot \boldsymbol{j} = \boldsymbol{k} \quad ; \quad \boldsymbol{j} \cdot \boldsymbol{i} = -\boldsymbol{k} \tag{2.23}$$

Esto implica que la multiplicación de dos quaterniones no es conmutativa:

$$\boldsymbol{q}_1\boldsymbol{q}_2 \neq \boldsymbol{q}_2\boldsymbol{q}_1 \tag{2.24}$$

El módulo del quaternion está dado por:

$$\boldsymbol{q}\boldsymbol{q}^* = \boldsymbol{q}^*\boldsymbol{q} = a^2 + b^2 + c^2 + d^2 \tag{2.25}$$

Los números complejos unitarios (con módulo igual a 1) se pueden utilizar para expresar rotaciones en un plano. Es sencillo de ver utilizando la notación exponencial de Euler. Supo-

niendo un número complejo unitario $e^{i\theta}$, cuando multiplica otro número complejo $z = \rho e^{i\phi}$, lo rota un ángulo θ:

$$w = e^{i\theta} z = e^{i\theta} \rho e^{i\phi} = \rho e^{i(\theta+\phi)} \tag{2.26}$$

Utilizando notación cartesiana, se define el quaternion unitario en base a las componentes μ_x, μ_y y μ_z de un vector $\boldsymbol{\mu}$, cuyo módulo μ es el ángulo de rotación:

$$\boldsymbol{q} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \cos(\mu/2) \\ (\mu/\mu_x)\ \sin(\mu/2) \\ (\mu/\mu_y)\ \sin(\mu/2) \\ (\mu/\mu_z)\ \sin(\mu/2) \end{bmatrix} \tag{2.27}$$

El mismo satisface la condición:

$$(\mu/\mu_x)^2 + (\mu/\mu_y)^2 + (\mu/\mu_z)^2 = 1 \tag{2.28}$$

Y por tener norma unitaria, también:

$$a^2 + b^2 + c^2 + d^2 = 1 \tag{2.29}$$

La expresión análoga a la ecuación 2.5 es:

$$\boldsymbol{r}^{E'} = \boldsymbol{q}_S^E\ \boldsymbol{r}^{S'}\ \left(\boldsymbol{q}_S^E\right)^* \tag{2.30}$$

Donde:

$$\boldsymbol{r}^S = \boldsymbol{i}x + \boldsymbol{j}y + \boldsymbol{k}z \tag{2.31}$$

$$\boldsymbol{r}^{S'} = 0 + \boldsymbol{i}x + \boldsymbol{j}y + \boldsymbol{k}z \tag{2.32}$$

Para transformar un quaternion a una matriz de rotación, se utiliza la siguiente expresión [64]:

$$\boldsymbol{C}_S^E = \begin{bmatrix} 2a^2 - 1 + 2b^2 & 2(bc + ad) & 2(bd - ac) \\ 2(bc - ad) & 2a^2 - 1 + 2c^2 & 2(cd + ab) \\ 2(bd + ac) & 2(cd - ab) & 2a^2 - 1 + 2d^2 \end{bmatrix} \tag{2.33}$$

Basada en estas representaciones en angulos de Euler:

$$Yaw: \quad \psi = Atan2(2bc - 2ad,\ 2a^2 + 2b^2 - 1) \tag{2.34}$$

$$Pitch: \quad \theta = -sin^{-1}(2bd + 2ac) \tag{2.35}$$

$$Roll: \quad \phi = Atan2(2cd - 2ab,\ 2a^2 + 2d^2 - 1) \tag{2.36}$$

Por su parte, teniendo en cuenta la condición 2.29, la transformación de matriz de rotación (eq. 2.6) a quaternion es:

$$a = \frac{1}{2}\sqrt{1 + c_{11} + c_{22} + c_{33}} \tag{2.37}$$

$$b = \frac{1}{4a}(c_{23} - c_{32}) \tag{2.38}$$

$$c = \frac{1}{4a}(c_{31} - c_{13}) \tag{2.39}$$

$$d = \frac{1}{4a}(c_{12} - c_{21}) \tag{2.40}$$

La propagación de los quaterniones a lo largo del tiempo se expresa en función de la ecuación 2.13 [65]:

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \dot{a} \\ \dot{b} \\ \dot{c} \\ \dot{d} \end{bmatrix} = 0{,}5 \begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix} \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.41}$$

Comparación entre representación de rotaciones

Las representaciones tridimensionales resultan en ecuaciones de movimiento no lineales. De forma opuesta, utilizar representaciones de mayor dimensionalidad, como la matriz de rotación o los quaterniones, resulta en ecuaciones de movimiento lineales.

Los ángulos de Euler presentan singularidades conocidas como "gimbal-lock". Se da cuando el ángulo de cabeceo es de $\pm 90^{\circ}$. Esa indeterminación provoca la pérdida de un grado de libertad. Es decir que dos ejes se alínean, provocando que no se puedan distinguir las rotaciones entre ellos. Remitiendo a la figura 2.25, si el cardan intermedio rota 90° en cualquier sentido, queda alineado con el cardán exterior.

Además, la propagación a lo largo del tiempo de los ángulos de Euler implica el uso de funciones trigonométricas, lo cual hace que el costo computacional sea mayor frente al resto de representaciones.

Por su parte, la propagación de la matriz de rotación (cosenos directores) debe mantenerla ortonormal. Lo cual, debido a errores en el cálculo computacional no se puede garantizar y se debe recurrir a otros métodos para recuperar tal propiedad.

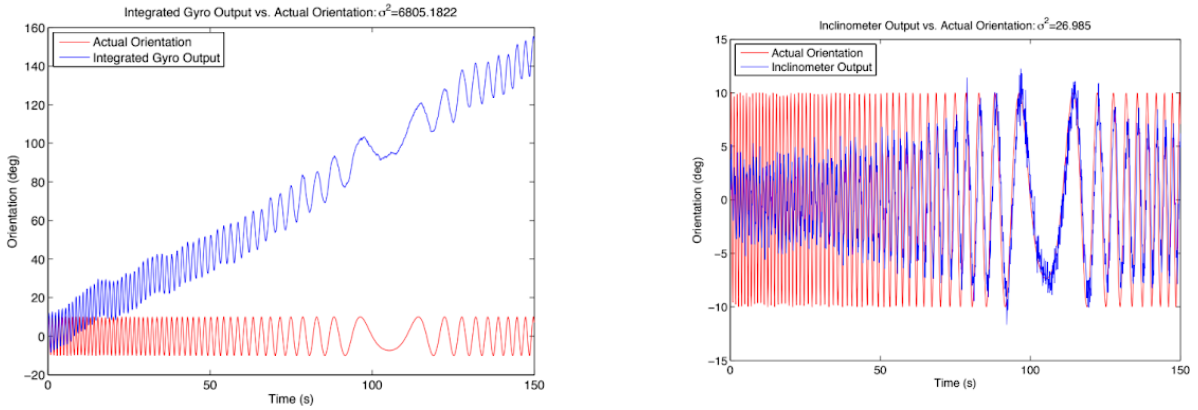
Los errores producto de la integración son menores para los quateriones que para la matriz de rotación. Además requieren menor cantidad de operaciones de punto flotante, por lo que la velocidad de cómputo es mayor [66].

2.3.6. Procesamiento de señales

En un escenario cuasi-estático se puede considerar que la aceleración medida por el acelerómetro es igual a la aceleración de gravedad. En ese caso, basta con un acelerómetro tri-axial para determinar la orientación. Desafortunadamente, en la mayoría de las aplicaciones (laboratorio de marcha incluido) suele haber aceleraciones significativas. Entonces es necesario

procesar la señal para poder calcular la pose del acelerómetro en el sistema de referencia global.

Un giróscopo tri-axial también sería una solución para el problema de conocer la orientación de la IMU. El método es directo: integrar la velocidad angular para obtener la rotación en cada eje. Pero los giróscopos MEMS suelen tener un error de compensación (de aquí en adelante más llamado offset bias) relativamente alto. Entonces, al integrar la señal, ese error constante se acumula a lo largo del tiempo causando deriva en la estimación de la orientación.



a b

Figura 2.27: Simulación de la respuesta de un giróscopo (a) y un acelerómetro (b) a una señal "chirp". La orientación calculada con el giróscopo estima las fluctuaciones pero deriva en el tiempo. Por su parte, la inclinación del acelerómetro es certera cuando el movimiento es de baja frecuencia. Imagen de [67].

En consecuencia, ninguno de los dos sensores resulta suficiente por sí solo. Por lo tanto, en la práctica se utilizan algoritmos de fusión ("sensor fusion") para combinarlos. Los más utilizados son filtros complementarios y filtros Bayesianos. Estos últimos en forma de filtro Kalman o sus derivaciones.

Filtros complementarios

Los filtros complementarios brindan una opción para combinar mediciones de posición de bajo ancho de banda con mediciones de velocidad de alto ancho de banda para sistemas cinemáticos de primer orden [68]. En este caso, la orientación calculada en base al acelerómetro es la señal de baja frecuencia. Por su parte, la velocidad angular capturada con el giróscopo es la señal de alta frecuencia. Entonces, la implementación se realiza aplicando un filtro pasa

altos a la velocidad angular integrada y un filtro pasa bajos a la orientación en función del acelerómetro [67]. La expresión general es de la forma:

$$\hat{x} = F_1(s)y_x + F_2(s)\frac{y_u}{s} \tag{2.42}$$

Donde y_x es la estimación de la orientación en base al acelerómetro, y_u es la velocidad angular capturada con el giróscopo, $F_1(s)$ es la transferencia del pasa bajos, $F_2(s)$ es la transferencia del pasa altos y $\hat{x}$ es la estimación final de la orientación.

El filtro es complementario siempre y cuando se cumpla que:

$$F_1(s) + F_2(s) = 1 \tag{2.43}$$

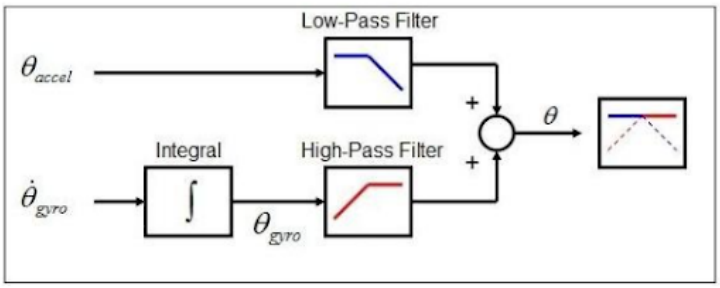


Figura 2.28: Diagrama de un filtro complementario. Las entradas son las predicciones de la orientación según el acelerómetro y la velocidad angular medida con el giróscopo. Luego se integra la velocidad angular y se aplica un filtro pasa bajos o un pasa altos a las señales respectivamente. Imagen de [69].

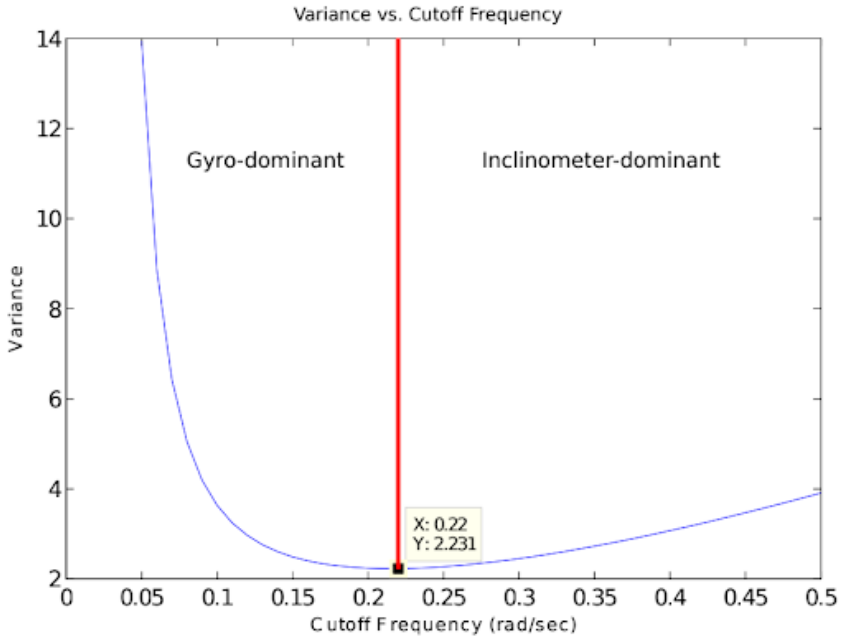


Figura 2.29: Gráfico de Jensen et. al. [67] mostrando la varianza del error para una serie de simulaciones variando la frecuencia de corte de un filtro complementario de segundo orden.

Filtros Bayesianos

Conociendo el proceso físico que describe nuestro sistema (ej.: las ecuaciones de movimiento de un cuerpo rígido), es posible calcular el próximo estado dado el actual. En la práctica, el proceso presenta incertidumbre, la cual se expresa mediante una distribución de probabilidad. En definitiva, se obtiene una estimación del próximo estado [70].

Al mismo tiempo, se pueden utilizar sensores para medir el estado del sistema. Dichas mediciones también poseen incertidumbre, de nuevo, descrita con una distribución de probabilidad [70].

Los filtros Bayesianos, realizan una predicción del estado del sistema y luego la corrigen en función de las observaciones del estado. La implementación de este tipo de filtros se hace en esos dos pasos conocidos como "predicción" y "corrección" [70].

$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1} \tag{2.44}$$

$$bel(x_t) = \eta \; p(z_t|x_t) \; \overline{bel}(x_t) \tag{2.45}$$

Donde $\overline{bel}(x_t)$ es el resultado de la predicción, $bel(x_t)$ es el resultado de la corrección, x es el estado del sistema, u son las variables de control, z son las observaciones, η es un coeficiente de normalización y los subíndices t y $t-1$ indican el instante actual y el instante anterior [70].

Conceptualmente, la predicción genera el espacio de posibles estados y la corrección los pondera de acuerdo a cual se corresponde mejor con la observación del estado [70].

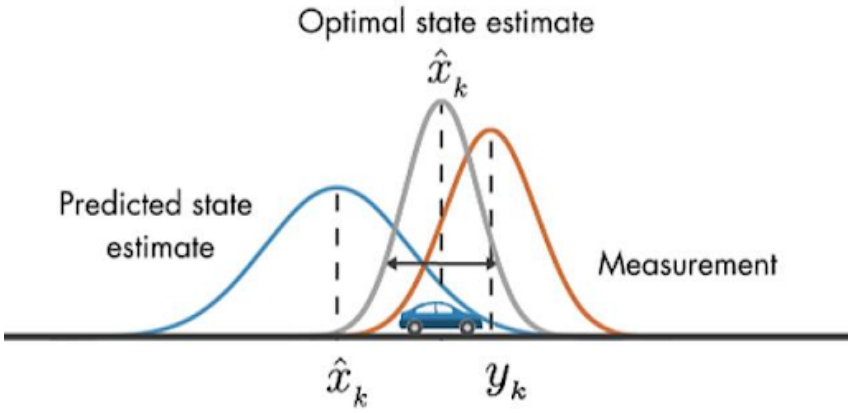


Figura 2.30: Diagrama de la distribución de probabilidad de la posición de un auto según el proceso del sistema (azul), según el proceso de la observación (naranja) y según la predicción final (gris). Imagen de Mathworks.

La implementación del filtro varía según las ecuaciones que modelan el sistema y las distribuciones de probabilidad. La implementación más utilizada para el problema de captura de movimiento es el filtro Kalman. En particular se utiliza para el caso en que los modelos sean lineales (o linealizables) y las distribuciones sean Gaussianas.

2.3.7. Ruido en sensores inerciales MEMS

Como todo sensor, los MEMS no son ajenos al ruido. El mismo consiste de una parte determinística y otra estocástica. La determinística incluye el offset bias, los factores de escala,

la no ortogonalidad de los ejes, la desalineación de ejes, etc., que son removidos de las mediciones a partir de calibraciones. La componente estocástica debe ser modelada como un proceso estocástico, ya que el error que induce es aleatorio y no puede ser removido directamente [71].

A la hora de implementar una aplicación de navegación strapdown con sensores MEMS, los tipos de ruido que más afectan el desempeño del sistema, son: (1) el offset bias, (2) los random walks y (3) los errores de calibración [63, 71]. A continuación se describen y se ve el tipo de perturbación que provocan sobre el cálculo de la orientación, velocidad y posición.

Offset bias

El offset bias es, en principio, una constante, positiva o negativa, que se suma a la señal de interés. Su integral introduce un error lineal y la segunda integral, un error cuadrático.

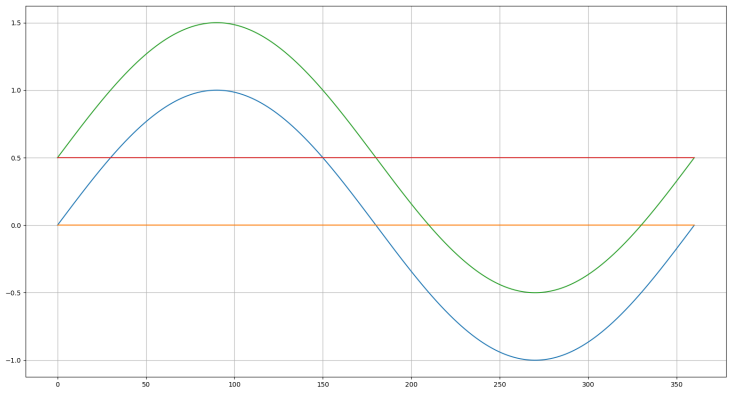


Figura 2.31: Diagrama de una función sinusoidal sin offset (azul) y otra con un offset de 0,5 (verde).

En la práctica, para obtener su valor se toma un promedio de muestras a lo largo de un tiempo prolongado mientras el sensor se encuentra en reposo. En un giróscopo se mide en $^{\circ}/hs$, para un acelerómetro en g/hs o $m/s^2/hs$. Donde $g = 9,8m/s^2/hs$ siendo este el valor de la aceleración de gravedad.

Bias instability

El bias instability es producto del ruido de parpadeo presente en componentes electrónicos y que se observa como fluctuaciones del bias del sensor. Se describe como un proceso de "*random walk*" de primer orden, con media cero y una desviación estándar que crece de manera proporcional a la raíz cuadrada del tiempo [63]:

$$\sigma_\theta(t) = \sigma \cdot \sqrt{\delta t \cdot t} \tag{2.46}$$

La incertidumbre del proceso aumenta con el tiempo, pero, concretamente, la magnitud del bias no puede crecer sin limitación. En consecuencia, el modelo de random walk es válido a corto plazo.

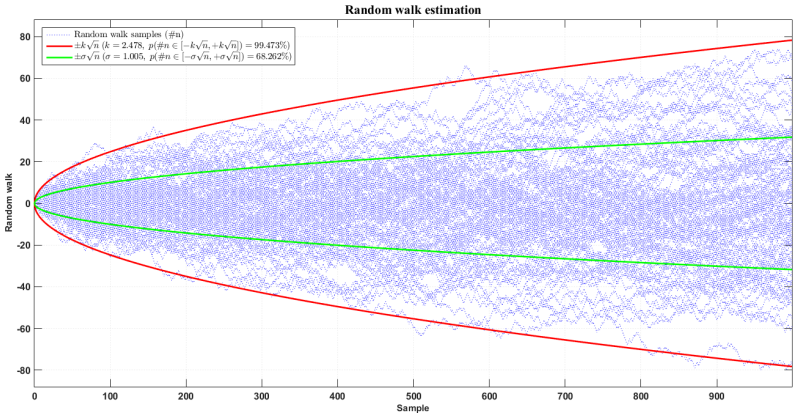


Figura 2.32: Ejemplo de un proceso de random walk en una dimensión. Consiste de una serie de pasos, los cuales tienen tamaño y dirección aleatoria. Imagen de Mathworks.

Random walk de velocidad angular/aceleración

El ruido blanco de las señales de velocidad angular y de aceleración, también se describe como un *random walk* y su desviación estándar sigue la expresión 2.46. Nuevamente, el modelo es válido a corto plazo.

Random walk de ángulo/velocidad

El ángulo y la velocidad se definen como la integral de la velocidad angular y la aceleración respectivamente. En consecuencia, la desviación estándar del ruido es un *random walk* de segundo orden, el cual se expresa de la siguiente manera [63]:

$$\sigma_s(t) \approx \sigma \cdot t^{3/2} \cdot \sqrt{\frac{\delta t}{3}} \tag{2.47}$$

Errores de calibración

Los errores en la sensibilidad de los sensores, la ortogonalidad entre ejes de un mismo sensor y la alineación de esos ejes con los de otros sensores y con el encapsulado, se conocen como errores de calibración. Cuando el sensor MEMs se encuentra rotando o acelerando, la orientación acumula un error que depende de la duración y la magnitud del movimiento [63].

Para calcular los coeficientes del modelo de error, es necesario realizar una calibración. Usualmente consistiendo de una serie de poses y luego una minimización de una función de costo.

Errores en magnetómetros

Los magnetómetros son susceptibles a distorsiones en el campo magnético conocidas como "soft iron" y "hard iron". La primera se debe a materiales ferromagnéticos presentes en el entorno, los cuales no generan un campo magnético propio pero causan una distorsión en el campo magnético presente. El efecto del soft iron sobre el magnetómetro depende de la orientación del sensor. Por su parte, el hard iron es producido por elementos que generen un campo magnético propio (ej.: imanes, parlantes, etc.) [72]. Ese campo magnético se suma al del entorno, generando un error constante que puede ser calibrado de forma similar al bias de los sensores inerciales.

Diseño e implementación

El prototipo a implementar tiene como objetivo la obtención de parámetros espacio temporales a partir de señales de aceleración lineal y velocidad angular de la espalda baja, capturadas por un módulo de sensado con una IMU.

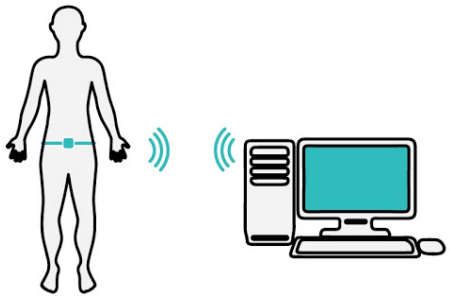


Figura 3.1: Ilustración del sistema implementado.

Las tecnologías y técnicas utilizadas, se escogen en base a:

- Disponibilidad de componentes
- Expertise adquirida
- Objetivos del proyecto
- Tiempo de implementación

Agregado a esto, se pretende que la elección de componentes brinde la oportunidad de futuras mejoras al prototipo y por lo tanto la IMU incluirá un magnetómetro y un sensor de temperatura interno.

En función del conocimiento adquirido sobre sistemas inerciales, el sistema implementado se dividirá en 4 bloques principales (Fig. 3.2):

- Adquisición
- Control
- Comunicación
- Procesamiento

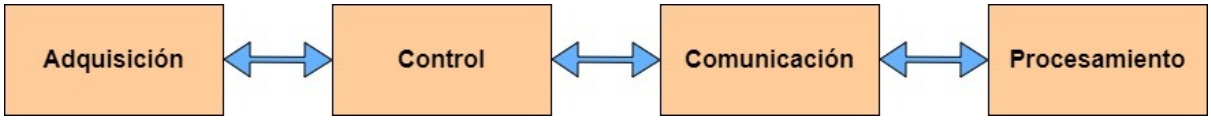


Figura 3.2: Bloques del sistema implementado y la interacción entre ellos.

Priorizando el enfoque sobre el software del sistema y para acelerar el tiempo de implementación, se determinó que el bloque de adquisición brindará la conversión analógica digital de forma resuelta. Por ello, además de la IMU, posee un conversor analógico digital con multiplexor, filtros digitales, filtros analógicos y comunicación por I2C con el módulo de control. Su función principal es la de adquirir las señales de aceleración lineal y velocidad angular cada una en sus 3 dimensiones. También se realiza la conversión analógica digital y un filtrado en función de las características conocidas de las señales.

El módulo de control se compone de un microcontrolador, el cual se utiliza para configurar la adquisición de la señal, la transmisión de datos hacia la computadora y para recibir comandos desde ella.

La comunicación se lleva a cabo por Bluetooth con un módulo en el dispositivo inalámbrico y el hardware nativo de la computadora.

Finalmente, el bloque de procesamiento es el encargado de calcular los parámetros de calibración y de obtener las métricas espaciotemporales.

A continuación se describe cada bloque del sistema por separado. En cada uno se detalla

los requerimientos, la selección de componentes, las características del hardware escogido y la implementación final de cada uno.

El bloque de procesamiento abarca más de un aspecto, por lo que tendrá un análisis más extensivo en la sección 3.5.

3.1. Adquisición

Requerimientos

El bloque de adquisición debe disponer de un acelerómetro triaxial, un giroscopo triaxial, un magnetómetro triaxial y un sensor de temperatura interno, todos en un mismo encapsulado. El chip debe realizar la conversión analógica digital, tener la posibilidad de almacenar los datos en memoria temporalmente y establecer una comunicación con el bloque de control.

Basado en la literatura, productos comerciales y el análisis de bases de datos de señales de IMU (Sección 3.1), se definen los requerimientos para los sensores, el conversor analógico digital y la frecuencia de muestreo mínima:

- Rango de acelerómetro: 8g
- Rango de giróscopo: 500/s
- Rango del magnetómetro: 8 gauss
- Resolución: 12 bits
- Frecuencia de muestreo: 100 Hz

Por otro lado, la comunicación deberá ser I2C, ya que este protocolo requiere sólo dos conexiones [73] en contraste con las tres (o cuatro) del SPI [74].

El chip debe ser de bajo consumo para contribuir a la autonomía del dispositivo.

Selección

En el mercado se encuentran kits de desarrollo que incluyen los sensores digitales a los que se apunta, pero tales soluciones se superponen con el resto de los bloques del sistema. Por lo tanto, se dirigió la búsqueda a encontrar placas de desarrollo sólo con sensores digitales, sin microcontrolador incorporado ni opciones de comunicación inalámbrica.

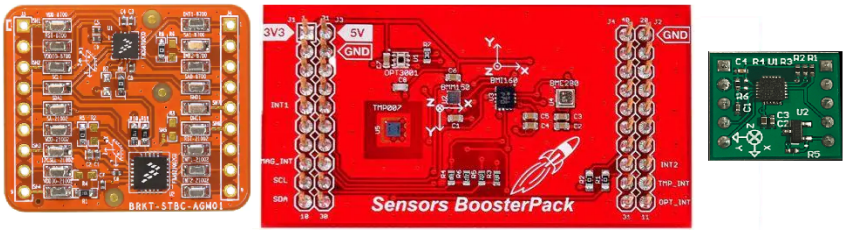


Figura 3.3: Placa de desarrollo BRKT-STBC-AGM01 de NXP (izquierda) presenta el giróscopo separado del resto de los sensores, por lo que no cumple uno de los requerimientos. La solución BOOSTXL-SENSORS de Texas Instruments (centro) tampoco lo cumple, en este caso por el magnetómetro que se encuentra separado del resto. Por su parte, la placa de desarrollo para el chip ICM-20948 de TDK Invensense (derecha) cumple los requerimientos establecidos.

Finalmente se compararon dos placas de desarrollo, una de TDK Invensense con el chip ICM-20948 y la segunda de ST Microelectronics con el chip LSM9DS1. Las dos cumplen los requerimientos establecidos, pero difieren en la tensión de alimentación. El ICM-20948 trabaja a 1,8V, mientras que el LSM9DS1 a 3,3V. Debido a que el microcontrolador utilizado se debe alimentar con 3,3V, se determinó que era conveniente utilizar la solución de ST Microelectronics para facilitar la compatibilidad eléctrica entre estos.

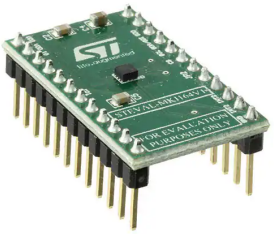


Figura 3.4: El sensor LSM9DS1 de ST Microelectronics cumple con todos los requerimientos necesarios y es la opción elegida frente al ICM-20948, ya que al compartir la misma tensión de trabajo con el microcontrolador, facilita el diseño de la electrónica de alimentación.

Características

El chip posee la capacidad de configurar el rango de cada sensor, ajustando la sensibilidad en valores predefinidos. El giróscopo se puede establecer en 245, 500 o 2000°/s. El acelerómetro en 2, 4, 8 o 16 g. El magnetómetro en 4, 8, 12 o 16 gauss. El conversor analógico digital es de 16 bits.

El LSM9DS1 permite aplicar filtros digitales pasa altos y pasa bajos. También se puede configurar una frecuencia de muestreo y que cada captura sea alertada cambiando el estado de un pin digital.

Por último, el chip consume 600μA con el acelerómetro y el giróscopo encendidos y trabaja con una tensión de 3.3V.

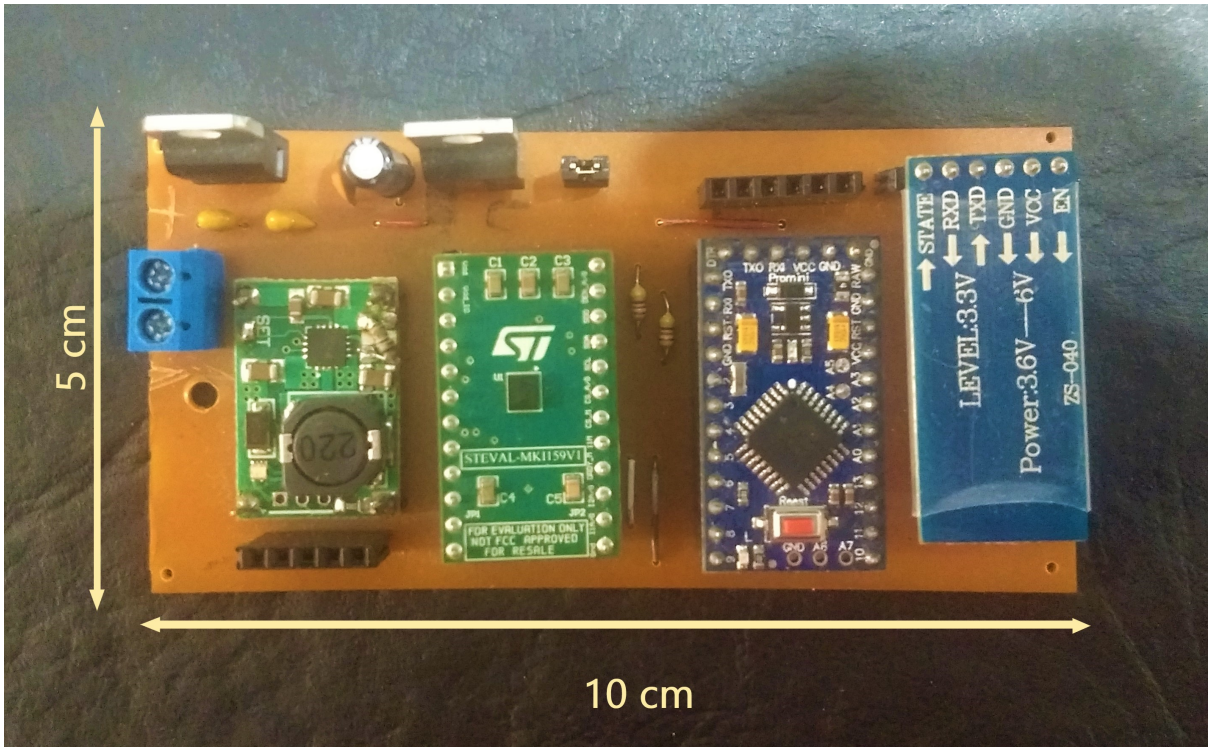


Figura 3.5: Vista superior del PCB construido

	Ancho de banda de aceleración			Ancho de banda de velocidad angular		
	Vertical	Lateral	Antero-posterior	Vertical	Lateral	Antero-posterior
Pelvis	7.20 Hz	6.35 Hz	10.38 Hz	3.55 Hz	24.78 Hz	6.15 Hz
Muslo	8.42 Hz	9.39 Hz	8.56 Hz	7.93 Hz	8.42 Hz	6.71 Hz
Pantorrilla	19.29 Hz	25.27 Hz	24.78 Hz	1.41 Hz	8.42 Hz	6.11 Hz

Tabla 3.1: Anchos de banda obtenidos para cada señal y en cada segmento. Los datos analizados fueron extraídos de [75].

Implementación

Una vez seleccionado el sensor, se prosiguió a definir el preprocesamiento a realizar sobre las señales capturadas. Para ello se realizó un análisis del espectro de señales de marcha, las cuales fueron obtenidas de una base de datos publicada por Hu et. al. [75].

La misma se escogió debido a que los datos no fueron capturados en un entorno de laboratorio altamente controlado, lo que hace más general la información contenida. Cada uno del total de diez individuos, realizó cincuenta repeticiones de un circuito de caminatas en horizontal, caminata en rampa, ascenso y descenso de escaleras, etc.

Para caracterizar el ancho de banda de las señales, primero se extrajeron los segmentos en que los individuos estaban caminando en horizontal. Luego se tomaron 2 enfoques distintos, el primero consistió en concatenar cada segmento, calcular la densidad espectral de potencia (PSD) del total y promediar todos los resultados. El segundo fue calcular la PSD de cada segmento por separado. Finalmente, el ancho de banda se determinó como el 95 % del área bajo la curva de PSD. Rescatando los valores máximos entre los dos enfoques, se obtuvieron los siguientes anchos de banda:

En consecuencia, se configuran los filtros pasa bajos del LSM9DS1 lo más cercano posible a los mayores anchos de banda de cada tipo de señal. Concretamente, para el acelerómetro se define una frecuencia de corte de 26.4 Hz y para el giróscopo 29 Hz.

Para poder hacer uso de los filtros digitales del chip en esas frecuencias de corte, la frecuencia de muestreo se configuró en 238 Hz.

Por otro lado, el periférico de comunicación se configura para operar con I2C. Para configurar la velocidad de transmisión (línea de clock), primero se deben realizar cálculos con respecto al tiempo entre muestras y el tamaño de cada muestra.

El protocolo I2C introduce un bit de confirmación por cada byte enviado o recibido, un bit inicio y otro bit de fin de comunicación. Además, previo a solicitar datos al dispositivo esclavo, se debe especificar desde qué registro se comenzará a leer. Esto implica una transmisión de escritura previa a la adquisición de muestras por parte del bloque de control. En total, por cada muestra de sensor (3 ejes) se requieren 84 bits y entonces, para dos sensores, se necesita transmitir 168 bits. Cada muestra debe transmitirse a la computadora antes de que la siguiente muestra sea capturada. La transmisión se hace por puerto serie a 115200 bits/s y para transmitir una muestra completa de dos sensores, se requieren 160 bits.

El tiempo entre muestras es de 4.2ms (1/238 Hz), I2C tarda 1.7ms y el puerto serie 1.4ms, lo que resulta en una holgura admisible de 1.1ms.

Si se mantiene la misma frecuencia de muestreo y la misma velocidad de transmisión hacia la computadora, para incorporar los datos del magnetómetro, se debe aumentar el baud rate del I2C a 400 kbits/s (modo rápido).

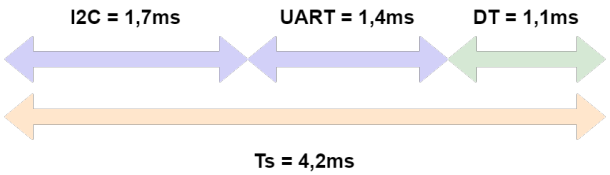


Figura 3.6: Tiempo de duración de cada paso en la transmisión de datos. DT referencia el tiempo de holgura y Ts el período de muestreo.

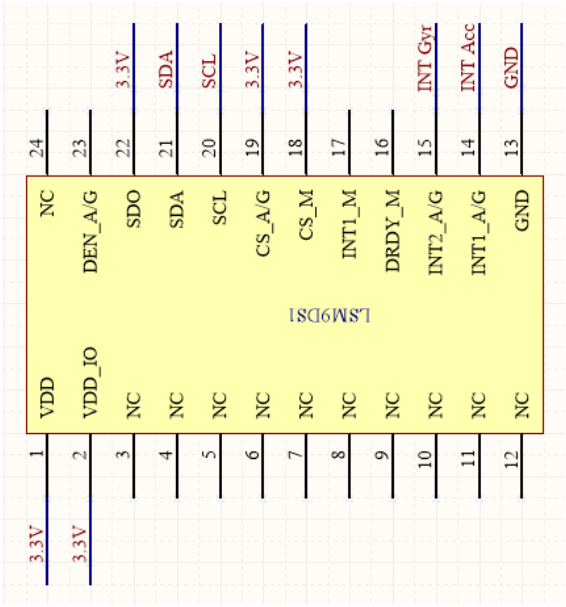


Figura 3.7: Esquemático del sensor LSM9DS1 y sus conexiones.

3.2. Control

Requerimientos

El bloque de control debe tener capacidad de comunicación UART capaz de operar con un baud rate mínimo de 115200 bits/s. Además, I2C en configuración normal (100 kbits/s) y rápida (400 kbits/s), esta última para facilitar la inclusión del magnetómetro en un futuro.

Por otro lado, para hacer uso de la operación en modo contínuo del LSM9DS1, es necesario contar con pines digitales de entrada de donde capturar los eventos.

También debe funcionar con 3.3V de alimentación y ser de bajo consumo para aumentar la autonomía del dispositivo inalámbrico.

Selección

La elección del microcontrolador fue influenciada por el conocimiento de los microcontroladores Atmel presentes en las placas Arduino. Se escogió el Arduino Pro Mini ya que es la placa de menor tamaño y con el microcontrolador ATmega328P, el cual es de bajo consumo, posee los pines digitales y periféricos de comunicación requeridos.

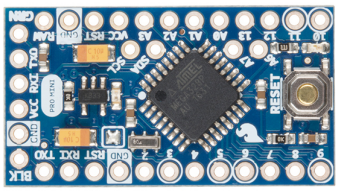


Figura 3.8: Arduino Pro Mini con un microcontrolador ATmega328P y un cristal de 8MHz.

Características

El Atmega328P posee 3 puertos con pines digitales de entrada y salida. Por el lado de los periféricos de comunicación, tiene un puerto UART, una interfaz I2C y un puerto serie SPI. Su tensión de trabajo es de 3.3V y opera a una frecuencia de clock de 8MHz con un consumo típico de 5.2mA.

Implementación

El entorno de programación es en lenguaje C utilizando el IDE de Arduino. Dicho esto, el objetivo del programa es hacer las veces de intermediario entre la computadora y el sensor. El microcontrolador tiene 2 estados posibles: standby o adquisición. El puerto UART dispara una interrupción cada vez que recibe datos. En ella, se verifica que la información recibida sea válida y modifica el estado del microcontrolador. Si el nuevo estado es el de adquisición,

el microcontrolador comienza a hacer un "polling" de dos pines digitales. Estos pines se encuentran conectados al LSM9DS1 e indican cuándo se han capturado muestras del giróscopo y del acelerómetro respectivamente. Una vez que se finalizó la lectura de los registros del sensor, la muestra obtenida se envía hacia el ordenador. Este proceso se repite hasta que se reciba un mensaje de fin de adquisición.

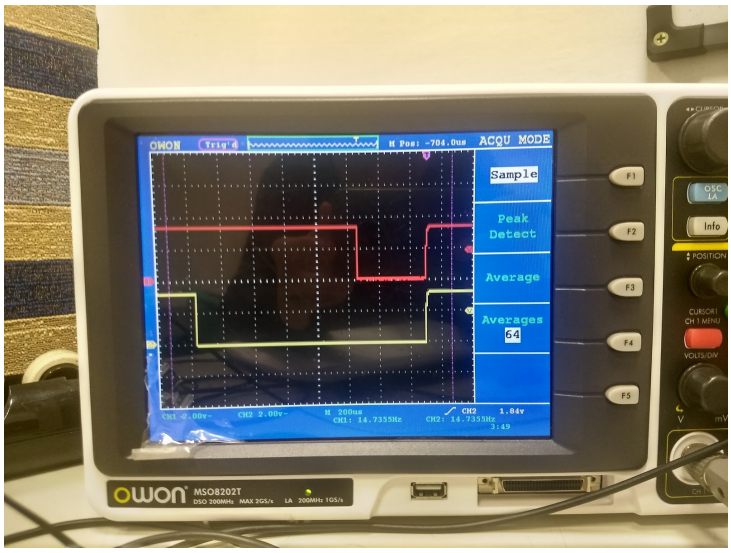


Figura 3.9: Señales de los pines digitales del LSM9DS1. Al escribirse los registros de los sensores con una nueva muestra, la línea asociada se pone baja. Al momento en que las dos muestras han sido escritas en memoria, el microcontrolador las lee. La línea amarilla es la tensión de entrada del pin asociado al giróscopo y la línea roja es la tensión asociada al acelerómetro.

La configuración del sensor se realiza previo a iniciar el loop principal del programa. Se escriben una serie de registros de memoria que inicializan cada sensor, definen la frecuencia de muestreo en 238 Hz, activan los filtros digitales y ubican las frecuencia de corte según lo descrito anteriormente (Tabla 3.1). También se define la velocidad del bus I2C en 100 kbits/s.

Adicionalmente se utilizaron resistencias de pull-up necesarias en las líneas del I2C (Figura 3.11).

En cuanto a la comunicación por USART, el módulo bluetooth se configuró mediante comandos AT [76] para trabajar con un baud rate de 115200 bits/s.

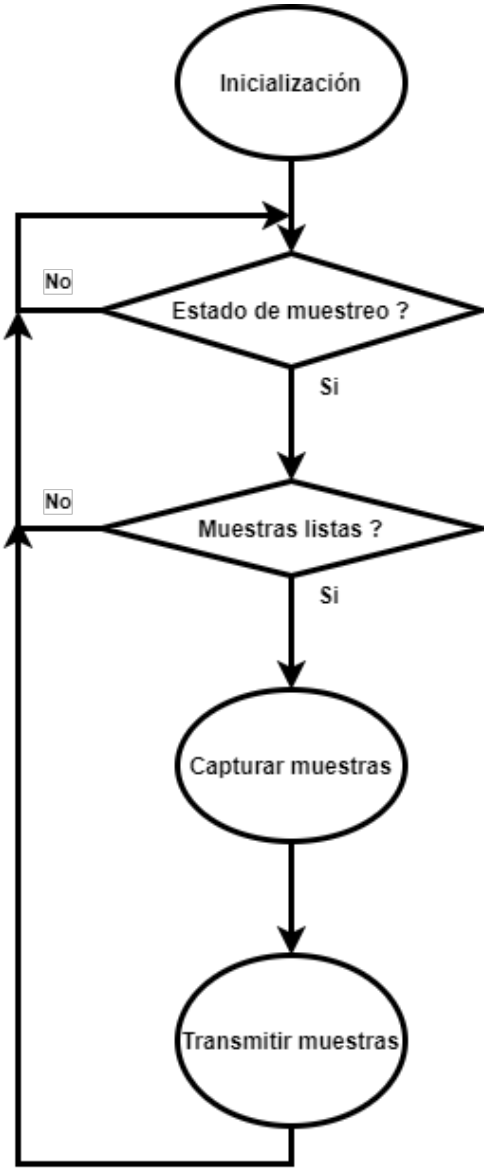


Figura 3.10: Flujo del código escrito para el Arduino Pro Mini (apéndice A).

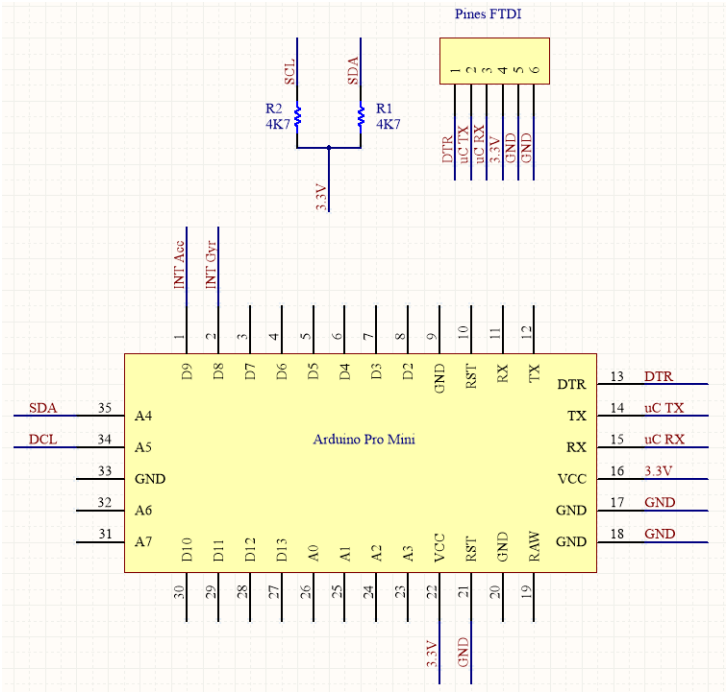


Figura 3.11: Esquemático del Arduino Pro Mini, las resistencias de pull-up del bus I2C y los pines de salida para programar el microcontrolador con un chip FTDI.

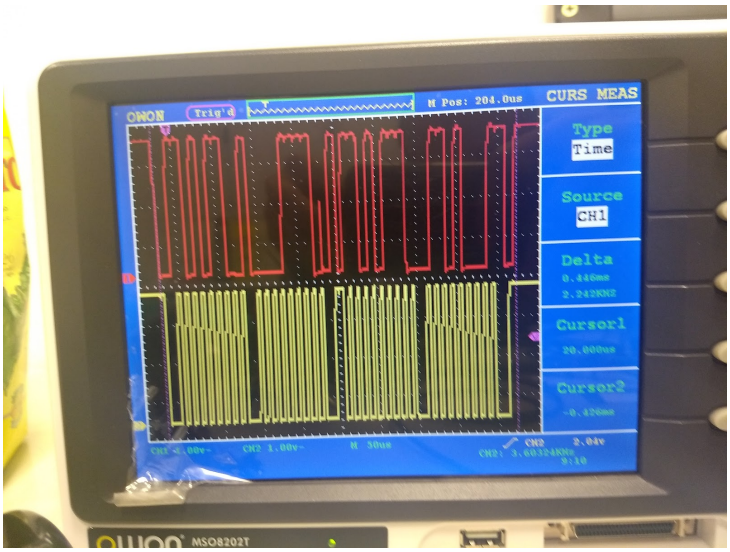


Figura 3.12: Imagen de las señales de clock (amarillo) y de datos (rojo) del bus I2C del dispositivo vistas en un osciloscopio. Las señales son una demostración de una lectura de un registro del sensor. En la señal de clock se puede distinguir la duración de 4 mensajes. El primero es un llamado al esclavo, indicando que se realizará una escritura. El segundo es dicha escritura, la cual contiene la dirección de memoria de un registro de interés para indicar al sensor que dirija un puntero hacia esa dirección. El tercero es un nuevo llamado al esclavo indicando que la transacción será de lectura. Finalmente, el cuarto mensaje es el esclavo enviando el valor del registro indicado en el segundo mensaje. Luego de cada byte transmitido, hay un bit de "acknowledge" que indica si hubo algún problema o no. Para leer registros consecutivos, el maestro debe mantener la línea activa hasta que se hayan capturado los bytes esperados.

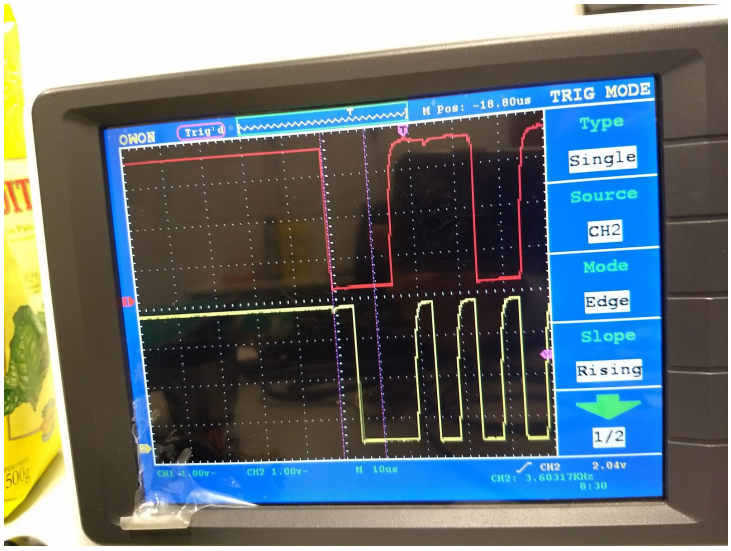


Figura 3.13: Imagen de las señales de clock (amarillo) y de datos (rojo) del bus I2C del dispositivo vistas en un osciloscopio. El maestro activa la línea llevando la línea de datos a cero mientras el clock sigue alto.

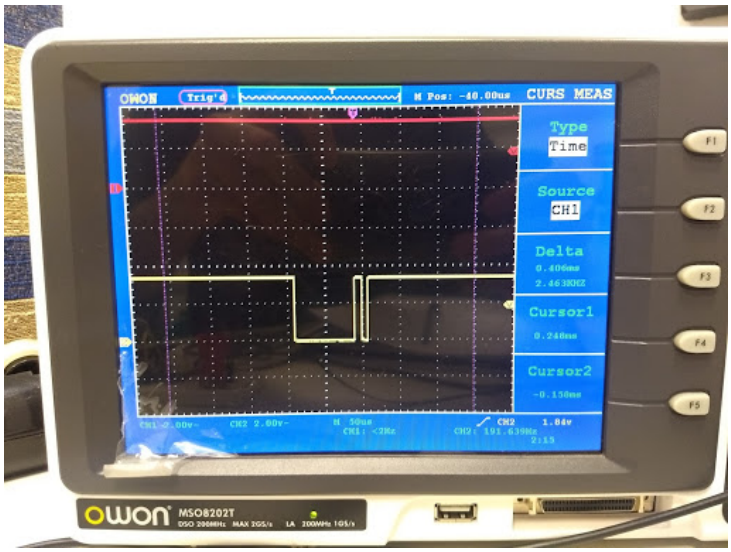


Figura 3.14: Imágen de la señal de transmisión del microcontrolador por el puerto USART (amarillo). Primero se envía un byte en 0x00 y luego 0xFF. La transmisión comienza con un bit bajo, seguido del byte transmitido y finaliza con un bit alto de stop.

3.3. Comunicación

Requerimientos

La comunicación inalámbrica elegida es el protocolo Bluetooth. El mismo puede operar en la banda de frecuencia ISM (industrial scientific & medical) de 2.4GHz, apta para dispositivos

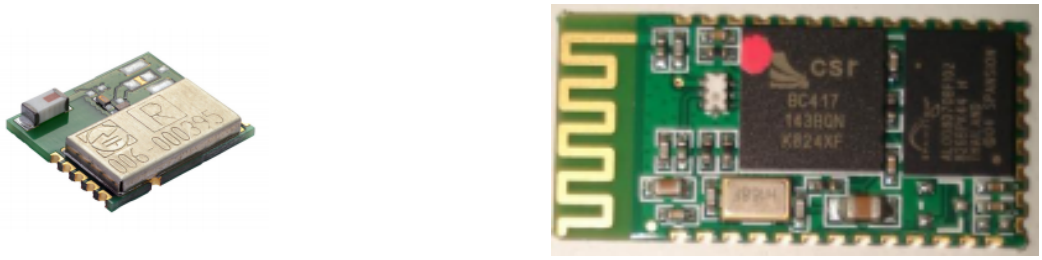
médicos. Además no necesita ningún tipo de router cercano para mantener la conexión con una computadora.

Considerando el ambiente de laboratorio de marcha en donde el individuo debe recorrer una senda de 7 m y el observador se encuentre a 5 m de la misma, el alcance mínimo de la transmisión se define en 10 m.

Selección

Primero se barajó la posibilidad de usar el chip SPBTLE-RF de ST Microelectronics debido a su pequeño tamaño, bajo consumo y capacidad para comunicarse por SPI. La desventaja de esta clase de chip es que para utilizarlos se requiere desarrollar una ACI (Application controller interface) con comandos HCI (host controller interface) [77] para configurar el dispositivo.

Dado el tiempo de desarrollo y que la implementación de una ACI no forma parte de los alcances del proyecto, se optó para utilizar un IC con esa funcionalidad integrada.



a b

Figura 3.15: Chip SPBTLE-RF (a). El módulo HC-05 (b) con el chip BC417 de Cambridge Silicon Radio (Qualcomm).

Se decidió utilizar el módulo HC-05 ya que brinda la posibilidad de configurarlo con comandos AT [76] código a través del puerto USART. Al momento de adquirir el módulo, se encontraba disponible el circuito integrado FC-114 que incluye el módulo HC-05, un regulador de tensión, LEDs y un pulsador que permite interactuar manualmente con el módulo.

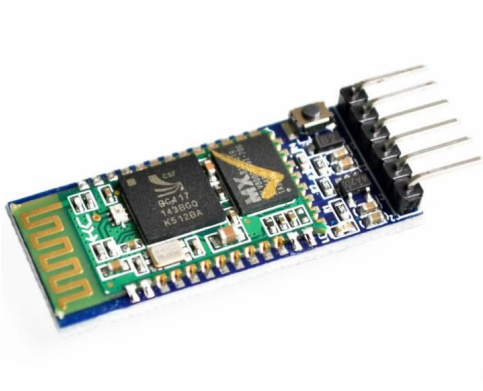


Figura 3.16: Placa FC-114 con el módulo HC-05 y otros accesorios.

Características

El módulo FC-114 puede configurarse con comandos AT. Tiene unas dimensiones de 44x15mm. Trabaja a 5V y el máximo de corriente que alcanza a consumir es de 40mA durante el "pairing" con otros dispositivos. Su frecuencia de transmisión es de 2.4GHz y es compatible con Bluetooth 2.0.

Implementación

El módulo se configuró mediante comandos AT para oprear con un baud rate de 115200 kbits/seg según lo descripto en la sección 3.2. Cabe aclarar que las líneas de transmisión del puerto UART entre el microcontrolador y el FC-114 no requieren resistencias de pull-up adicionales ya que los periféricos de cada chip lo resuelven [78].

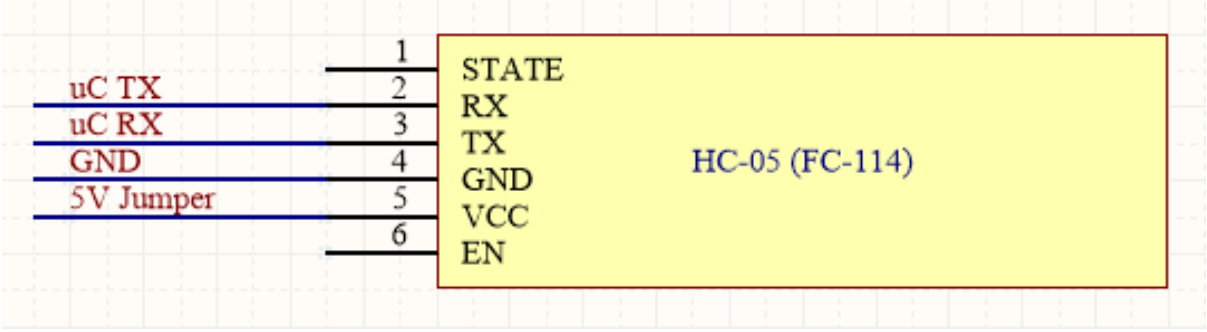


Figura 3.17: Esquemático del HC-05 y sus conexiones.

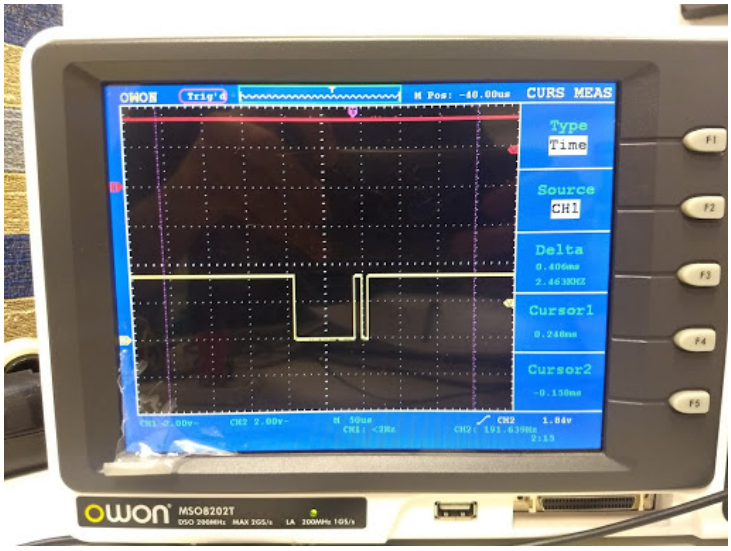


Figura 3.18: Transmisión por el bus USART. El bit de start es bajo, el de stop es alto y no hay bit de paridad. Los tratos transmitidos al momento de tomar la imagen fueron dos bytes seguidos, el primero con valor cero y el segundo con valor 255.

3.4. Alimentación eléctrica

Requerimientos

El único requerimiento es que el dispositivo sea autónomo por una hora, tiempo en el que se pueden evaluar al menos dos pacientes complejos.

Selección

Para alimentar al microcontrolador y al sensor es necesario una línea de 3.3V, mientras que el bloque de comunicación requiere 5V. Por lo tanto es necesario una fuente que supere los 5V y dos reguladores de tensión para ajustarla, uno para cada nivel requerido.

Las fuentes de tensión más pequeñas y con mayor autonomía encontradas fueron baterías del tipo Li-Po de 3.7V. Se utilizaron dos celdas en serie para alcanzar los 7.4V y superar los 5V

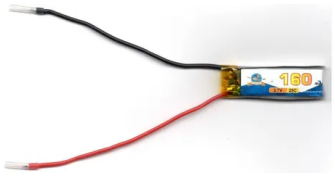


Figura 3.19: Batería Li-Po escogida.

Para recargar la batería LiPo es necesario utilizar un circuito integrado que regule la corriente de carga y asegure que las baterías no excedan límites superiores e inferiores de tensión. Para administrar dos celdas se encontró un módulo con el chip TP5100 que además posee protección contra cortocircuito.

Figura 3.20: Módulo con el chip TP5100 y el circuito necesario para su funcionamiento.

Características

Las baterías escogidas proporcionan una tensión de 3.7V cada una y 160mAh.

El chip TP5100 puede entregar desde 100mA de corriente hasta 2A y además puede configurarse para cargar una o dos celdas. Posee dos leds que indican cuando la batería se está cargando y cuando finalizó su carga. La tensión límite de carga es de 8.4V para dos celdas [79].

Implementación

Para alimentar el bloque de comunicación, se introdujo el regulador de tensión LM7805 que toma los 7.4V de la batería y entrega 5V [80]. Luego, en cascada, un LD1117V33 para pasar de 5V a 3.3V [81] y suplir al microcontrolador y al sensor.

Además, se tuvo que soldar dos pads en el módulo del TP5100 para activar la carga de dos celdas y se reemplazó una resistencia por una de 1 para regular la corriente de carga a 100mA [79].

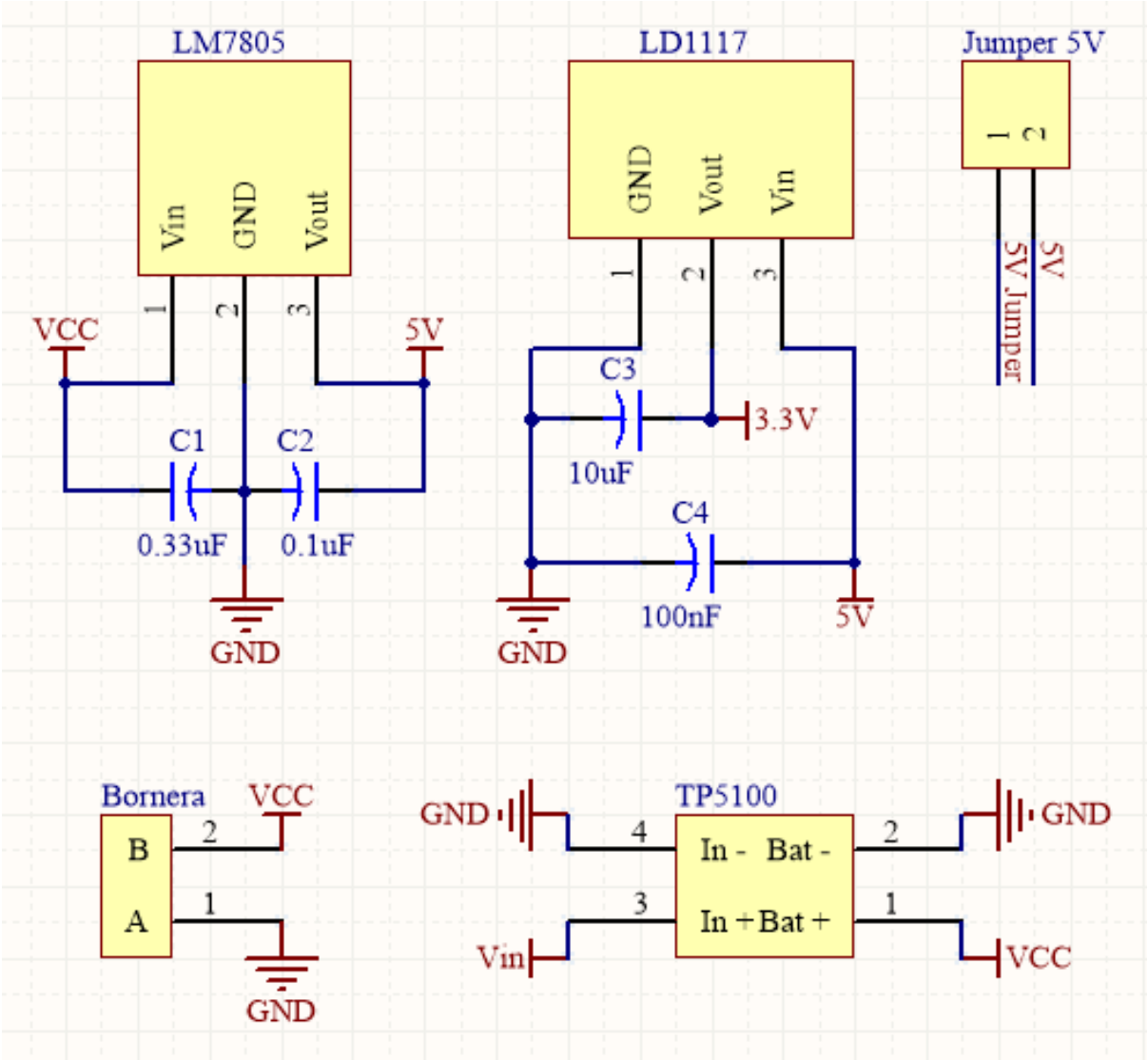


Figura 3.21: Esquemático de todos los elementos del bloque de alimentación. El "Jumper 5V" se agregó para conectar y desconectar el módulo Bluetooth y así poder liberar el puerto UART para programar el microcontrolador sin quitarlo del PCB.

3.5. Procesamiento

El bloque de procesamiento abarca 4 aspectos del proyecto:

- La caracterización de ruido de los sensores (sección 3.5.1)
- La calibración de la IMU (sección 3.5.2)
- La obtención de la orientación en sistema de referencia global (3.5.3)
- El cálculo de parámetros espacio-temporales (sección 3.5.4)

La caracterización de ruido se lleva a cabo con un análisis de varianza Allan. Este paso ayuda a entender el comportamiento del sensor, ajustar coeficientes en los algoritmos de procesamiento y puede ser útil al momento de explicar los resultados finales.

La calibración es el punto de partida para la fusión de sensores y para obtener los parámetros espacio-temporales. La misma se implementa en base a un método de optimización cuyas funciones de costo se definen a partir de dos premisas, las cuales son válidas durante los momentos en que la IMU se encuentra estática.

La estimación de la orientación representa el componente central del proyecto ya que implementa la fusión de sensores y por lo tanto se encarga de transformar las señales crudas en información confiable para poder sacar provecho de la IMU.

Finalmente, para la obtención de parámetros espacio-temporales, se requiere detectar eventos del ciclo de marcha, para luego realizar los cálculos necesarios y presentar el resultado final del dispositivo.

La totalidad del código fue desarrollado en Python3. A continuación se describen los algoritmos y se presentan los resultados del análisis de varianza Allan.

3.5.1. Caracterización de ruido: Varianza Allan

La varianza allan (VA) es una técnica utilizada para describir los tipos de ruido presentes en las señales de sensores inerciales [82, 83, 71, 84, 85, 63]. El resultado del método se visualiza en un gráfico logaritmico del cual se obtiene la varianza para cada ruido presente en los sensores. Los valores se encuentran sobre rectas de pendiente 0 (Figura B.3), $-\frac{1}{2}$ (Figura B.1) y $\frac{1}{2}$ (Figura B.2) para el bias instability (sección 2.3.7), angle random walk (sección 2.3.7) y rate random walk (sección 2.3.7) respectivamente. Los puntos a observar de las rectas se definen en base a la relación de la varianza allan con la PSD (Densidad espectral de potencia).

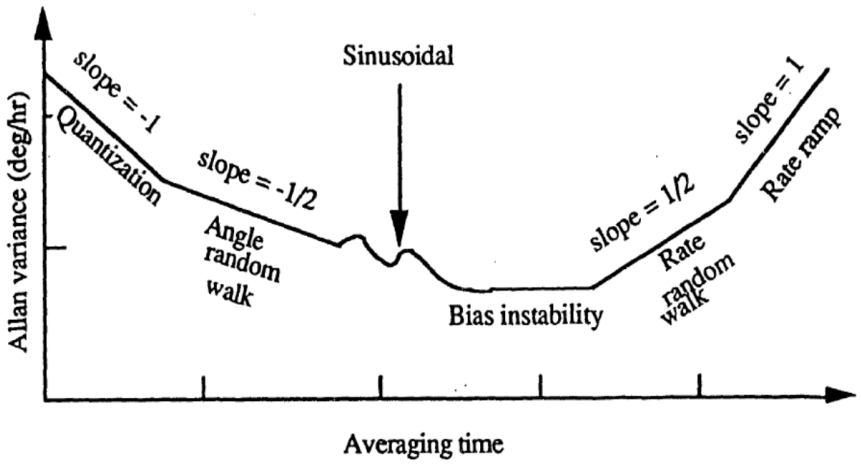


Figura 3.22: Diagrama los de la varianza allan para un giróscopo en un gráfico log-log. En la curva se pueden identificar segmentos con distintas pendientes. Cada una se corresponde con un tipo de ruido.

La definición formal de la VA y su relación con la PSD, se encuentran en el apéndice B. A continuación se muestran los resultados obtenidos para el LSM9DS1 utilizado.

En los casos en que la zona donde ajustar la curva fuese ambigua, se escogió la que resultara en un coeficiente mayor.

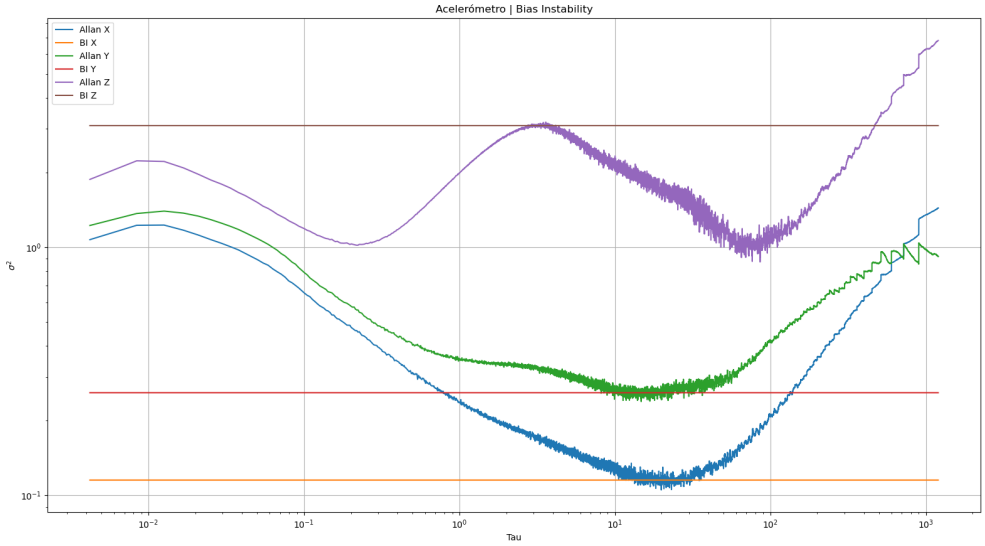


Figura 3.23: Varianza allan de los tres ejes del acelerómetro y las curvas ajustadas con pendiente 0, asociadas al bias instability.

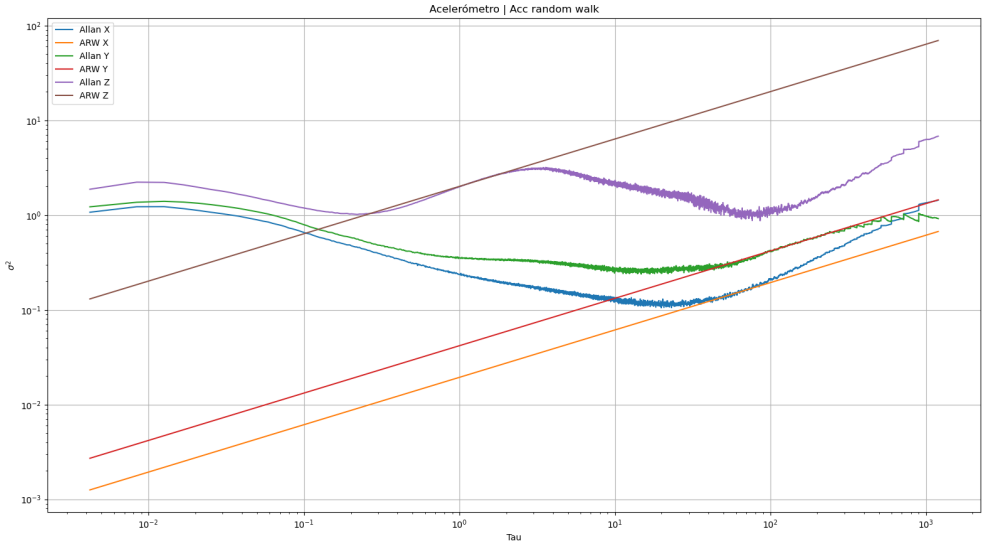


Figura 3.24: Varianza allan de los tres ejes del acelerómetro y las curvas ajustadas con pendiente $\frac{1}{2}$, asociadas al acceleration random walk.

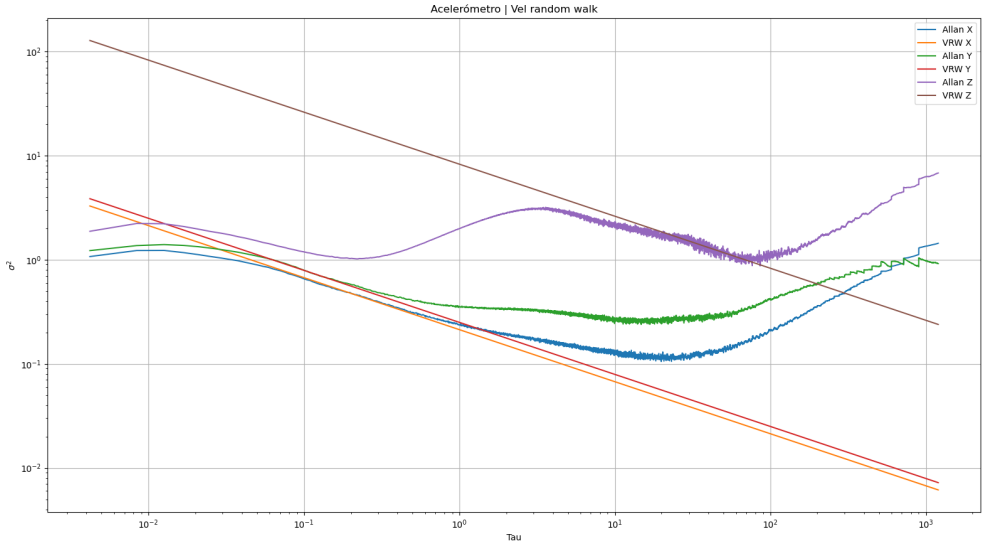


Figura 3.25: Varianza allan de los tres ejes del acelerómetro y las curvas ajustadas con pendiente $-\frac{1}{2}$, asociadas al velocity random walk.

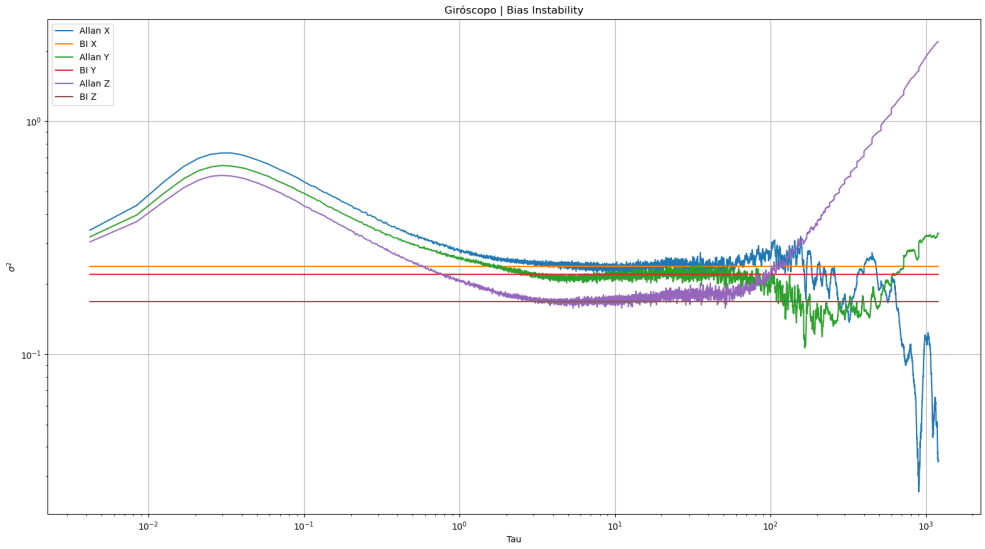


Figura 3.26: Varianza allan de los tres ejes del giróscopo y las curvas ajustadas con pendiente 0, asociadas al bias instability.

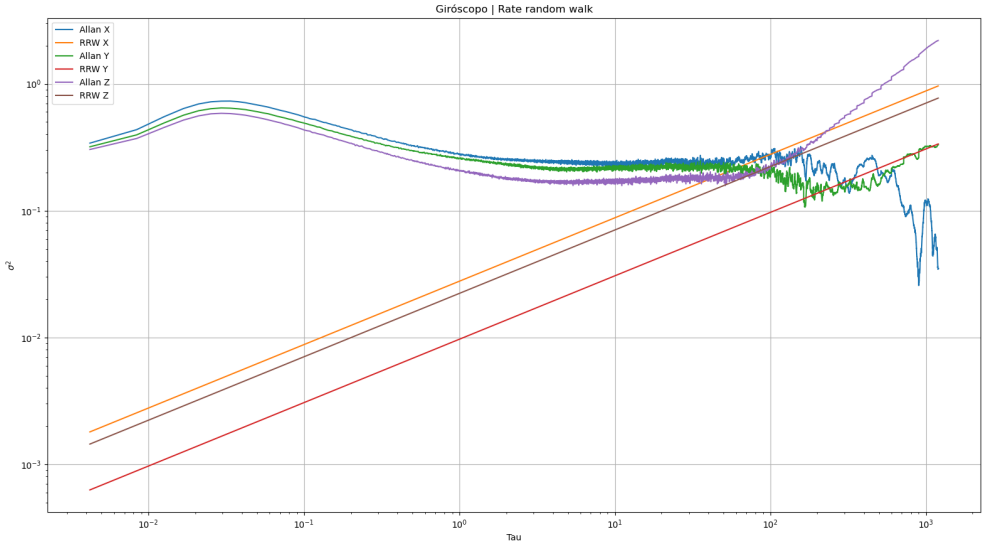


Figura 3.27: Varianza allan de los tres ejes del giróscopo y las curvas ajustadas con pendiente $\frac{1}{2}$, asociadas al rate random walk.

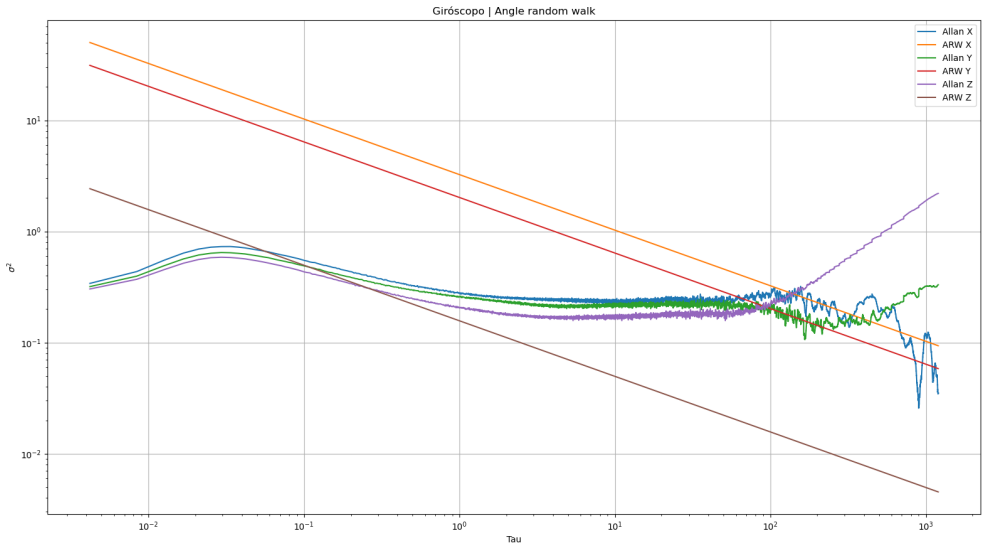


Figura 3.28: Varianza allan de los tres ejes del giróscopo y las curvas ajustadas con pendiente $-\frac{1}{2}$, asociadas al angle random walk.

	Bias instability	RW de ángulo/velocidad	RW de velocidad angular/aceleración
Ax	0.1153	0.2128	0.0336
Ay	0.2593	0.2501	0.0725
Az	3.0913	8.2811	3.4856
Gx	0.2383	3.2475	0.0481
Gy	0.2204	2.0218	0.0168
Gz	0.1684	0.1572	0.0386

Tabla 3.2: Varianzas obtenidas para el sensor utilizado como resultado del análisis por varianza allan. RW significa random walk.

3.5.2. Calibración

El proceso de calibración debe poder ser realizado en el campo de aplicación sin necesidad de equipamiento extra (ver sección 2.2.2), por lo que se determinó utilizar la técnica propuesta por Fong et. al. [53].

El método se basa en 2 premisas que imponen condiciones físicas y matemáticas a las señales capturadas. La primera premisa es que: la magnitud de la aceleración medida con el sensor estático, debe ser igual a la magnitud de la gravedad. La segunda premisa es que: con el sensor estático luego de un desplazamiento y rotación arbitrarios, el vector de gravedad medido con el acelerómetro debe coincidir con el calculado propagando la orientación con el giróscopo.

Cada premisa permite definir una función de costo sobre la cual se lleva a cabo un proceso de optimización. De tal forma se obtienen los parámetros que definen el modelo de error de los sensores.

Concretamente, el procedimiento consta de mover y estacionar la IMU de forma repetitiva la mayor cantidad de veces que sea posible. Esto es para mejorar la performance del algoritmo de optimización. Además, es preferible que los movimientos sean amplios para que los errores

de calibración superen a los procesos de ruido estocásticos.

Es importante determinar los momentos en que el sensor se encuentra estático ya que las premisas deben corroborarse en tales momentos. Además el bias del giróscopo se calcula promediando su señal durante tales períodos. Por tal motivo, esta técnica requiere la implementación de un algoritmo adicional que permita establecer los momentos en que el sensor se encuentra estático. El utilizado en la publicación es el de Saxena et. al. [86] y su implementación se encuentra en el apéndice C.

Para el acelerómetro, su modelo es:

$$h(\boldsymbol{a}_{S,k},\, \boldsymbol{\theta}_a) = \boldsymbol{E}(\boldsymbol{a}_{S,k} - \boldsymbol{b}_a) = \boldsymbol{a}_{P,k} \tag{3.1}$$

Donde $\boldsymbol{a}_{P,k}$ es el valor de la aceleración en el instante k en el sistema de referencia de la plataforma donde está montado, $\boldsymbol{a}_{S,k}$ es el valor de la aceleración en el instante k en el sistema de referencia del sensor, $\boldsymbol{E}$ es la matriz de desalineación de ejes y de factores de escala, $\boldsymbol{b}_a$ es el valor del offset bias y $\boldsymbol{\theta}_a$ es el conjunto de parámetros del modelo definidos por $\boldsymbol{E}$ y $\boldsymbol{b}_a$.

La función de costo para evaluar la primera premisa sólo depende del acelerómetro::

$$\boldsymbol{L}(\boldsymbol{\theta}_a) = \sum_{k=0}^{K-1} \left(1 - \|\boldsymbol{h}(\boldsymbol{a}_{S,k}, \boldsymbol{\theta}_a)\|^2\right)^2 \tag{3.2}$$

Donde K es la cantidad de muestras durante tales períodos. El 1 representa la magnitud de 1g de la gravedad. La función de costo penaliza la diferencia entre ese valor y el modelo del sensor.

Para el caso del giróscopo, previo a realizar el ajuste del modelo, se sustrae el bias de las

mediciones. Por lo tanto el modelo se representa de la siguiente forma:

$$\boldsymbol{w}_{P,k} = \boldsymbol{E}_g \ \ \boldsymbol{w}_{S,k} \tag{3.3}$$

Donde $\boldsymbol{w}_{P,k}$ es la velocidad angular en el sistema de referencia de la plataforma donde se encuentra montado el sensor, $\boldsymbol{w}_{S,k}$ es la velocidad angular en el sistema de referencia del sensor y $\boldsymbol{E}_g$ es la matriz de desalineación de ejes y de factores de escala definida con un conjunto de parámetros $\boldsymbol{\theta}_g$.

La función de costo para la segunda premisa será:

$$\boldsymbol{L}(\boldsymbol{\theta}_g) = \sum_{k=0}^{K-1} \|\boldsymbol{u}_a - \boldsymbol{u}_g(\boldsymbol{\theta}_g)\|^2 \tag{3.4}$$

Donde $\boldsymbol{u}_a$ es el vector gravedad estimado usando el acelerómetro, $\boldsymbol{u}_g$ es el vector gravedad estimado usando el giróscopo.

Para calcular $\boldsymbol{u}_g$, es necesario definir el método para propagar la rotación del dispositivo. Fong et. al. [53] utiliza un algoritmo de segundo orden definido en [65]. Por otro lado, el algoritmo que se utiliza para la optimización es el "Downhill simplex" [87]. La implementación de ambos se encuentra en el apéndice C.

3.5.3. Estimación de la orientación

A partir de lo introducido en la sección 2.3.6, se contempló utilizar un filtro Kalman [88] o el filtro de Madgwick et. al. [89] que es un filtro complementario diseñado específicamente para navegación strapdown. Se optó por este último ya que, según la publicación original, su implementación es más simple y presenta resultados similares al filtro Kalman.

A continuación se describe el filtro Madgwick. La implementación se encuentra en el

apéndice D.

El filtro de Madgwick es un filtro complementario que da como resultado el quaternion que representa la orientación del sensor en el sistema de referencia global. La ecuación 2.42, en este caso, es expresada de la siguiente forma:

$$ {}^{S}_{E}\boldsymbol{q}_{est,t} = \gamma_t \, {}^{S}_{E}\boldsymbol{q}_{\nabla,t} + (1-\gamma_t) \, {}^{S}_{E}\boldsymbol{q}_{\omega,t} \quad ; \quad 0 \leq \gamma_t \leq 1 \tag{3.5} $$

Donde, para el instante de tiempo t, ${}^{S}_{E}\boldsymbol{q}_{est,t}$ es el quaternion estimado, γ_t es el coeficiente del filtro, ${}^{S}_{E}\boldsymbol{q}_{\nabla,t}$ es el quaternion obtenido con el acelerómetro y ${}^{S}_{E}\boldsymbol{q}_{\omega,t}$ es el obtenido con el giróscopo.

Una vez estimado el quaternion, se puede calcular la matriz de rotación o los ángulos de Roll, Pitch y Yaw que describen la orientación de la IMU en el sistema de referencia global E, utilizando las ecuaciones 2.33, 2.36, 2.35 y 2.34.

El método se describe siguiendo el diagrama de la figura 3.29.

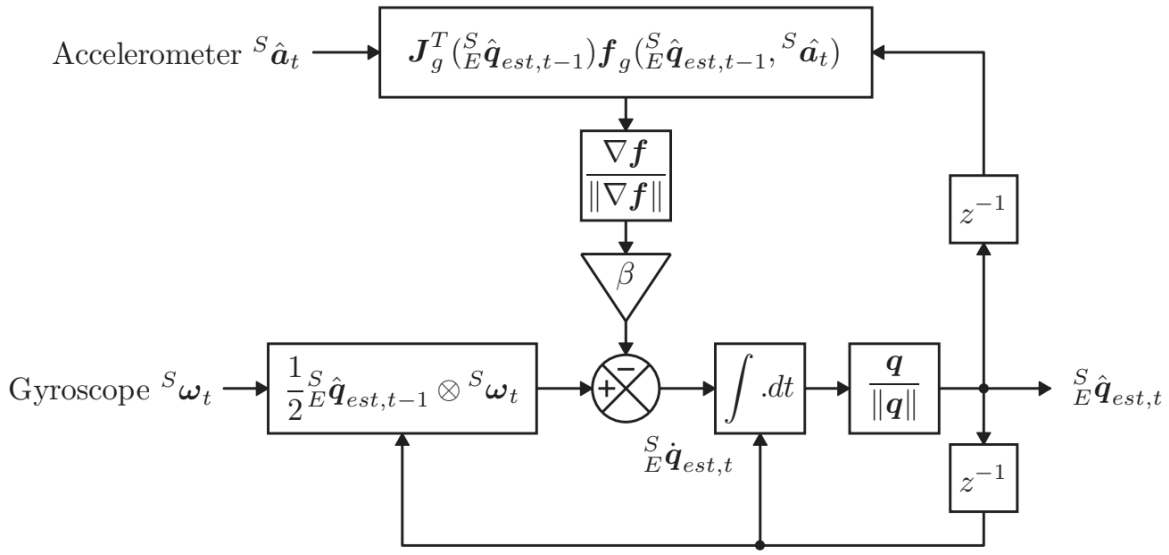


Figura 3.29: Diagrama de la implementación para IMU del filtro Madgwick [89].

La velocidad angular es utilizada para calcular la rapidez de cambio de la orientación. La expresión 3.6 es equivalente a 2.41.

$$ {}^{S}_{E}\dot{\boldsymbol{q}}_{est,t} = \frac{1}{2} \, {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} \otimes \, {}^{S}\omega_t \tag{3.6} $$

Por otro lado, la aceleración es utilizada para estimar la dirección del error del paso anterior. Tal dirección se expresa como el gradiente normalizado $\frac{\nabla f}{\|\nabla f\|}$. Donde f es una función de costo que mide la diferencia entre el eje vertical del sistema de referencia global (paralelo a la gravedad) y la orientación estimada en base al acelerómetro. Cabe aclarar para esta estimación, se asume que la aceleración medida por la IMU es igual a la aceleración de gravedad.

$$ f(\, {}^{S}_{E}\hat{\boldsymbol{q}}, \, {}^{E}\hat{\boldsymbol{g}}, \, {}^{S}\hat{\boldsymbol{a}}) = \, ({}^{S}_{E}\hat{\boldsymbol{q}})^{*} \, \otimes \, {}^{E}\hat{\boldsymbol{g}} \otimes \, {}^{S}_{E}\hat{\boldsymbol{q}} \, - \, {}^{S}\hat{\boldsymbol{a}} \tag{3.7} $$

$$ {}^{E}\hat{\boldsymbol{g}} = [\, 0 \; g_x \; g_y \; g_z \,] = [\, 0 \; 0 \; 0 \; 1 \,] \tag{3.8} $$

$$ {}^{S}\hat{\boldsymbol{a}} = [\, 0 \; a_x \; a_y \; a_z \,] \tag{3.9} $$

Donde ${}^{S}_{E}\hat{\boldsymbol{q}}$ es la orientación estimada en base al acelerómetro, ${}^{E}\hat{\boldsymbol{g}}$ es la gravedad en el sistema de referencia global y ${}^{S}\hat{\boldsymbol{a}}$ es la aceleración sensada por la IMU.

Como ${}^{E}\hat{\boldsymbol{g}}$ es una constante, se redefine 3.7:

$$ f(\, {}^{S}_{E}\hat{\boldsymbol{q}}, \, {}^{S}\hat{\boldsymbol{a}}) = \, ({}^{S}_{E}\hat{\boldsymbol{q}})^{*} \, \otimes [\, 0 \; 0 \; 0 \; 1 \,] \otimes \, {}^{S}_{E}\hat{\boldsymbol{q}} \, - \, {}^{S}\hat{\boldsymbol{a}} \tag{3.10} $$

El gradiente ∇f, se calcula en base al jacobiano de la función f:

$$ \nabla f(\, {}^{S}_{E}\hat{\boldsymbol{q}}, \, {}^{S}\hat{\boldsymbol{a}}) = J^{T}(\, {}^{S}_{E}\hat{\boldsymbol{q}}) f(\, {}^{S}_{E}\hat{\boldsymbol{q}}, \, {}^{S}\hat{\boldsymbol{a}}) \tag{3.11} $$

$$ f(\, {}^{S}_{E}\hat{\boldsymbol{q}}, \, {}^{S}\hat{\boldsymbol{a}}) = \begin{bmatrix} 2(bd - ac) - a_x \\ 2(ab + cd) - a_y \\ 2(\frac{1}{2} - b^2 - c^2) - a_z \end{bmatrix} \tag{3.12} $$

$$J(\,{}^{S}_{E}\hat{\boldsymbol{q}}) = \begin{bmatrix} -2c & 2d & -2a & 2b \\ 2b & 2a & 2d & 2c \\ 0 & -4b & -4c & 0 \end{bmatrix} \tag{3.13}$$

La magnitud del error con dirección $\frac{\nabla f}{\|\nabla f\|}$, se define con el parámetro β. El mismo, representa el ruido con media cero presente en el giróscopo que, en la sección 2.3.7, se describe como el random walk de velocidad angular. El coeficiente β se calcula de la siguiente forma en base a la ecuación 3.6:

$$\beta = \left\| \frac{1}{2}\hat{\boldsymbol{q}} \otimes [0\ \tilde{\omega}_\beta\ \tilde{\omega}_\beta\ \tilde{\omega}_\beta] \right\| = \sqrt{\frac{3}{4}}\ \tilde{\omega}_\beta \tag{3.14}$$

Donde $\tilde{\omega}_\beta$ es un coeficiente que cuantifica el random walk de velocidad angular. El mismo se obtiene como la raíz cuadrada del mayor coeficiente de varianza allan obtenido para el random walk de velocidad angular (tabla 3.2).

Finalmente, se ejecuta la integración 3.17 en base a 3.15 y 3.16:

$${}^{S}_{E}\dot{\hat{\boldsymbol{q}}}_{\epsilon,t} = \frac{\nabla \boldsymbol{f}}{\|\nabla \boldsymbol{f}\|} \tag{3.15}$$

$${}^{S}_{E}\dot{\boldsymbol{q}}_{est,t} = {}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t} - \beta\ {}^{S}_{E}\dot{\hat{\boldsymbol{q}}}_{\epsilon,t} \tag{3.16}$$

$${}^{S}_{E}\boldsymbol{q}_{est,t} = {}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1} + {}^{S}_{E}\dot{\boldsymbol{q}}_{est,t}\Delta t \tag{3.17}$$

Donde ${}^{S}_{E}\dot{\hat{\boldsymbol{q}}}_{\epsilon,t}$ es la dirección del error estimado, ${}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t}$ es la rapidez de cambio de la orientación obtenida con el giróscopo y $\boldsymbol{f}$ es la función de costo definida para el cálculo con el acelerómetro. La ecuación 3.16 muestra el concepto del filtro, el cual calcula la rapidez de cambio de la orientación a partir del giróscopo y realiza un ajuste en base al acelerómetro.

El filtro se puede implementar con un magnetómetro y de esa forma se puede obtener la orientación absoluta de la MIMU (sección 2.3). Además esa implementación del filtro introduce un ajuste continuo del bias drift del giróscopo y de las distorsiones magnéticas.

3.5.4. Cálculo de parámetros espacio-temporales

Para calcular los parámetros espacio-temporales, es fundamental segmentar el ciclo de marcha. Para ello es necesario detectar los contactos de talón y despegue de dedos (a partir de ahora llamados contacto inicial y final).

Trojaniello et. al. [90] compara cuatro métodos de detección que utilizan una sola IMU en la espalda baja y evalúa su performance. El más robusto es el método implementado por McCamley et. al. [91]. Este algoritmo consiste en 3 pasos:

- Suavizar la señal
- Obtener la primera y segunda derivada con Wavelets [92]
- Detectar máximos y mínimos en las transformadas

El suavizado se implementa con un filtro pasabajos Butterworth de 2^{do} orden en 2Hz. Luego se aplican las transformadas contínuas primero con una Wavelet Gaus 1 y luego con una Gaus 2, utilizando la función *cwt* de la librería PyWavelets [93].

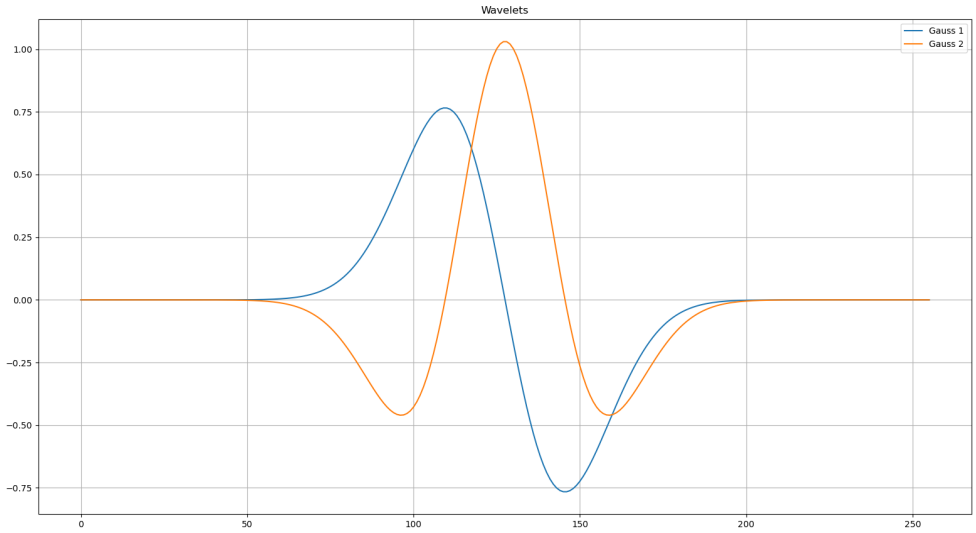


Figura 3.30: Wavelets Gaussianas utilizadas para calcular las derivadas necesarias para el algoritmo presentado en [91]

La transformada Wavelet requiere de una escala o una serie de escalas a especificar.

La escala es un factor por el cual se estira y contrae la forma de onda de la Wavelet y se relaciona con la banda de frecuencia que se está representando. En este caso, la escala se definió empíricamente, partiendo de saber que un ciclo de marcha tiene un período cercano a 1 segundo. Concretamente, la escala escogida es 70, que a 238Hz de frecuencia de sampleo, para la Wavelet Gauss 1, representa una banda de frecuencia centrada en 0,68Hz y para la Wavelet Gauss 2, 1,02Hz

El contacto inicial de cada paso se obtiene hallando los mínimos de la primera transformada (o los máximos de la transformada con signo negado). Por su parte los contactos finales, se detectan en los máximos de la segunda transformada.

Para eso, se utilizó la función *find_peaks* de la librería Scipy [94]. La misma proporciona una serie de parámetros ajustables para optimizar la performance del algoritmo según la morfología de los picos a detectar. En particular se utilizaron dos: el ancho y la prominencia[1] de pico. Nuevamente, definidos empíricamente. El ancho para ambas detecciones es de 80 muestras. Para la primera transformada, la prominencia de los picos se establece en 0,5. Para la segunda, en 0,2.

Una vez detectados los eventos, se pueden calcular los parámetros temporales, pero las métricas que implican distancia (como el largo de paso o la velocidad), requieren un postprocesamiento mayor.

Basándose en un modelo de péndulo invertido, se calcula el largo de paso [40, 41]:

$$largo\ de\ paso = 2\ \sqrt{2\ l\ h - h^2} \tag{3.18}$$

Donde l es el largo de las piernas y h es la amplitud del desplazamiento vertical de la pelvis.

El desplazamiento vertical se calcula integrando dos veces la señal de aceleración vertical

[1] La prominencia mide la diferencia entre el valor del máximo y el menor valor de sus alrededores. Da una noción de cuánto sobresale el pico por sobre la morfología adyacente.

en el sistema de referencia global.

El resto de los parámetros se calcula de la siguiente forma:

$$velocidad = pasos \ \cdot \ largo\ de\ paso \ / \ tiempo \tag{3.19}$$

$$cadencia = pasos \ / \ tiempo \tag{3.20}$$

$$tiempo\ de\ ciclo = tiempo \ / \ (pasos \ / \ 2) \tag{3.21}$$

$$tiempo\ de\ sustentación = \frac{1}{pasos}\left(\sum_{k=1}^{pi}(t_k^{cfi} - t_k^{cii}) + \sum_{j=1}^{pd}(t_j^{cfd} - t_j^{cid}) \right) \tag{3.22}$$

Donde la sumatoria del primer término es sobre la diferencia de tiempo entre el contacto final e inicial de los pasos de pié izquierdo. La segunda sumatoria es lo mismo, pero para los pasos de pié derecho

Experimentos

Al finalizar la implementación del prototipo, se llevaron a cabo una serie de experimentos para optimizar parámetros de los algoritmos, definir filtrados en frecuencia y evaluar el desempeño general.

Como se menciona en la sección 2.2.1, los sistemas de captura de movimiento basados en sensores inerciales, se contrastan y validan con los sistemas optoelectrónicos. Al no poder acceder al uso de uno de estos sistemas, se diseñaron y aplicaron experimentos simples y posibles de realizar en cualquier entorno. Además se utilizó la base de datos para validar el algoritmo de orientación (sección 4.1.1).

El procesamiento de las señales capturadas, resulta en la orientación de la IMU y en parámetros espacio-temporales de la marcha: largo de paso, tiempo de ciclo, tiempo de sustentación (y balanceo), cadencia y velocidad.

La métrica utilizada para calcular el error de la orientación con quaterniones es el coeficiente de desviación, medido en radianes, definido en [95] como:

$$DI = 2 * arccos(|\hat{q} * q|) \tag{4.1}$$

Donde $\hat{q}$ es la estimación del filtro Madgwick y q es la referencia.

Por otro lado, para evaluar la performance de la detección de eventos, se utiliza el cociente entre pasos detectados y reales.

A continuación se describen los experimentos realizados y el por qué de cada uno de ellos. En cada apartado, se presentan los resultados obtenidos. La discusión de los resultados se encuentra en la sección 5.1.

4.1. Orientación

4.1.1. Análisis de base de datos (RepoIMU)

El repositorio RepoIMU [30] es publicado con el propósito de validar algoritmos de orientación basados en IMUs. Como referencia se brindan los resultados de un sistema optoelectrónico Vicon en formato de quaterniones. Los datos entre los sistemas se encuentran sincronizados entre sí. La frecuencia de muestreo es de 100Hz.

La base de datos contiene 5 experimentos distintos

- Sensor estático
- Rotaciones lentas en cada eje
- Rotaciones rápidas en cada eje
- Traslación en cada eje
- Rotación libre

Las series de datos de rotaciones rápidas presentan saturación en las señales de velocidad angular. Por tal motivo los resultados obtenidos con esos datos no se presentan aquí.

Experimento	Resultado por experimento [rad]	Promedio total [rad]
Estático	0,0895	0,0667
Rotación uniaxial	0,0518	
Traslación uniaxial	0,0642	
Rotación libre	0,0613	

Tabla 4.1: El resultado por experimento es el promedio del coeficiente de desviación entre los ensayos de cada experimento. Luego se presenta el promedio de los coeficientes como una métrica global para el prototipo.

4.1.2. Experimento sobre el prototipo

Por otro lado se decidió realizar un experimento estático y otro dinámico aplicando movimientos y poses similares a los de la calibración propuesta en [96]. Esto para verificar el desempeño de los algoritmos en el prototipo y bajo las condiciones de calibración implementadas.

El experimento estático consistió en realizar una captura por cada eje de forma paralela y antiparalela a la dirección de gravedad. Completando un total de 6 mediciones, cada una de 10 segundos. Se calculó el coeficiente de desviación de cada una y luego se obtuvo el promedio (Tabla 4.2)

Por otro lado, el experimento dinámico consistió en mover la IMU iniciando y finalizando en la misma posición y orientación conocidas. Para garantizar tal condición es necesario reposar la IMU contra la esquina entre una superficie plana y algún objeto pesado o una pared. 3 Se procuró aplicar rotaciones positivas y negativas de 360° sobre cada eje. En este caso el coeficiente de desviación se calcula entre la orientación inicial y la final.

Experimento	Coeficiente de desviación [rad]
Estático	0,3877
Dinámico	0,1556

Tabla 4.2: El resultado del experimento estático es el promedio del coeficiente de desviación de las 6 mediciones realizadas. El resultado para el experimento dinámico es el coeficiente de desviación entre la orientación inicial y la final de la IMU.

4.2. Parámetros espacio-temporales

Para evaluar la detección de eventos, se realizaron una serie de caminatas rectas, de 5 metros, sobre terreno plano. La IMU se ubicó sujeta a la zona lumbar. Se contó la cantidad de pasos realizados y se midió el largo de cada uno. Los participantes caminaron sin calzado, con medias y talco en los pies.

Luego de la captura de datos, se procesa la señal para obtener la aceleración en el marco de referencia global. La aceleración vertical es compensada con la gravedad y filtrada en 0,4Hz con un filtro Butterworth de 4^{to} orden (Figura 4.4) para atenuar la acumulación de error en procesamiento siguiente. Por el mismo motivo, el filtro se vuelve a aplicar sobre la velocidad (Figura 4.5). El desplazamiento vertical de la pelvis se calcula con una segunda integral. Se aplica la ecuación 3.18 para obtener los largos de paso.

En base a los eventos detectados, se calcula: el largo de paso, la cadencia, la velocidad, la duración del ciclo de marcha y la duración de la fase de balanceo.

A continuación se presentan los resultados para uno de los ensayos realizados.

Figura 4.1: Foto del resultado del experimento del cual se muestran los resultados. El sujeto dejó marcados sus pasos con talco. Las líneas negras sobre el suelo se encuentran distanciadas 1m entre sí.

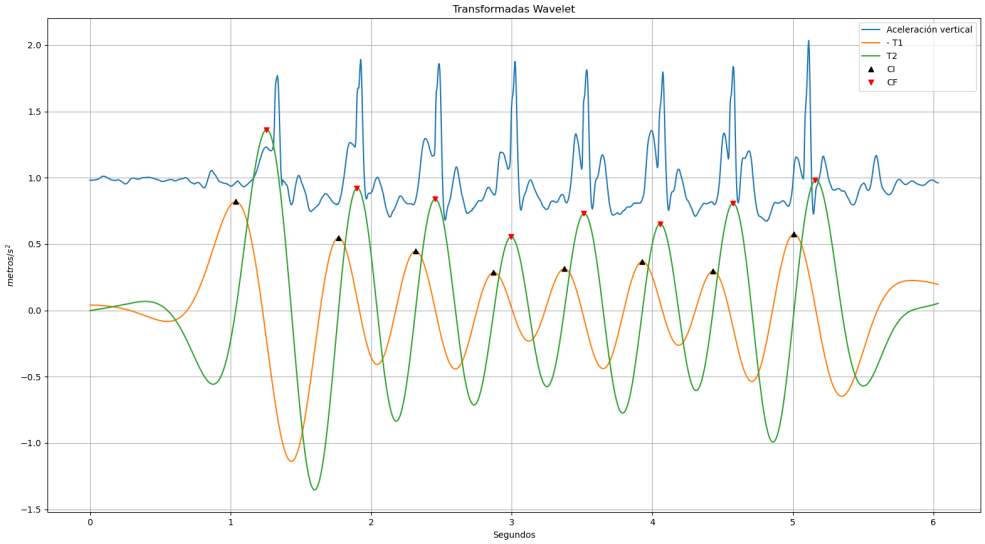


Figura 4.2: Resultado del algoritmo de detección de eventos. Se identificaron los 8 pasos correctamente. En naranja la transformada Wavelet con la que se obtienen los contactos iniciales. En verde, la segunda transformada con la que se obtienen los contactos finales. Los marcadores negros y rojos indican los contactos iniciales y finales respectivamente. En azul se ve la señal de aceleración vertical en el sistema de referencia global.

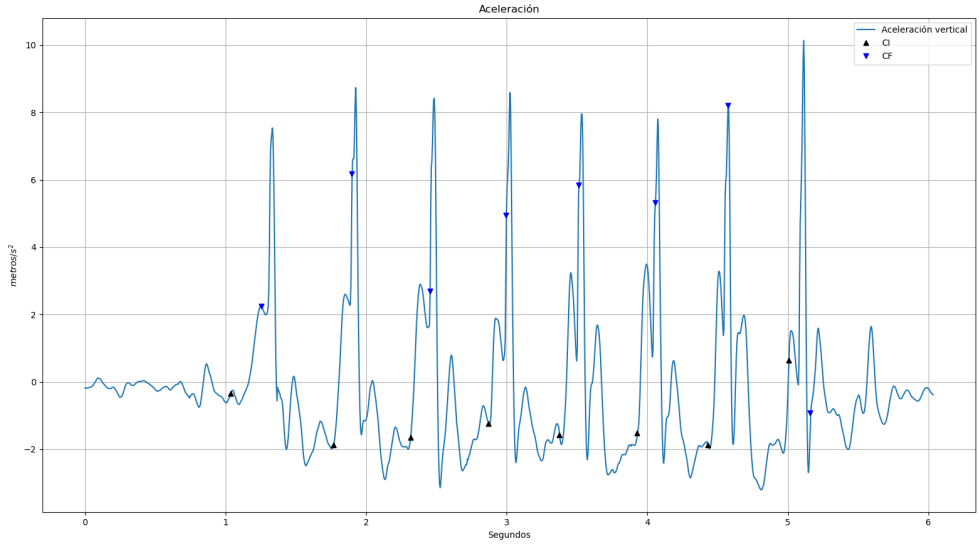


Figura 4.3: Señal de aceleración mostrada en la figura 4.2. Los marcadores indican los contactos iniciales y finales respectivamente.

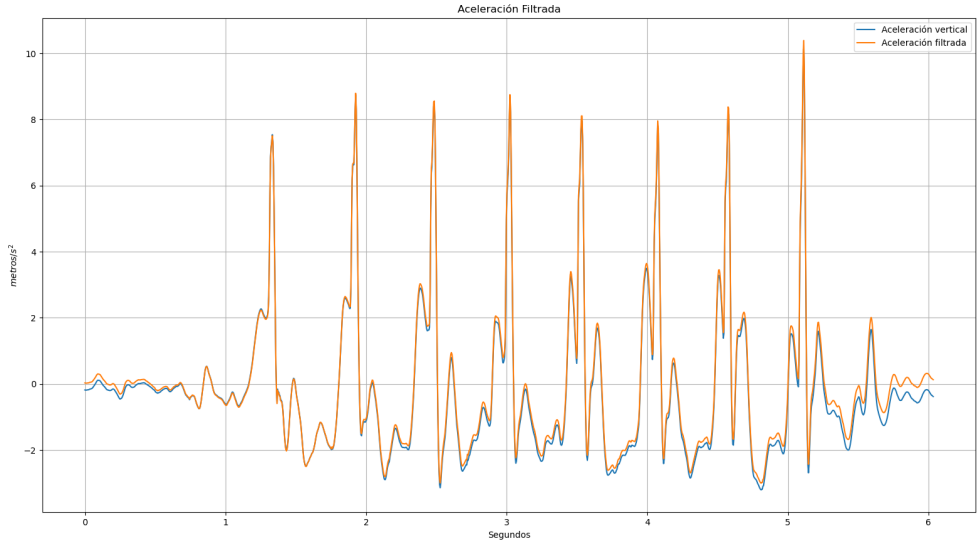


Figura 4.4: Señal de aceleración mostrada en la figura 4.3 (azul) y la misma señal filtrada con un Butterworth pasaltos de 4to orden en 0,4Hz (naranja).

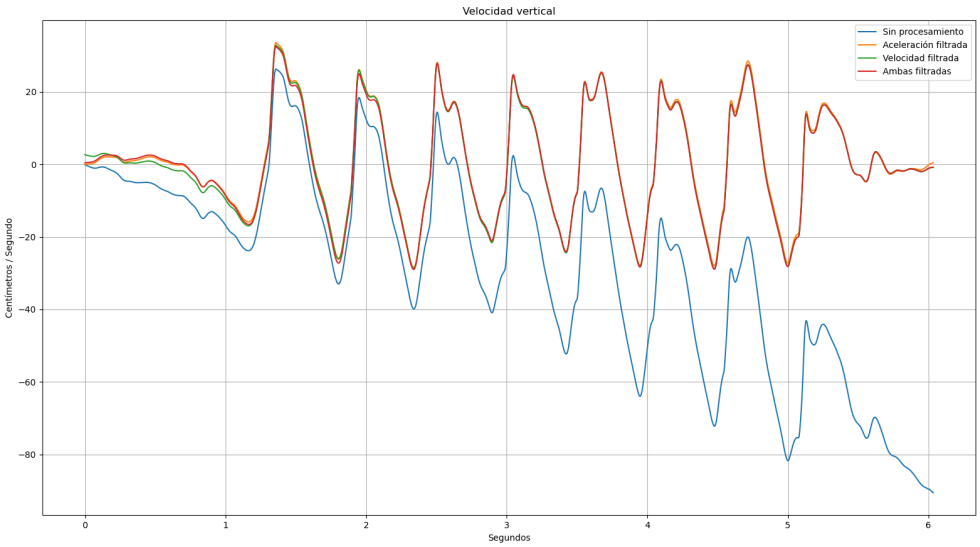


Figura 4.5: Señal de velocidad como resultado de distintos procesamientos. Se aplicó el filtro Butterworth en la señal de aceleración, en la de velocidad y en ambas para poder visualizar el efecto del filtrado en cada caso.

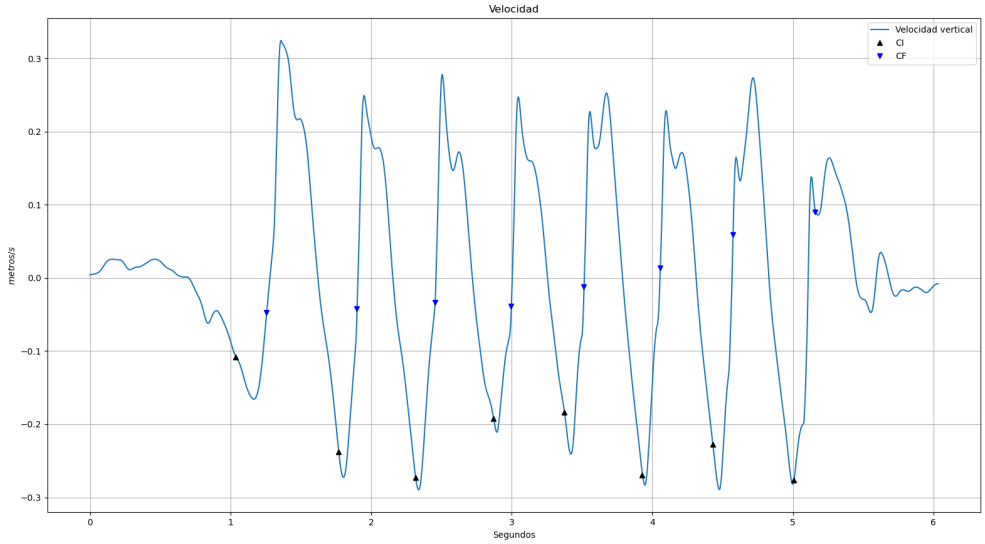


Figura 4.6: Señal de velocidad (4^{ta} serie de la figura 4.5) en base a la aceleración filtrada de la figura 4.4.

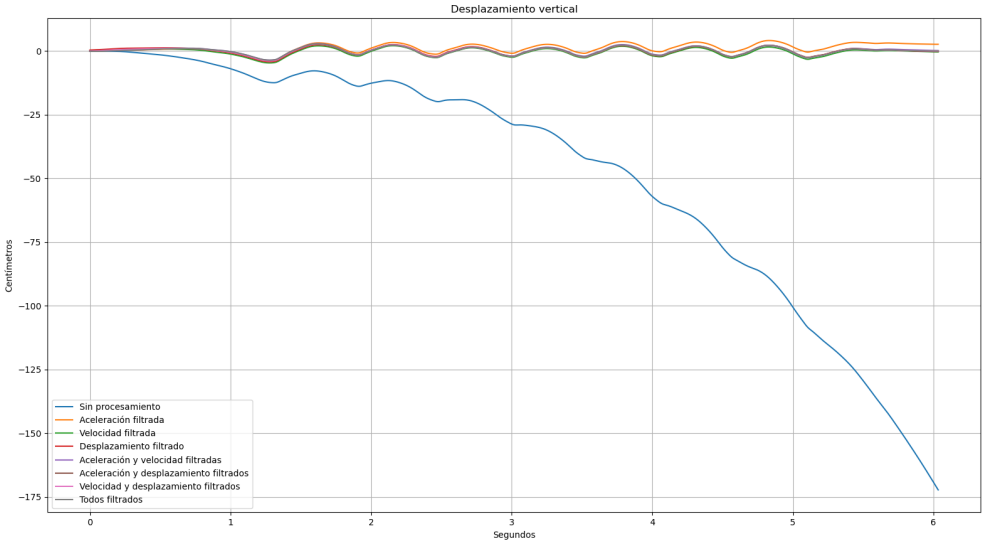


Figura 4.7: Señal de desplazamiento en base a distintos filtrados de la aceleración, la velocidad y el mismo desplazamiento. Se aplicó el filtro Butterworth en todas las posibles combinaciones para visualizar el impacto del filtrado en el resultado final del desplazamiento.

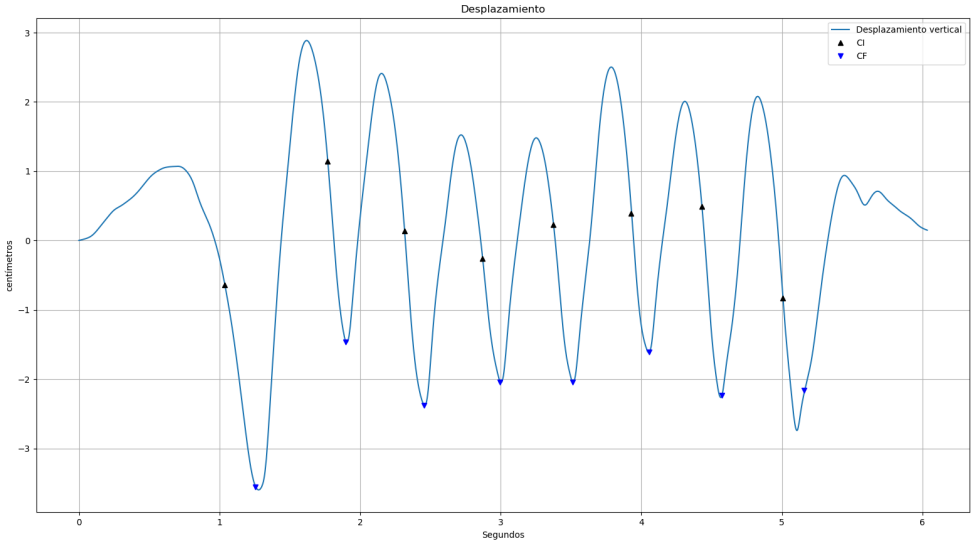


Figura 4.8: Señal de desplazamiento (5^{ta} serie de la figura (4.7) en base a la velocidad mostrada en la figura 4.6. Los marcadores indican los contactos iniciales y finales respectivamente. A diferencia de la aceleración y de la velocidad, se decidió no filtrar el desplazamiento ya que no se podía asegurar que fuese un procesamiento necesario.

Métrica	Valor
Largo de paso	55 cm
Velocidad	98 cm / seg
Cadencia	106 / min
Tiempo de ciclo	1,1 seg
Tiempo de sustentación	0,7 seg (63,6 %)

Tabla 4.3: Parámetros espaciotemporales obtenidos

Finalmente, se agregan los ángulos de rotación, inclinación y abducción de la pelvis. Estos ángulos se obtienen con las ecuaciones 2.34, 2.35 y 2.36. Luego se promedian los ciclos de marcha, comenzando desde el segundo contacto inicial para prevenir efectos por las distorsiones mencionadas anteriormente.

Recordando que al no utilizar el magnetómetro, los ángulos de rotación de pelvis (eje vertical) no pueden ser obtenidos de forma absoluta en el sistema de referencia global.

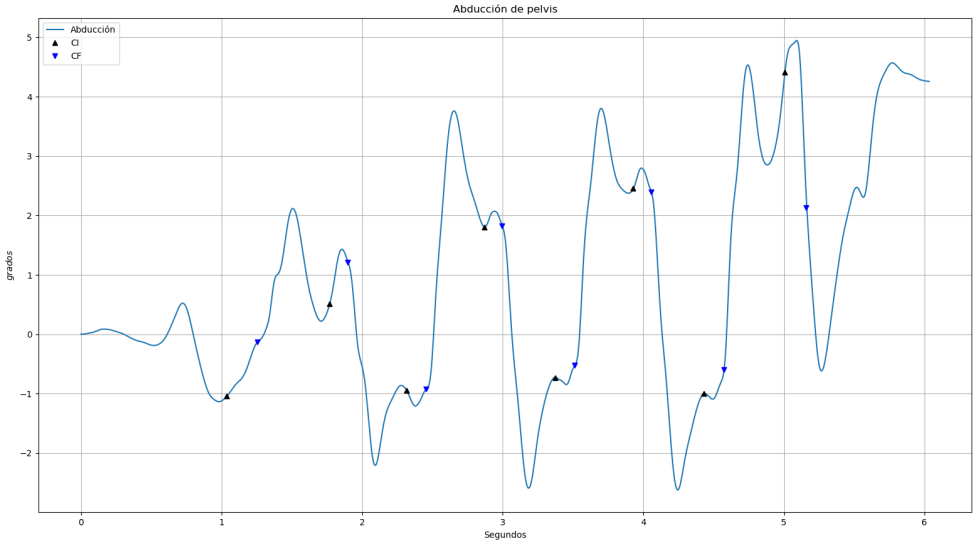


Figura 4.9: Abducción de pelvis a lo largo de la caminata.

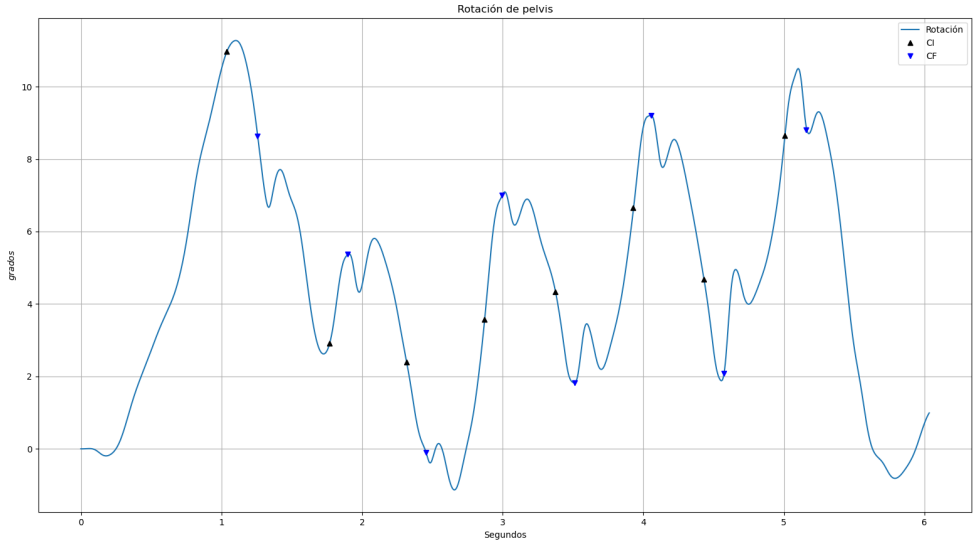


Figura 4.10: Rotación de pelvis a lo largo de la caminata.

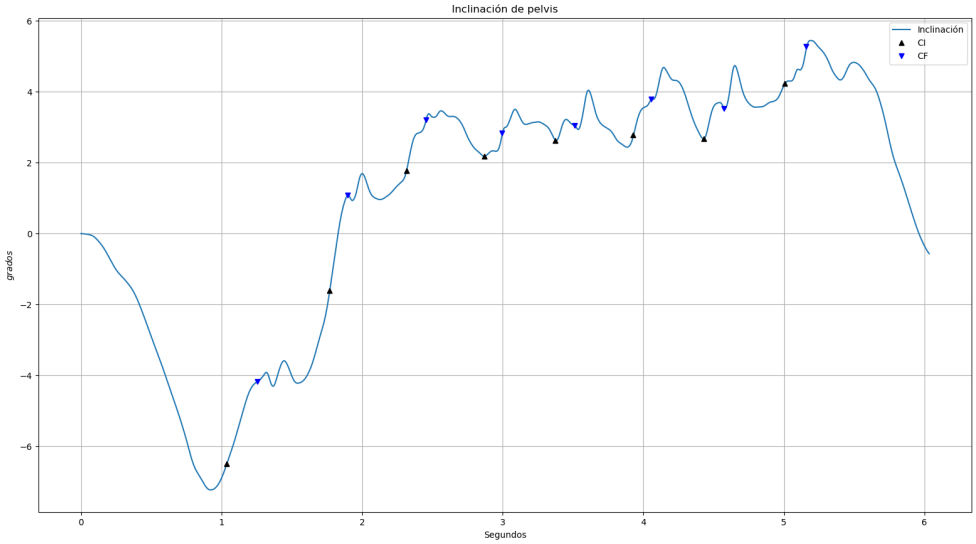


Figura 4.11: Inclinación de pelvis a lo largo de la caminata.

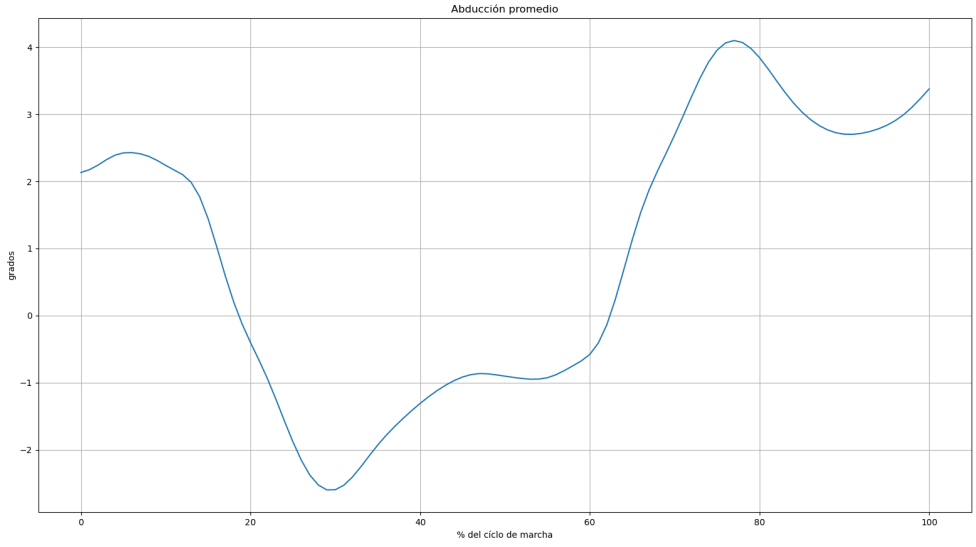


Figura 4.12: Promedio de la abducción de cada ciclo.

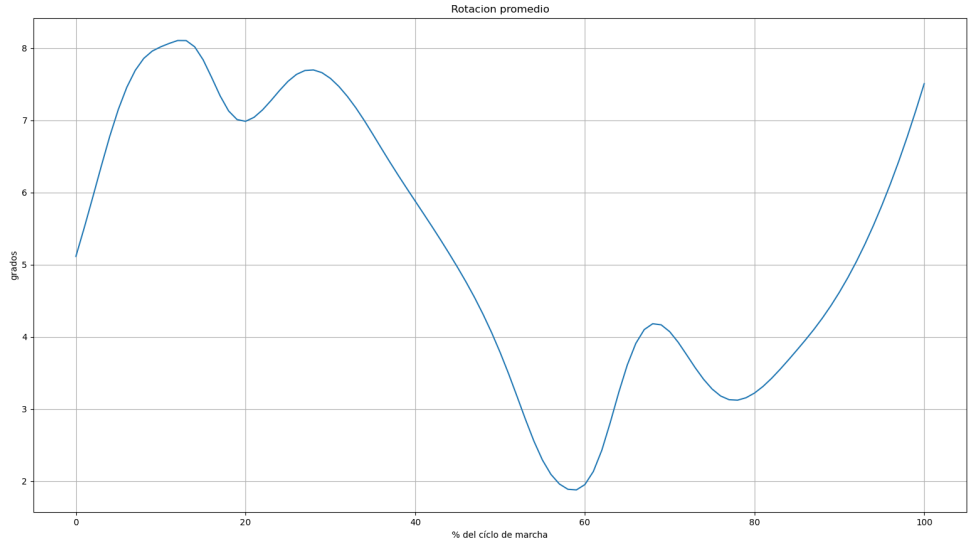


Figura 4.13: Promedio de la rotación de cada ciclo.

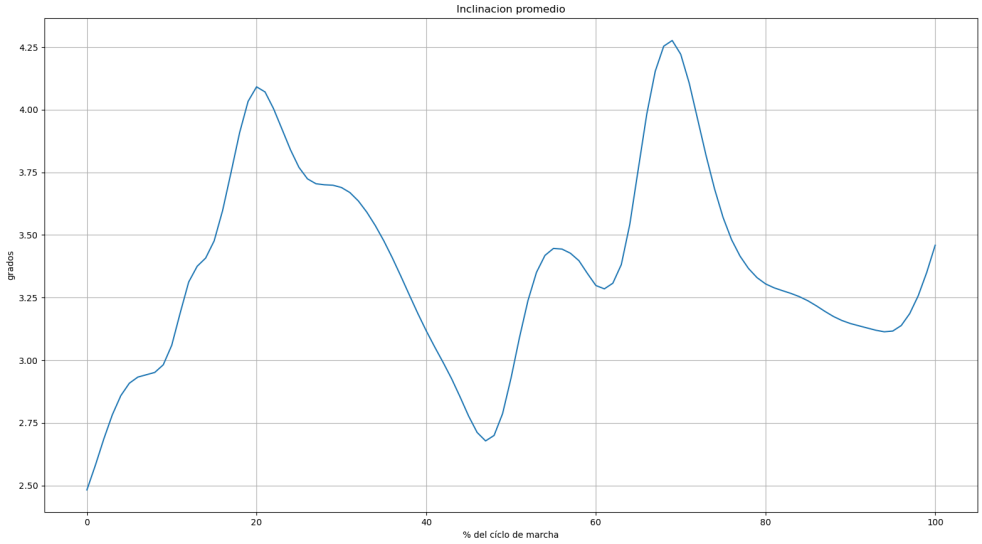


Figura 4.14: Promedio de la inclinación de cada ciclo.

Discusión y conclusiones

5.1. Discusión

En concordancia con la literatura revisada, los resultados obtenidos muestran que tener una correcta estimación de la orientación, es fundamental para poder calcular otras variables como la posición, el desplazamiento (Sección 2.3.4) y los parámetros espacio-temporales de la marcha (Sección 3.5.4). Tanto la calibración como el algoritmo de fusión de sensores son la base desde la cual se puede extraer la información que se encuentra en los datos capturados. Partiendo de esos algoritmos, fue posible implementar la detección de eventos y se culminó el postprocesamiento con dos simples filtros Butterworth aplicados a la señal de aceleración y de velocidad.

En base a las tablas 4.1 y 4.2, la estimación de la orientación tiene mejor desempeño con el sensor en movimiento (traslación, rotación o ambas) que con el sensor estático. En principio, esta diferencia no presenta un impedimento en la usabilidad del dispositivo en un laboratorio de marcha ya que su condición de funcionamiento sería la misma. Esto puede ser debido a que, estando estática la IMU, el algoritmo de Madgwick no es capaz de estimar el error (Eq. 3.15). En la publicación asociada [89], se describe su implementación completa utilizando un magnetómetro para compensar el bias de los giróscopos. No solo eso, sino que la incorporación del magnetómetro resuelve la orientación absoluta de la IMU, lo cual por sí sólo es una herramienta para mitigar el drift de la orientación. Por el otro lado, que el algoritmo de

fusión se vea afectado por el bias de los sensores, significa que la calibración del dispositivo no logró mitigar la totalidad del error. Esto presenta dos posibles problemas: que la metodología utilizada para calibrar la IMU no sea lo suficientemente robusta y que la fluctuación del bias de la IMU sea significativa como para requerir compensación en tiempo real.

Para la calibración propuesta en [53], se implementó un algoritmo que estima si la IMU se encuentra estática o no (Apéndice C). El mismo podría modificarse para funcionar en tiempo real y abordar la problemática detectada. Esto no alcanzó a llevarse a cabo dado el alcance establecido del presente trabajo.

Por el lado de la detección de eventos, el algoritmo implementado logró identificar el 100 % de ellos. Esto se dió tanto en los experimentos controlados (desarrollados en la sección anterior) como en caminatas informales, puertas adentro en línea recta y curvas suaves.

Obervando la figura 4.2, se puede ver que la detección es certera en todos los pasos. La ubicación de los eventos sobre la forma de onda del primer y último paso, se ve distinta a la del resto (figura 4.3). La morfología de la onda en esos dos momentos también es distinta y entonces sus componenentes de frecuencia deben seguir esa misma tendencia. Esto explica por qué los picos de las transformadas Wavelet se ubican en distintos instantes de tiempo en relación a las formas de onda. Como se mencionó en la sección 3.5.4, la escala de las trasnformadas Wavelet se definió empíricamente, partiendo de saber que la duración de un ciclo de marcha es de al rededor de 1 segundo.

Para calcular el largo de paso, se utilizó un modelo de péndulo invertido, ya validado en otras publicaciones [40, 41, 46]. De todas formas el resultado obtenido (55 cm) fue un 24 % menor al valor real medio entre pisadas de talco (73 cm). Zijlstra *et. al.* [46] menciona resultados similares, los cuales fueron corregidos empíricamente con un factor de 1,25. Aplicando esa misma corrección, el error se reduce de 24 % a 5 %. De todas formas, para validar el coeficiente utilizado es necesario especificar una población y realizar ensayos sobre una muestra más grande.

En cuanto a los ángulos de pelvis, los de rotación (figura 4.13) y de abducción (figura

4.12) muestran una morfología similar a la vista en la literatura. Por su parte, los picos de onda del ángulo de inclinación presentan fluctuaciones que parecen ser producidos por artefactos de movimientos a causa del agarre de la IMU al sujeto (figura 4.14).

Antes de cerrar la discusión, es necesario dejar en claro que el prototipo requiere mejoras y ser validado frente a un sistema optoelectrónico. No sólo eso, sino que, para garantizar su correcto desempeño en un laboratorio de marcha se deben realizar experimentaciones en campo. En primera instancia con una población de control y luego con individuos que sufran algún trastorno de la marcha. A partir de ello se podrá definir el alcance del producto y si es apto para su uso como herramienta de diagnóstico y rehabilitación.

5.2. Conclusiones

El prototipo construido sienta las bases para el desarrollo de un sistema de captura de movimiento con proyección a ser utilizado en centros de rehabilitación y laboratorios de marcha. Además es un paso con la intención de brindar una solución asequible dentro del contexto socio-económico de la región frente a opciones de alto costo producidas en el extranjero.

Para llevar a cabo este trabajo fue necesario utilizar conocimientos adquiridos durante la carrera de grado así como también desarrollar muchos otros. El proyecto abarca fundamentos de electrónica, matemática, métodos numéricos, marcha humana y conceptos de rehabilitación. Al margen de la complejidad de cada tema por sí sólo, el mayor desafío se presentó al afrontar un proyecto que no suele ser conducido por una persona sola, sino por un conjunto de especialistas interdisciplinarios.

Los resultados obtenidos muestran que lo investigado se llevó a la práctica con éxito aunque hay mucho lugar para mejoras, tanto en el diseño como en la experimentación y la validación.

5.2.1. Trabajo a futuro

En concordancia con el objeto de este proyecto, se identifican una serie de futuros trabajos a realizar para completar el proceso hasta alcanzar un producto que sea apto para su uso en rehabilitación.

Partiendo de la base de un prototipo funcional, la medida que mayor impacto tendría es la de trabajar en la presentación del dispositivo en lo que a usabilidad se refiere. Tanto en la parte de hardware (gabinete a medida y sujetadores dedicados) como en el software (interfaz de usuario gráfica y conexión con base de datos). Esto impacta directamente en la facilidad para corregir errores detectados durante esta primera etapa del proceso y continuar desarrollando el dispositivo.

Luego, por el lado del procesamiento de las señales, el próximo paso sería la incorporación del magnetómetro y del sensor de temperatura interno. En cuanto a la experimentación en campo, el prototipo debe ser puesto a prueba con un número significativo de personas y debe ser contrastado contra un sistema optoelectrónico.

Bibliografía

[1] R. W. Bohannon, A. W. Andrews, and M. B. Smith, "Rehabilitation goals of patients with hemiplegia," *International Journal of Rehabilitation Research*, vol. 11, no. 2, pp. 181–184, 1988.

[2] P. Grover and O. Volshteyn, "Energy expenditure during basic mobility and approaches to energy conservation,"

[3] R. L. Waters and S. Mulroy, "The energy expenditure of normal and pathologic gait," *Gait & posture*, vol. 9, no. 3, pp. 207–231, 1999.

[4] E. Y. Han, J. H. Choi, S.-H. Kim, and S. H. Im, "The effect of weight bearing on bone mineral density and bone growth in children with cerebral palsy: A randomized controlled preliminary trial," *Medicine*, vol. 96, no. 10, 2017.

[5] R. P. Bhide, C. Solomons, S. Devsahayam, and G. Tharion, "Exercise and gait training in persons with paraplegia and its effect on muscle properties," *Journal of back and musculoskeletal rehabilitation*, vol. 28, no. 4, pp. 739–747, 2015.

[6] M. Farhadian, A. Jameh Bozorgi, F. Zadeh, M. Fakhreh, and Z. Morovati, "Effect of gait retraining on balance, activities of daily living, quality of life and depression in stroke patients," *Iranian Rehabilitation Journal*, vol. 13, no. 4, pp. 116–119, 2015.

[7] A. Godfrey, R. Conway, D. Meagher, and G. ÓLaighin, "Direct measurement of human movement by accelerometry," *Medical engineering & physics*, vol. 30, no. 10, pp. 1364–1386, 2008.

[8] T. A. Wren, N. Y. Otsuka, R. E. Bowen, A. A. Scaduto, L. S. Chan, M. Sheng, R. Hara, and R. M. Kay, "Influence of gait analysis on decision-making for lower extremity orthopaedic surgery: Baseline data from a randomized controlled trial," *Gait & posture*, vol. 34, no. 3, pp. 364–369, 2011.

[9] N. Margiotta, G. Avitabile, and G. Coviello, "A wearable wireless system for gait analysis for early diagnosis of alzheimer and parkinson disease," in *2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA)*, pp. 1–4, IEEE, 2016.

[10] R. Postuma, A. Lang, J. Gagnon, A. Pelletier, and J. Montplaisir, "How does parkinsonism start? prodromal parkinsonism motor changes in idiopathic rem sleep behaviour disorder," *Brain*, vol. 135, no. 6, pp. 1860–1870, 2012.

[11] A. Mirelman, T. Gurevich, N. Giladi, A. Bar-Shira, A. Orr-Urtreger, and J. M. Hausdorff, "Gait alterations in healthy carriers of the lrrk2 g2019s mutation," *Annals of neurology*, vol. 69, no. 1, pp. 193–197, 2011.

[12] S. Lerche, K. Seppi, S. Behnke, I. Liepelt-Scarfone, J. Godau, P. Mahlknecht, A. Gaenslen, K. Brockmann, K. Srulijes, H. Huber, *et al.*, "Risk factors and prodromal markers and the development of parkinson's disease," *Journal of neurology*, vol. 261, no. 1, pp. 180–187, 2014.

[13] D. T. Wade, "Goal setting in rehabilitation: an overview of what, why and how," 2009.

[14] D. H. Sutherland, "The evolution of clinical gait analysis part l: kinesiological emg," *Gait & posture*, vol. 14, no. 1, pp. 61–70, 2001.

[15] D. H. Sutherland, "The evolution of clinical gait analysis: Part ii kinematics," *Gait & posture*, vol. 16, no. 2, pp. 159–179, 2002.

[16] D. H. Sutherland, "The evolution of clinical gait analysis part iii–kinetics and energy assessment," *Gait & posture*, vol. 21, no. 4, pp. 447–461, 2005.

[17] M. W. Whittle, *Gait analysis: an introduction*. Butterworth-Heinemann, 2014.

[18] J. L. Robinson and G. L. Smidt, "Quantitative gait evaluation in the clinic," *Physical Therapy*, vol. 61, no. 3, pp. 351–353, 1981.

[19] A. D. Koelewijn, D. Heinrich, and A. J. Van Den Bogert, "Metabolic cost calculations of gait using musculoskeletal energy models, a comparison study," *PloS one*, vol. 14, no. 9, p. e0222037, 2019.

[20] M. W. Whittle, "Clinical gait analysis: A review," *Human movement science*, vol. 15, no. 3, pp. 369–387, 1996.

[21] V. Cimolin and M. Galli, "Summary measures for clinical gait analysis: a literature review," *Gait & posture*, vol. 39, no. 4, pp. 1005–1010, 2014.

[22] C. L. Lewis, N. M. Laudicina, A. Khuu, and K. L. Loverro, "The human pelvis: variation in structure and function during gait," *The Anatomical Record*, vol. 300, no. 4, pp. 633–642, 2017.

[23] M. Iosa, P. Picerno, S. Paolucci, and G. Morone, "Wearable inertial sensors for human movement analysis," *Expert review of medical devices*, vol. 13, no. 7, pp. 641–659, 2016.

[24] W. Tao, T. Liu, R. Zheng, and H. Feng, "Gait analysis using wearable sensors," *Sensors*, vol. 12, no. 2, pp. 2255–2283, 2012.

[25] F. Petraglia, L. Scarcella, G. Pedrazzi, L. Brancato, R. Puers, and C. Costantino, "Inertial sensors versus standard systems in gait analysis: a systematic review and meta-analysis," *Eur. J. Phys. Rehabil. Med*, vol. 55, pp. 265–280, 2019.

[26] R. Rajagopalan, I. Litvan, and T.-P. Jung, "Fall prediction and prevention systems: recent trends, challenges, and future research directions," *Sensors*, vol. 17, no. 11, p. 2509, 2017.

[27] C.-C. Yang and Y.-L. Hsu, "A review of accelerometry-based wearable motion detectors for physical activity monitoring," *Sensors*, vol. 10, no. 8, pp. 7772–7788, 2010.

[28] A. Cappozzo, U. Della Croce, A. Leardini, and L. Chiari, "Human movement analysis using stereophotogrammetry: Part 1: theoretical background," *Gait & posture*, vol. 21, no. 2, pp. 186–196, 2005.

[29] L. Chiari, U. Della Croce, A. Leardini, and A. Cappozzo, "Human movement analysis using stereophotogrammetry: Part 2: Instrumental errors," *Gait & posture*, vol. 21, no. 2, pp. 197–211, 2005.

[30] A. Szczesna, P. Skurowski, P. Pruszowski, D. Peszor, M. Paszkuta, and K. Wojciechowski, "Reference data set for accuracy evaluation of orientation estimation algorithms for inertial motion capture systems," in *International Conference on Computer Vision and Graphics*, pp. 509–520, Springer, 2016.

[31] A. M. Aurand, J. S. Dufour, and W. S. Marras, "Accuracy map of an optical motion capture system with 42 or 21 cameras in a large measurement volume," *Journal of biomechanics*, vol. 58, pp. 237–240, 2017.

[32] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "A study of vicon system positioning performance," *Sensors*, vol. 17, no. 7, p. 1591, 2017.

[33] J. L. McGinley, R. Baker, R. Wolfe, and M. E. Morris, "The reliability of three-dimensional kinematic gait measurements: a systematic review," *Gait & posture*, vol. 29, no. 3, pp. 360–369, 2009.

[34] R. B. Davis III, S. Ounpuu, D. Tyburski, and J. R. Gage, "A gait analysis data collection and reduction technique," *Human movement science*, vol. 10, no. 5, pp. 575–587, 1991.

[35] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "Opensim: open-source software to create and analyze dynamic simulations of movement," *IEEE transactions on biomedical engineering*, vol. 54, no. 11, pp. 1940–1950, 2007.

[36] A. Cappozzo, F. Catani, U. Della Croce, and A. Leardini, "Position and orientation in space of bones during movement: anatomical frame definition and determination," *Clinical biomechanics*, vol. 10, no. 4, pp. 171–178, 1995.

[37] A. J. Van den Bogert, T. Geijtenbeek, O. Even-Zohar, F. Steenbrink, and E. C. Hardin, "A real-time system for biomechanical analysis of human movement and muscle function," *Medical & biological engineering & computing*, vol. 51, no. 10, pp. 1069–1077, 2013.

[38] E. Flux, M. van der Krogt, P. Cappa, M. Petrarca, K. Desloovere, and J. Harlaar, "The human body model versus conventional gait models for kinematic gait analysis in children with cerebral palsy," *Human Movement Science*, vol. 70, p. 102585, 2020.

[39] D. Jarchi, J. Pope, T. K. Lee, L. Tamjidi, A. Mirzaei, and S. Sanei, "A review on accelerometry-based gait analysis and emerging clinical applications," *IEEE reviews in biomedical engineering*, vol. 11, pp. 177–194, 2018.

[40] F. Buganè, M. G. Benedetti, V. D'Angeli, and A. Leardini, "Estimation of pelvis kinematics in level walking based on a single inertial sensor positioned close to the sacrum: validation on healthy subjects with stereophotogrammetric system," *Biomedical engineering online*, vol. 13, no. 1, p. 146, 2014.

[41] F. Bugané, M. Benedetti, G. Casadio, S. Attala, F. Biagi, M. Manca, and A. Leardini, "Estimation of spatial-temporal gait parameters in level walking based on a single accelerometer: Validation on normal subjects by standard gait analysis," *Computer methods and programs in biomedicine*, vol. 108, no. 1, pp. 129–137, 2012.

[42] E. P. Washabaugh, T. Kalyanaraman, P. G. Adamczyk, E. S. Claflin, and C. Krishnan, "Validity and repeatability of inertial measurement units for measuring gait parameters," *Gait & posture*, vol. 55, pp. 87–93, 2017.

[43] A. Hartmann, K. Murer, R. A. de Bie, and E. D. de Bruin, "Reproducibility of spatio-temporal gait parameters under different conditions in older adults using a trunk tri-axial accelerometer system," *Gait & posture*, vol. 30, no. 3, pp. 351–355, 2009.

[44] M. Henriksen, H. Lund, R. Moe-Nilssen, H. Bliddal, and B. Danneskiod-Samsøe, "Test–retest reliability of trunk accelerometric gait analysis," *Gait & posture*, vol. 19, no. 3, pp. 288–297, 2004.

[45] S. S. Yeo and G. Y. Park, "Accuracy verification of spatio-temporal and kinematic parameters for gait using inertial measurement unit system," *Sensors*, vol. 20, no. 5, p. 1343, 2020.

[46] W. Zijlstra and A. L. Hof, "Assessment of spatio-temporal gait parameters from trunk accelerations during human walking," *Gait & posture*, vol. 18, no. 2, pp. 1–10, 2003.

[47] P. Esser, H. Dawes, J. Collett, M. G. Feltham, and K. Howells, "Assessment of spatio-temporal gait parameters using inertial measurement units in neurological populations," *Gait & posture*, vol. 34, no. 4, pp. 558–560, 2011.

[48] D. Kobsar, C. Olson, R. Paranjape, and J. M. Barden, "The validity of gait variability and fractal dynamics obtained from a single, body-fixed triaxial accelerometer," *Journal of applied biomechanics*, vol. 30, no. 2, pp. 343–347, 2014.

[49] B. Sijobert, M. Benoussaad, J. Denys, R. Pissard-Gibollet, C. Geny, and C. A. Coste, "Implementation and validation of a stride length estimation algorithm, using a single basic inertial sensor on healthy subjects and patients suffering from parkinson's disease," *ElectronicHealthcare*, pp. 704–714, 2015.

[50] S. Sessa, M. Zecca, L. Bartolomeo, T. Takashima, H. Fujimoto, and A. Takanishi, "Reliability of the step phase detection using inertial measurement units: pilot study," *Healthcare technology letters*, vol. 2, no. 3, pp. 58–63, 2015.

[51] Y.-L. Kuo, K. M. Culhane, P. Thomason, O. Tirosh, and R. Baker, "Measuring distance walked and step count in children with cerebral palsy: an evaluation of two portable activity monitors," *Gait & posture*, vol. 29, no. 2, pp. 304–310, 2009.

[52] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for imu calibration without external equipments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3042–3049, IEEE, 2014.

[53] W. Fong, S. Ong, and A. Nee, "Methods for in-field user calibration of an inertial measurement unit without external equipment," *Measurement Science and technology*, vol. 19, no. 8, p. 085202, 2008.

[54] B. Fang, W. Chou, and L. Ding, "An optimal calibration method for a mems inertial measurement unit," *International Journal of Advanced Robotic Systems*, vol. 11, no. 2, p. 14, 2014.

[55] F. Narváez, F. Árbito, and R. Proaño, "A quaternion-based method to imu-to-body alignment for gait analysis," in *International Conference on Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management*, pp. 217–231, Springer, 2018.

[56] G. Wu and P. R. Cavanagh, "Isb recommendations for standardization in the reporting of kinematic data," *Journal of biomechanics*, vol. 28, no. 10, pp. 1257–1261, 1995.

[57] J. Favre, F. Luthi, B. Jolles, O. Siegrist, B. Najafi, and K. Aminian, "A new ambulatory system for comparative evaluation of the three-dimensional knee kinematics, applied to anterior cruciate ligament injuries," *Knee Surgery, Sports Traumatology, Arthroscopy*, vol. 14, no. 7, pp. 592–604, 2006.

[58] K. Liu, T. Liu, K. Shibata, and Y. Inoue, "Ambulatory measurement and analysis of the lower limb 3d posture using wearable sensor system," in *2009 International Conference on Mechatronics and Automation*, pp. 3065–3069, IEEE, 2009.

[59] T. Seel, J. Raisch, and T. Schauer, "Imu-based joint angle measurement for gait analysis," *Sensors*, vol. 14, no. 4, pp. 6891–6909, 2014.

[60] T. Seel, T. Schauer, and J. Raisch, "Joint axis and position estimation from inertial measurement data by exploiting kinematic constraints," in *2012 IEEE International Conference on Control Applications*, pp. 45–49, IEEE, 2012.

[61] R. Takeda, S. Tadano, A. Natorigawa, M. Todoh, and S. Yoshinari, "Gait posture estimation by wearable acceleration and gyro sensor," in *World Congress on Medical Physics and Biomedical Engineering, September 7-12, 2009, Munich, Germany*, pp. 111–114, Springer, 2009.

[62] J. Favre, B. Jolles, R. Aissaoui, and K. Aminian, "Ambulatory measurement of 3d knee joint angle," *Journal of biomechanics*, vol. 41, no. 5, pp. 1029–1035, 2008.

[63] O. J. Woodman, "An introduction to inertial navigation," tech. rep., University of Cambridge, Computer Laboratory, 2007.

[64] J. B. Kuipers, *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999.

[65] C. Jekeli, *Inertial navigation systems with geodetic applications*. Walter de Gruyter, 2012.

[66] A. M. Sabatini, "Estimating three-dimensional orientation of human body parts by inertial/magnetic sensing," *Sensors*, vol. 11, no. 2, pp. 1489–1525, 2011.

[67] A. Jensen, C. Coopmans, and Y. Chen, "Basics and guidelines of complementary filters for small uas navigation," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 500–507, IEEE, 2013.

[68] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Complementary filter design on the special orthogonal group so (3)," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 1477–1484, IEEE, 2005.

[69] S. M. Rajesh, S. B. Sanman Bhargava, and M. K. Sivanathan, "Mission planning and waypoint navigation of a micro quad copter by selectable gps co-ordinates," *International Journal*, vol. 4, no. 4, 2014.

[70] C. Stachniss, "Slam course 2013 material," *Universidad de Friburgo*, 2013.

[71] L. B. Pupo, *Characterization of errors and noises in MEMS inertial sensors using Allan variance method*. PhD thesis, Universitat Politècnica de Catalunya. Escola Tècnica Superior d'Enginyeria, 2016.

[72] D. Yang, Z. You, B. Li, W. Duan, and B. Yuan, "Complete tri-axis magnetometer calibration with a gyro auxiliary," *Sensors*, vol. 17, no. 6, p. 1223, 2017.

[73] "I2c-bus specification and user manual," *NXP Semiconductors*, 2012.

[74] "Spi block guide v03. 06," *Freescale Semiconductor*, 2000.

[75] B. Hu, E. Rouse, and L. Hargrove, "Benchmark datasets for bilateral lower-limb neuromechanical signals from wearable sensors during unassisted locomotion in able-bodied individuals," *Frontiers in Robotics and AI*, vol. 5, p. 14, 2018.

[76] "Serial port bluetooth module (master/slave) : Hc-05," *ITEAD INTELLIGENT SYSTEMS CO., LTD*.

[77] "Bluetooth core specification," *Bluetooth SIG, Inc*.

[78] "Atmega328p datasheet," *Atmel*.

[79] "Tp5100 datasheet," *Microlab*.

[80] "Lm7805 datasheet," *Fairchild Semiconductors*.

[81] "Ld1117v33 datasheet," *ST Microelectronics*.

[82] A. Hussen and I. Jleta, "Low-cost inertial sensors modeling using allan variance," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, no. 5, pp. 1237–1242, 2015.

[83] V. Vukmirica, I. Trajkovski, and N. Asanovic, "Two methods for the determination of inertial sensor parameters," *methods*, vol. 3, no. 1, 2018.

[84] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using allan variance," *IEEE Transactions on instrumentation and measurement*, vol. 57, no. 1, pp. 140–149, 2007.

[85] A. D'Alessandro, G. Vitale, S. Scudero, R. D'Anna, A. Costanza, A. Fagiolini, and L. Greco, "Characterization of mems accelerometer self-noise by means of psd and allan variance analysis," in *2017 7th IEEE International workshop on advances in sensors and interfaces (IWASI)*, pp. 159–164, IEEE, 2017.

[86] A. Saxena, G. Gupta, V. Gerasimov, and S. Ourselin, "In use parameter estimation of inertial sensors by detecting multilevel quasi-static states," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pp. 595–601, Springer, 2005.

[87] W. T. Vetterling, W. H. Press, S. A. Teukolsky, and B. P. Flannery, *Numerical recipes: example book C (The Art of Scientific Computing)*. Press Syndicate of the University of Cambridge, 1992.

[88] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[89] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE international conference on rehabilitation robotics*, pp. 1–7, IEEE, 2011.

[90] D. Trojaniello, S. Cao, A. Cereatti, and U. Della Croce, "Single imu gait event detection methods: sensitivity to imu positioning," *Gait Posture*, vol. 37, p. S24, 2013.

[91] J. McCamley, M. Donati, E. Grimpampi, and C. Mazza, "An enhanced estimate of initial contact and final contact instants of time using lower trunk inertial sensor data," *Gait & posture*, vol. 36, no. 2, pp. 316–318, 2012.

[92] X. Shao and C. Ma, "A general approach to derivative calculation using wavelet transform," *Chemometrics and Intelligent Laboratory Systems*, vol. 69, no. 1-2, pp. 157–165, 2003.

[93] G. Lee, F. Wasilewski, R. Gommers, K. Wohlfahrt, A. O'Leary, and H. Nahrstaedt, "Pywavelets-wavelet transforms in python," 2006.

[94] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[95] A. Szczesna and P. Pruszowski, "Model-based extended quaternion kalman filter to inertial orientation tracking of arbitrary kinematic chains," *SpringerPlus*, vol. 5, no. 1, p. 1965, 2016.

[96] F. Ferraris, U. Grimaldi, and M. Parvis, "Procedure for effortless in-field calibration of three-axis rate gyros and accelerometers," *Sensors and Materials*, vol. 7, pp. 311–311, 1995.

[97] I. Board, "Ieee standard specification format guide and test procedure for single-axis interferometric fiber optic gyros," *IEEE Std*, pp. 952–1997, 1998.

Código de Arduino

Main

```
#include <Wire.h>
#include <SPI.h>
#include "sensor.h"
#include "sensor_setup.h"
#include <NeoHWSerial.h>
#include <NeoHWSerial_private.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>

volatile char input;
volatile bool read_data = false;
const int INT1_PIN_DRDY = 9; // INT1 pin to D9 - attached to accel
const int INT2_PIN_DRDY = 8; // INT2 pin to D8 - attached to gyro

LSM9DS1 imu;

static void port_listener(uint8_t c){
  if (c == 's'){
    read_data = true;
  }
  else if (c == 'x'){
    read_data = false;
```

```
  }
  else{
    read_data = false;
  }
}

void setup() {
  // Disable interrupts for sutup
  cli();
  pinMode(INT1_PIN_DRDY, INPUT_PULLUP);
  pinMode(INT2_PIN_DRDY, INPUT_PULLUP);

  uint16_t status = initLSM9DS1(imu);
  configureLSM9DS1Interrupts(imu);

  NeoSerial.attachInterrupt(port_listener);
  NeoSerial.begin(115200);
  // Renable interrupts
  sei();
}

void loop() {
  if (read_data){
      // INT1 fires when new accelerometer data
      // is available.
      // It always fires after gyro
      // It's configured to be active LOW:
      if (digitalRead(INT1_PIN_DRDY) == LOW){
        // Get accelerometer data
        imu.readAccelRaw();
        // Get gyroscopo data
        imu.readGyroRaw();
        // Send header, accelerometer data and gyroscope data
        NeoSerial.write(imu.header, 4);
        NeoSerial.write(imu.dataAccel, 6);
        NeoSerial.write(imu.dataGyro, 6);
      }
  }
}
```

Varianza Allan

Teoría

Definición

La varianza allan es una técnica de análisis temporal que puede utilizarse para determinarlos procesos estocásticos que resultan en el ruido de una señal. A continuación se define con el caso de uso para un giróscopo a forma de ejemplo.

Suponiendo una señal con N muestras del giróscopo separadas por un tiempo τ_o, se forman clusters de largo $\tau_0, 2\tau_0, ..., k\tau_0$ con $k < N/2$. Para cada cluster se obtiene el promedio de las muestras. La varianza allan se define en función de la duración de los clusters $t = k\tau_0$.

La varianza allan se puede definir partiendo de la velocidad angular, $\Omega(t)$, o del ángulo:

$$\theta(t) = \int^{t} \Omega(t^{'})dt^{'} \tag{B.1}$$

La notación simplifica: $\theta_k = \theta(k\tau_0)$

La velocidad angular promedio entre los tiempos t_k y $t_{k+\tau}$ es:

$$\bar{\Omega}_k(\tau) = \frac{\theta_{t+m} - \theta_k}{\tau} \tag{B.2}$$

Donde $\tau = m\tau_0$.

La varianza allan se define como:

$$\sigma^2(\tau) = \frac{1}{2}\left\langle(\bar{\Omega}_{k+m} - \bar{\Omega}_k)^2\right\rangle = \frac{1}{2\tau^2}\left\langle(\theta_{k+2m} - 2\theta_{k+m} + \theta_k)^2\right\rangle \tag{B.3}$$

Donde $\langle\rangle$ es el *ensemble average*.

Finalmente, la varianza allan se estima como:

$$\sigma^2 = \frac{1}{2\tau^2(N-2m)}\sum_{k=1}^{N-2m}(\theta_{k+2m} - 2\theta_{k+m} + \theta_k)^2 \tag{B.4}$$

Relación con la PSD

La relación de la varianza allan con la densidad de potencia espectral está dada por

$$\sigma^2(\tau) = 4\int_0^\infty S_\Omega(f)\frac{\sin^4(\pi f\tau)}{(\pi f\tau)^2}df \tag{B.5}$$

La Tabla B.1 muestra las expresiones para obtener la varianza allan de los procesos de bias instability, angle random walk y rate random walk [97].

	Espectro de potencia	Relación con VA
Angle/Velocity random walk	$S_\Omega(f) = N^2$	$\sigma^2(T) = \frac{N^2}{T}$
Rate/Acceleration random walk	$S_\Omega(f) = \left(\frac{B}{2\pi}\right)^2 \frac{1}{f^2}$	$\sigma^2(T) = \frac{K^2 T}{3}$
Bias instability	$S_\Omega(f) = \begin{cases} \left(\frac{B^2}{2\pi}\right)\frac{1}{f} & ; f \leq f_0 \\ 0 & ; f > f_0 \end{cases}$	$\sigma(T) \to \sqrt{\frac{2\ln 2}{\pi}} B = 0{,}644B$

Tabla B.1: Aplicando la transformación del espectro de potencia hacia la VA [71, 63, 82], sobre la expresión para cada tipo de ruido (columna izquierda), se obtienen las relaciones correspondientes entre los coeficientes de cada uno con la VA (columna derecha).

La relación de los coeficientes N, K y B con la VA se ve en el gráfico log-log como rectas de pendiente $\frac{-1}{2}$ (Figura B.1), $\frac{1}{2}$ (Figura B.2) y 0 (Figura B.3) para el angle random walk, el rate random walk y el bias stability respectivamente.

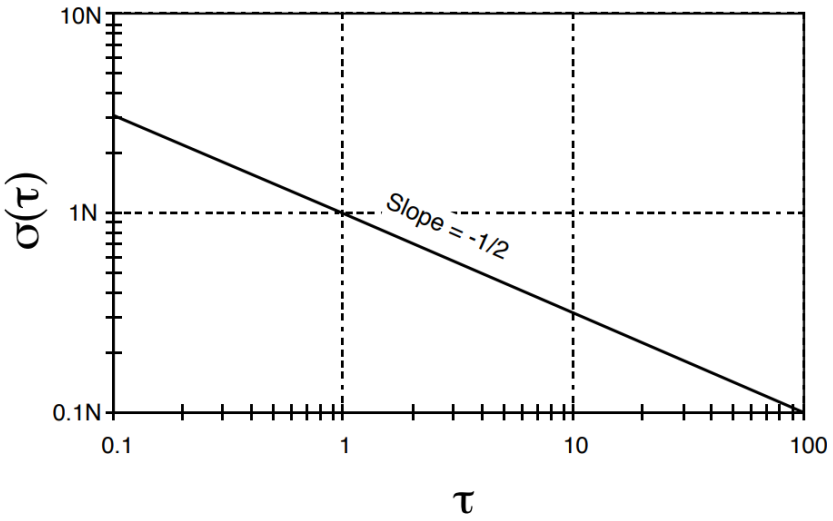


Figura B.1: Para el angle/velocity random walk, el valor del coeficiente se encuentra en el punto de la recta en donde $T = 1$.

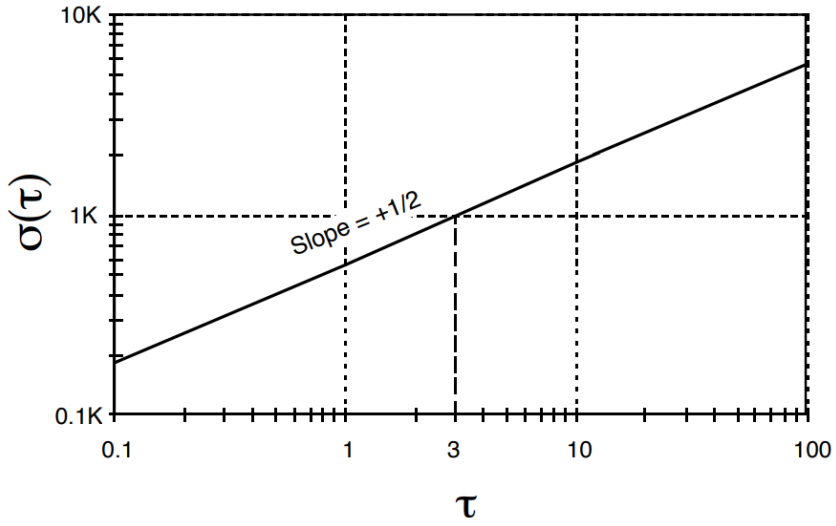


Figura B.2: Para el rate/acceleration random walk, el valor del coeficiente se encuentra en el punto de la recta en donde $T = 3$.

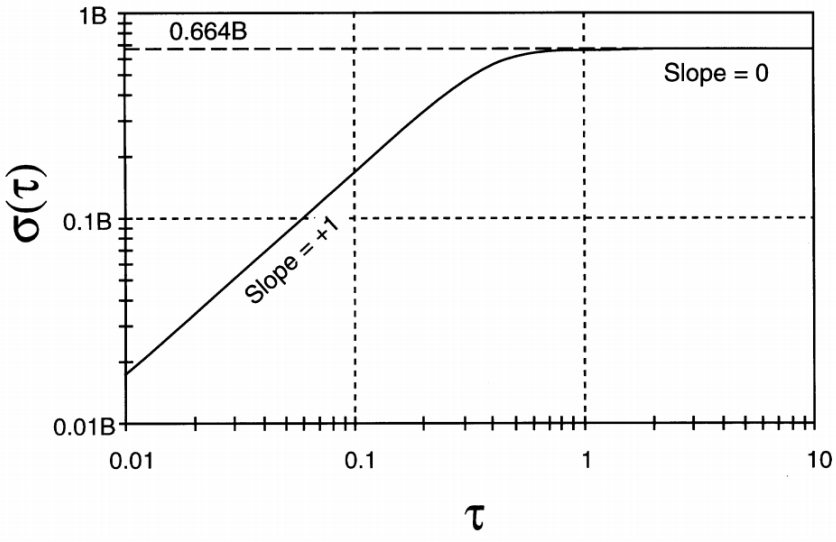


Figura B.3: Para el bias stability, el valor en donde la recta cruza al eje vertical, es igual 0.664 veces el coeficiente asociado.

Implementación

Código en Python

Se utilizaron dos scripts de Python 3. el primero para calcular la varianza allan de cada eje de cada sensor. El segundo para realizar el ajuste de las rectas en los gráficos obtenidos y obtener los parámetros de cada una.

Calculo de varianza allan

```
from ClassLibraries.Helpers.DataLoaders.
    get_file_or_directory_via_explorer import load_npy_file,
    get_directory_path, save_npy_file
from allantools import adev
from matplotlib.pyplot import loglog, grid, show, figure, title


def get_allan_variance():
    print("Select accelerometer data.")
    print("First X axis")
    ax = load_npy_file()
    print("Y axis")
    ay = load_npy_file()
    print("Z axis")
    az = load_npy_file()
    print("\nNow gyroscope data.")
    print("X axis")
    gx = load_npy_file()
    print("Y axis")
    gy = load_npy_file()
    print("Z axis")
    gz = load_npy_file()

    fs = float(input("What is the sampling rate of these samples ?...\t
    "))

    print("Ok, now select where do you want to save the allan variance
    data")
    directory_to_save = get_directory_path()

    print("Processing ax...")
    ax_avar = adev(data=ax, rate=fs, data_type='freq', taus='all')
    print("Processing ay...")
    ay_avar = adev(data=ay, rate=fs, data_type='freq', taus='all')
    print("Processing az...")
    az_avar = adev(data=az, rate=fs, data_type='freq', taus='all')
    print("Processing gx...")
```

```
    gx_avar = adev(data=gx, rate=fs, data_type='freq', taus='all')
    print("Processing gy...")
    gy_avar = adev(data=gy, rate=fs, data_type='freq', taus='all')
    print("Processing gz...")
    gz_avar = adev(data=gz, rate=fs, data_type='freq', taus='all')

    print("Saving data to " + directory_to_save + " ...")

    print("ax...")
    file_name = "ax_otaus"
    save_npy_file(ax_avar[0], file_name, directory_to_save)
    file_name = "ax_oavar"
    save_npy_file(ax_avar[1], file_name, directory_to_save)

    print("ay...")
    file_name = "ay_otaus"
    save_npy_file(ay_avar[0], file_name, directory_to_save)
    file_name = "ay_oavar"
    save_npy_file(ay_avar[1], file_name, directory_to_save)

    print("az...")
    file_name = "az_otaus"
    save_npy_file(az_avar[0], file_name, directory_to_save)
    file_name = "az_oavar"
    save_npy_file(az_avar[1], file_name, directory_to_save)

    print("gx...")
    file_name = "gx_otaus"
    save_npy_file(gx_avar[0], file_name, directory_to_save)
    file_name = "gx_oavar"
    save_npy_file(gx_avar[1], file_name, directory_to_save)

    print("gy...")
    file_name = "gy_otaus"
    save_npy_file(gy_avar[0], file_name, directory_to_save)
    file_name = "gy_avar"
    save_npy_file(gy_avar[1], file_name, directory_to_save)

    print("gz...")
```

```
    file_name = "gz_otaus"
    save_npy_file(gz_avar[0], file_name, directory_to_save)
    file_name = "gz_oavar"
    save_npy_file(gz_avar[1], file_name, directory_to_save)

    figure()
    title("Accelerometer")
    loglog(ax_avar[0], ax_avar[1])
    loglog(ay_avar[0], ay_avar[1])
    loglog(az_avar[0], az_avar[1])
    grid(which='major')
    grid(which='minor')
    figure()
    title("Gyroscope")
    loglog(gx_avar[0], gx_avar[1])
    loglog(gy_avar[0], gy_avar[1])
    loglog(gz_avar[0], gz_avar[1])
    grid(which='major')
    grid(which='minor')
    show()


if __name__ == '__main__':
    print("Running \"get_allan_variance()...\"")
    get_allan_variance()
```

Ajuste de rectas

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as optimize
from ClassLibraries.Helpers.DataLoaders.
   get_file_or_directory_via_explorer import load_npy_file


def fit_curve_to_allan_variance_plot(slope_to_fit: float = None):
    print("Select taus")
    taus = load_npy_file()
```

```
    print("Select samples")
    adev = load_npy_file()

    if slope_to_fit is None:
        slope_to_fit = float(input("Provide a slope to fit in the log
log plot\t"))

    if slope_to_fit == 0:
        def shape_to_fit(x, a):
            return a
    elif (slope_to_fit == 1):
        def shape_to_fit(x, a):
            return a * x
    elif (slope_to_fit == -1):
        def shape_to_fit(x, a):
            return a * x**-1
    elif (slope_to_fit == 0.5):
        def shape_to_fit(x, a):
            return a * x ** 0.5
    elif (slope_to_fit == -0.5):
        def shape_to_fit(x, a):
            return a * x**-0.5
    else:
        if (type(slope_to_fit) != 'float'):
            print("Slope provided is not a number")
        else:
            print("Slope not supported")
        raise NotImplemented

    plt.loglog(adev)
    plt.grid(which='major')
    plt.grid(which='minor')
    plt.show()

    start = int(input("Type the sample from which to start fitting\t"))
    end = int(input("And the last one ?\t"))
    plt.close()

    parameters, parameters_covariance = optimize.curve_fit(shape_to_fit
```

```
    , taus[start:end], adev[start:end], bounds=(0, np.inf))
48
49      print("Parameters obtained: ")
50      print(parameters)
51
52      plt.loglog(taus, adev)
53      plt.loglog(taus, [shape_to_fit(t, *parameters) for t in taus])
54
55      plt.grid(which='major')
56      plt.grid(which='minor')
57      plt.show()
58
59
60  if __name__ == '__main__':
61      print("Running \"fit_curve_to_allan_variance_plot()...\"")
62      fit_curve_to_allan_variance_plot()
```

Librería AllanTools

Para la implementación del código se utilizó una librería bajo la Licencia Pública General Reducida de GNU

Su documentación se encuentra en https://allantools.readthedocs.io/en/latest/index.html. Posee no sólo la implementación de la varianza allan, sino también otras modificaciones, entre ellas la "Overlapping allan Variance".

En ella se encuentra el cálculo que realiza el método utilizado que se corresponde con la ecuación B.4.

Calibración

Detector de estado estático

```
import numpy as np


class static_detector_saxena_implementation:
    def __init__(self, threshold: float = 0.1, rms_a: float = 1, rms_g: float = 60):
        '''
        Static detector class constructor. Implementation for Saxena et. al. 2005 quasi-static state detector
        :param threshold: to compare gamma and determine state
        :param rms_a: RMS for accelerometer in m/s**2
        :param rms_g: RMS for gyroscope in deg/seg
        '''
        self.threshold = threshold
        self.variances_defined = False
        self.variances = np.array([])
        self.rms_a = rms_a
        self.rms_g = rms_g

    def define_variances(self, samples: np.array([], dtype='float')):
        '''
        Define variances for algorithm given a sequence of samples from a stationary measurement
```

```
        :param samples: (n axis, window length) shape numpy float array
        :return: True or False meaning success
        '''
        try:
            self.variances = np.var(samples, axis=1)
            self.variances_defined = True
            return True
        except:
            return False

    def calculate_gamma(self, window: np.array([], dtype='float')):
        '''
        Calculate gamma for current samle
        :param window: (n axis, 0) shape numpy float array
        :return: gamma value or False if variances are not defined
        '''
        if self.variances_defined:
            sample = np.var(window, axis=1)
            acc = np.sum(sample[:3]**2/self.variances[:3])/(self.rms_a
**2)
            gyro = np.sum(sample[3:]**2/self.variances[3:])/(self.rms_g
**2)
            return (acc + gyro)/6
        else:
            print("Variance for sensors is not defined.")
            return False

    def get_state(self, window: np.array([], dtype='float'),
lowpass_filter_callback):
        '''
        Get static or dynamic state for current sample
        :param window: (n axis, 0) shape numpy float array
        :param lowpass_filter_callback: low pass filtering function
        :return: True or False for stationary or dynamic state
        '''
        low_passed_gamma = lowpass_filter_callback(self.calculate_gamma
(window))
        return low_passed_gamma < self.threshold
```

Propagación de orientación

```
import numpy as np


class QuaternionIntegrator:
    @staticmethod
    def second_order_algorithm_step(q0, w, fs=238.):
        dt = 1./fs
        do = w * dt
        modulus = np.sqrt(np.dot(do.T, do))
        b = np.array([[  0,     do[0],  do[1],   do[2]],
                      [-do[0],    0,    do[2],  -do[1]],
                      [-do[1], -do[2],    0,     do[0]],
                      [-do[2],  do[1],  -do[0],    0]])
        argument = 0.5 * modulus * np.pi / 180.
        result = np.cos(argument) * np.identity(4) + np.sin(argument) *
    b / modulus
        return np.dot(result, q0)

    def second_order_algorithm(self, q0, w, fs):
        q = [q0]
        for i in range(len(w)):
            q.append(self.second_order_algorithm_step(q[-1], w[i], fs))
        return np.asarray(q)

    def quaternion_sequence_from_angular_rates(self, gx, gy, gz, fs
   =238.):
        angular_rate = []
        for i in range(len(gx)):
            angular_rate.append([gx[i], gy[i], gz[i]])
        angular_rate = np.asarray(angular_rate)
        return self.second_order_algorithm([1., 0., 0., 0.],
   angular_rate, fs)
```

Optimizador

```
from scipy.optimize import minimize
```

```
import numpy as np

__MAX_ITERATIONS__ = 5000


class Minimizer:
    def __init__(self, func, w0):
        self.func = func
        self.weights0 = w0
        self.weights = []
        self.w_history = []
        self.cost = np.inf
        self.result = []
        return None

    def optimize_accelerometer_weigths(self, features, labels):
        initial_simplex = [self.weights0]
        for n in range(len(self.weights0)):
            aux = self.weights0[:]
            aux[n] = aux[n] + (np.random.rand(1)[0]*2-1)*2
            initial_simplex.append(aux)
        result = minimize(self.func, x0=self.weights0, args=(features,
    labels), method='Nelder-Mead',
                          options={'disp': True, 'maxiter':
    __MAX_ITERATIONS__, 'initial_simplex': initial_simplex,
                                   'adaptive': False})
        self.weights = result.x
        self.cost = result.fun
        self.result = result

    def optimize_gyroscope_weigths(self, features, labels, states):
        initial_simplex = [self.weights0]
        for n in range(len(self.weights0)):
            aux = self.weights0[:]
            aux[n] = aux[n] + (np.random.rand(1)[0]*2-1)*1
            initial_simplex.append(aux)
        result = minimize(self.func, x0=self.weights0, args=(features,
    labels, states), method='Nelder-Mead',
                          options={'disp': True, 'maxiter':
```

```
    __MAX_ITERATIONS__, 'initial_simplex': initial_simplex,
38                                      'adaptive': False})
39
40          self.weights = result.x
41          self.cost = result.fun
42          self.result = result
```

Madgwick

```
import numpy as np


class MadgwickIMUFilter:
    def __init__(self, q1=1.0, q2=0.0, q3=0.0, q4=0.0, fs=238.0, roll
   =0.0, pitch=0.0, yaw=0.0, beta=0.049):

        self.q1 = q1
        self.q2 = q2
        self.q3 = q3
        self.q4 = q4
        self.fs = fs
        self.beta = beta
        self.toRad = np.pi/180.

    def updateIMU(self, meassure):
        ax = meassure[0]
        ay = meassure[1]
        az = meassure[2]
        gx = meassure[3]
        gy = meassure[4]
        gz = meassure[5]

        # Convert gyroscope degrees/sec to radians/sec
        gx *= self.toRad
        gy *= self.toRad
        gz *= self.toRad
```

```
27
28          # Auxiliary variables to avoid repeated arithmetic
29          q1_half = 0.5 * self.q1
30          q2_half = 0.5 * self.q2
31          q3_half = 0.5 * self.q3
32          q4_half = 0.5 * self.q4
33          q1_double = 2.0 * self.q1
34          q2_double = 2.0 * self.q2
35          q3_double = 2.0 * self.q3
36
37          # Normalize accelerometer measurement
38          norm = (ax ** 2 + ay ** 2 + az ** 2) ** 0.5
39          ax /= norm
40          ay /= norm
41          az /= norm
42
43          # Compute the objective function and the Jacobian
44          f_1 = q2_double * self.q4 - q1_double * self.q3 - ax
45          f_2 = q1_double * self.q2 + q3_double * self.q4 - ay
46          f_3 = 1.0 - q2_double * self.q2 - q3_double * self.q3 - az
47
48          J_11or24 = q3_double
49          J_12or23 = 2.0 * self.q4
50          J_13or22 = q1_double
51          J_14or21 = q2_double
52          J_32 = 2.0 * J_14or21
53          J_33 = 2.0 * J_11or24
54
55          # Compute the gradient (matrix multiplication)
56          q1_HatDot = J_14or21 * f_2 - J_11or24 * f_1
57          q2_HatDot = J_12or23 * f_1 + J_13or22 * f_2 - J_32 * f_3
58          q3_HatDot = J_12or23 * f_2 - J_33 * f_3 - J_13or22 * f_1
59          q4_HatDot = J_14or21 * f_1 + J_11or24 * f_2
60
61          # Normalise the gradient
62          norm = (q1_HatDot ** 2 + q2_HatDot ** 2 + q3_HatDot ** 2 +
    q4_HatDot ** 2) ** 0.5
63          q1_HatDot /= norm
64          q2_HatDot /= norm
```

```
        q3_HatDot /= norm
        q4_HatDot /= norm

        # Compute the quaternion derivative measured by gyroscopes
        q1_DotOmega = -q2_half * gx - q3_half * gy - q4_half * gz
        q2_DotOmega = q1_half * gx + q3_half * gz - q4_half * gy
        q3_DotOmega = q1_half * gy - q2_half * gz + q4_half * gx
        q4_DotOmega = q1_half * gz + q2_half * gy - q3_half * gx

        # Compute then integrate the estimated quaternion derivative
        self.q1 += (q1_DotOmega - (self.beta * q1_HatDot)) / self.fs
        self.q2 += (q2_DotOmega - (self.beta * q2_HatDot)) / self.fs
        self.q3 += (q3_DotOmega - (self.beta * q3_HatDot)) / self.fs
        self.q4 += (q4_DotOmega - (self.beta * q4_HatDot)) / self.fs

        # Normalise quaternion
        norm = (self.q1 ** 2 + self.q2 ** 2 + self.q3 ** 2 + self.q4 **
    2) ** 0.5
        self.q1 /= norm
        self.q2 /= norm
        self.q3 /= norm
        self.q4 /= norm

    def getQuaternion(self):
        return np.array([self.q1, self.q2, self.q3, self.q4])
```

Scripts

Eventos

```
from ClassLibraries.Helpers.DataLoaders.
    get_file_or_directory_via_explorer import load_npy_file,
    get_directory_path
from ClassLibraries.DSP.GaitParameters.get_gait_events import
    get_gait_events
import matplotlib.pyplot as plt
import numpy as np
from ClassLibraries.DSP.Filtering.ComplementaryFilter.Madgwick.
    madgwick_imu import MadgwickIMUFilter
from ClassLibraries.Helpers.AngleRepresentations.quaternion2rot import
    quaternion2rot
from ClassLibraries.Helpers.AngleRepresentations.quaternion2rpy import
    quaternion2rpy
from scipy.signal import butter, filtfilt


# Madgwick
Madgwick = MadgwickIMUFilter()
gyro_error = np.sqrt(0.0481)  # 0.0481 es random walk de velocidad
    angular
Madgwick.beta = np.sqrt(3./4.) * np.pi / 180. * gyro_error

# par metros
```

```
fs = 238
scales = 70
leg_length = float(input("Ingrese el largo de pierna en cent metros: "
    ))

# cargo datos
print("Seleccione el directorio donde est n guardados los datos.")
directory = get_directory_path()
samples = np.load(directory + "/samples.npy")
biases = np.load(directory + "/biases.npy")
sensitivities = np.load(directory + "/sensitivities.npy")

# aplico par metros de calibraci n
calibrated = ((samples.T - np.hstack((biases[0], biases[1]))).T *
    sensitivities)

# aplico Madgwick
q = np.zeros((len(calibrated[0]), 4))
rpy = np.zeros((len(calibrated[0]), 3))
rot = np.zeros((len(calibrated[0]), 3, 3))
for i in range(len(calibrated[0])):
    sample = calibrated[:, i]
    Madgwick.updateIMU(sample)
    q[i] = Madgwick.getQuaternion()
    rot[i] = quaternion2rot(q[i])
    rpy[i] = quaternion2rpy(q[i])
    acceleration = np.dot(rot[i], sample[:3])
    vel_angular = np.dot(rot[i], sample[3:])
    calibrated[:3, i] = acceleration
    calibrated[3:, i] = vel_angular

# tomo aceleraci n en z
vertical_acceleration = calibrated[2]

# detecci n de eventos
initial_contacts, final_contacts, transform, transform2 =
    get_gait_events(
    data=vertical_acceleration,
    fs=fs,
```

```
    scales=scales,
    return_transforms=True
)

# descarto contactos finales al inicio y contactos iniciales al final.
final_contacts = np.array([i for i in final_contacts if i >
    initial_contacts[0]])
initial_contacts = np.array([i for i in initial_contacts if i <
    final_contacts[-1]])

if len(final_contacts) > len(initial_contacts):
    final_contacts = final_contacts[:-1]
elif len(initial_contacts) > len(final_contacts):
    initial_contacts = initial_contacts[1:]


print("Contactos iniciales: {}".format(len(initial_contacts)))
print("Contactos finales: {}".format(len(final_contacts)))

plt.figure()
plt.plot(vertical_acceleration)
plt.plot(transform)
plt.plot(transform2)
plt.plot(initial_contacts, transform[initial_contacts], 'k^')
plt.plot(final_contacts, transform2[final_contacts], 'rv')
plt.title("Transformadas Wavelet")
plt.grid()
plt.xlabel("Muestras")
plt.ylabel("$metros/s^2$")
plt.legend(["Aceleraci n vertical", "- T1", "T2", "CI", "CF"])

# resto la gravedad y paso de "g" a metros sobre segundo cuadrado
real_acceleration = (vertical_acceleration - 1) * 9.8

plt.figure()
plt.plot(real_acceleration)
plt.plot(initial_contacts, real_acceleration[initial_contacts], 'k^')
plt.plot(final_contacts, real_acceleration[final_contacts], 'bv')
plt.title("Aceleraci n")
```

```
plt.grid()
plt.xlabel("Muestras")
plt.ylabel("$metros/s^2$")
plt.legend(["Aceleraci n vertical", "CI", "CF"])

# filtro Butterworth para la aceleraci n
order = 4
cutoff = 0.4
nyq = 0.5 * fs
cutoff = cutoff / nyq
b, a = butter(order, cutoff, btype='highpass')
filtered_acceleration = filtfilt(b, a, real_acceleration)

plt.figure()
plt.plot(real_acceleration)
plt.plot(filtered_acceleration)
plt.title("Aceleraci n Filtrada")
plt.grid()
plt.xlabel("Muestras")
plt.ylabel("$metros/s^2$")
plt.legend(["Aceleraci n vertical", "Aceleraci n filtrada", "CF"])

# filtro Butterworth para la velocidad
real_velocity = np.cumsum(filtered_acceleration / fs)
real_velocity = filtfilt(b, a, real_velocity)

plt.figure()
plt.plot(real_velocity)
plt.plot(initial_contacts, real_velocity[initial_contacts], 'k^')
plt.plot(final_contacts, real_velocity[final_contacts], 'bv')
plt.title("Velocidad")
plt.grid()
plt.xlabel("Muestras")
plt.ylabel("$metros/s$")
plt.legend(["Velocidad vertical", "CI", "CF"])

# metros a cent metros
real_displacement = np.cumsum(real_velocity / fs) * 100

```

```
129 plt.figure()
130 plt.plot(real_displacement)
131 plt.plot(initial_contacts, real_displacement[initial_contacts], 'k^')
132 plt.plot(final_contacts, real_displacement[final_contacts], 'bv')
133 plt.title("Desplazamiento")
134 plt.grid()
135 plt.xlabel("Muestras")
136 plt.ylabel("cent metros")
137 plt.legend(["Desplazamiento vertical", "CI", "CF"])
138
139
140 # defino funci n para obtener promediar los  ngulos  entre c clos de
    marcha
141 def average_cycles(angles, ics):
142     resampled_cycles = []
143     t_cycle = np.linspace(0, 100, 101)
144     for i in range(0, len(ics)-2, 2):
145         cycle = angles[ics[i]:ics[i+2]]
146         t_angles = np.linspace(0, 100, len(cycle))
147         resampled_cycles.append(np.interp(t_cycle, t_angles, cycle))
148     return np.array(resampled_cycles)
149
150
151 # calculo y muestro  ngulos
152 ox = -rpy[:, 0] * 180./np.pi
153 oy = -rpy[:, 1] * 180./np.pi
154 oz = -rpy[:, 2] * 180./np.pi
155
156 plt.figure()
157 plt.plot(oy)
158 plt.plot(initial_contacts, oy[initial_contacts], 'k^')
159 plt.plot(final_contacts, oy[final_contacts], 'bv')
160 plt.grid()
161 plt.xlabel("Muestras")
162 plt.ylabel("$grados$")
163 plt.title("Inclinaci n de pelvis")
164 plt.legend(["Inclinaci n", "CI", "CF"])
165 plt.figure()
166 plt.plot(ox)
```

```
167 plt.plot(initial_contacts, ox[initial_contacts], 'k^')
168 plt.plot(final_contacts, ox[final_contacts], 'bv')
169 plt.grid()
170 plt.xlabel("Muestras")
171 plt.ylabel("$grados$")
172 plt.title("Abducci n de pelvis")
173 plt.legend(["Abducci n", "CI", "CF"])
174 plt.figure()
175 plt.plot(oz)
176 plt.plot(initial_contacts, oz[initial_contacts], 'k^')
177 plt.plot(final_contacts, oz[final_contacts], 'bv')
178 plt.grid()
179 plt.xlabel("Muestras")
180 plt.ylabel("$grados$")
181 plt.title("Rotaci n de pelvis")
182 plt.legend(["Rotaci n", "CI", "CF"])
183
184 # descarto el primer paso
185 ox = average_cycles(ox, initial_contacts[1:])
186 oz = average_cycles(oz, initial_contacts[1:])
187 oy = average_cycles(oy, initial_contacts[1:])
188
189 plt.figure()
190 plt.plot(np.mean(ox[1:], axis=0))
191 plt.grid()
192 plt.ylabel("grados")
193 plt.xlabel("% del c clo de marcha")
194 plt.title("Abducci n promedio")
195
196 plt.figure()
197 plt.plot(np.mean(oz[1:], axis=0))
198 plt.grid()
199 plt.ylabel("grados")
200 plt.xlabel("% del c clo de marcha")
201 plt.title("Rotacion promedio")
202
203 plt.figure()
204 plt.plot(np.mean(oy[1:], axis=0))
205 plt.grid()
```

```
206 plt.ylabel("grados")
207 plt.xlabel("% del c clo de marcha")
208 plt.title("Inclinacion promedio")
209
210 # calculo el largo de todos los pasos
211 step_lengths = []
212 for i in range(len(initial_contacts)-2):
213     start_swing = initial_contacts[i]
214     end_swing = initial_contacts[i+1]
215     dif = np.max(real_displacement[start_swing:end_swing]) - np.min(
    real_displacement[start_swing:end_swing])
216     amplitude = (dif**2)**0.5
217     step_lengths.append(2 * np.sqrt(2 * leg_length * amplitude -
    amplitude**2))
218
219
220 # divido pasos de una pierna y de la otra
221 right_ics = initial_contacts[0::2]
222 left_ics = initial_contacts[1::2]
223 right_fcs = final_contacts[1::2]
224 left_fcs = final_contacts[0::2]
225
226 if len(right_ics) > len(right_fcs):
227     right_ics = right_ics[:-1]
228 elif len(right_fcs) > len(right_ics):
229     right_fcs = right_fcs[1:]
230 if len(left_ics) > len(left_fcs):
231     left_ics = left_ics[:-1]
232 elif len(left_fcs) > len(left_ics):
233     left_fcs = left_fcs[1:]
234
235 # calculo los par metros espacio-temporales
236 mean_step_length = np.mean(step_lengths)
237 cycle_duration = np.mean(np.hstack((np.diff(right_ics), np.diff(
    right_fcs), np.diff(left_ics), np.diff(left_fcs)))) / fs
238 stance_time = np.mean(np.hstack((right_fcs - right_ics, left_fcs[1:] -
    left_ics[:-1]))) / fs
239 full_steps = (len(initial_contacts)-1)
240 time = ((initial_contacts[-1] - initial_contacts[0])/fs)
```

```
path = mean_step_length * full_steps
speed = path / time
cadence = full_steps / time * 60
cycle_duration = 2 / cadence * 60
stance_percentage = stance_time / cycle_duration * 100.

print("Largo de paso: {}".format(mean_step_length))
print("Tiempo de c clo: {}".format(cycle_duration))
print("Fase de sustentaci n: {} ({}%)".format(stance_time,
    stance_percentage))
print("Fase de balanceo: {} ({}%)".format(cycle_duration-stance_time,
    100-stance_percentage))
print("Cadencia: {}".format(cadence))
print("Recorrido: {}".format(path))
print("Velocidad: {}".format(speed))
print("Tiempo: {}".format(time))

plt.show()
```

Captura

```
from ClassLibraries.Helpers.DataLoaders.
    get_file_or_directory_via_explorer import save_npy_file,
    get_directory_path
from ClassLibraries.DSP.Filtering.ComplementaryFilter.Madgwick.
    madgwick_imu import MadgwickIMUFilter
from ClassLibraries.Helpers.AngleRepresentations.quaternion2rot import
    quaternion2rot
from Modules.Communication.imu_port import *
from Modules.Driver.driver import *
from Modules.Driver.sensor import *
from Modules.Driver.storage import *
from vpython import scene, arrow, color, vector
import msvcrt


def check_keyboard_input(character, port):
    if character == 's':
```

```
14          print("Comenzar captura")
15          port.write_byte(character)
16          return True
17      elif character == 'x':
18          print("Finalizando")
19          port.write_byte(character)
20          return False
21      else:
22          print("Comando no comprendido")
23          return True  # None
24
25
26  print("Seleccione el directorio donde est n guardados los datos.")
27  directory = get_directory_path()
28  biases = np.load(directory + "/biases.npy")
29  sensitivities = np.load(directory + "/sensitivities.npy")
30
31  input("Presione una tecla y luego enter para comenzar.")
32  print("Iniciando...")
33
34  ####################################
35  cSensor = Sensor()
36  cDriver = Driver()
37  cStorage = Storage()
38  cMadgwick = MadgwickIMUFilter()
39  gyro_error = np.sqrt(0.0481)  # 0.0481 es random walk de velocidad
        angular
40  cMadgwick.beta = np.sqrt(3./4.) * np.pi / 180. * gyro_error
41  cIMU_port = IMU_port(connect_on_startup=True)
42  ####################################
43
44  # inicializo visualizaci n
45  scene.range = 2
46  toRad = 2*np.pi/360.0
47  toDeg = 1 / toRad
48
49  scene.forward = vector(0, 0, -1)
50  scene.width = 600
51  scene.height = 600
```

```
52
53 zarrow = arrow(length=1.5, shaftwidth=.05, color=color.blue, axis=
       vector(0, 1, 0))
54 yarrow = arrow(length=1.5, shaftwidth=.05, color=color.red, axis=vector
       (1, 0, 0))
55 xarrow = arrow(length=1.5, shaftwidth=.05, color=color.green, axis=
       vector(0, 0, 1))
56
57 z = arrow(length=1, shaftwidth=.1, color=color.blue, axis=vector(0, 1,
       0))
58 y = arrow(length=1, shaftwidth=.1, color=color.red, axis=vector(1, 0,
       0))
59 x = arrow(length=1, shaftwidth=.1, color=color.green, axis=vector(0, 0,
        1))
60
61
62 cIMU_port.write_byte('s')
63 cIMU_port.serial_port.reset_input_buffer()
64 # busco header para inicial captura
65 if cDriver.catch_sample_header(cIMU_port, cSensor):
66     print("Capturando")
67     capturing = True
68     while capturing:
69         # chequeo que haya datos en el buffer
70         while cIMU_port.serial_port.in_waiting < cSensor.package_length
       :
71             pass
72         data = cIMU_port.read_bytes(cSensor.package_length - cSensor.
       header_length)
73
74         # descarto pr ximo header
75         cIMU_port.read_bytes(cSensor.header_length)
76
77         # proceso muestra
78         data = cDriver.convert_from_high_low_byte(data, 6)
79         cStorage.store_sample(data)  # save raw data
80         data = ((data - biases) * sensitivities.T)[0]
81
82         cMadgwick.updateIMU(data)
```

```
        q = cMadgwick.getQuaternion()
        rot = quaternion2rot(q).T
        aux_rot = np.roll(rot, -1, axis=1)

        z.axis = vector(aux_rot[2, 0], aux_rot[2, 1], aux_rot[2, 2])
        y.axis = vector(aux_rot[1, 0], aux_rot[1, 1], aux_rot[1, 2])
        x.axis = vector(aux_rot[0, 0], aux_rot[0, 1], aux_rot[0, 2])

        z.length = 1
        y.length = 1
        x.length = 1

        key_pressed = msvcrt.kbhit()
        if key_pressed:
            print("Tecla presionada!")
            char = msvcrt.getch().decode("utf-8")
            capturing = check_keyboard_input(char, cIMU_port)

IMU_port.disconnect()

print("Guardando muestras...")
save_npy_file(cStorage.get_samples(), 'captured_samples', directory)
print("Proceso finalizado!")
```