UNIVERSIDAD DE VALLADOLID

FACULTAD DE MEDICINA

ESCUELA DE INGENIERÍAS INDUSTRIALES

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA BIOMÉDICA

Dispositivos wearables. Aplicación práctica.

Autor: **José María Pallarés García**

Tutor: **José Manuel González de la Fuente**

Valladolid, 1 de julio de 2025

TÍTULO:	**Dispositivos wearables. Aplicación práctica.**
AUTOR/A:	**D. José María Pallarés García**
TUTOR/A:	**D. José Manuel González de la Fuente**
DEPARTAMENTO:	**Departamento de Tecnología Electrónica**

TRIBUNAL

PRESIDENTE:	**D.ª Isabel del Valle González**
SECRETARIO:	**D.ª Cristina Pérez Barreiro**
VOCAL:	**D. José Manuel González de la Fuente**
SUPLENTE 1:	**D. Juan Carlos Fraile Marinero**
SUPLENTE 2:	**D. Javier Pérez Turiel**

FECHA:
CLASIFICACIÓN

Resumen

El presente trabajo se basa en el estudio y aplicación práctica de la tecnología *wearable*. En primera instancia, se presenta una revisión del estado del arte, abordando la evolución histórica de los dispositivos vestibles, su impacto económico creciente y su clasificación funcional y anatómica, destacando aplicaciones en entornos médicos.

La parte práctica del trabajo consiste en el diseño e implementación de un prototipo de goniómetro *wearable*, destinado a monitorizar el rango de movimiento de la rodilla durante procesos de rehabilitación. El sistema combina una unidad de procesamiento, sensores inerciales, un vibrador para permitir la retroalimentación y un módulo Bluetooth BLE, integrado en un dispositivo compacto y de bajo coste. La comunicación con el usuario se realiza mediante una interfaz gráfica desarrollada en la plataforma Universal Windows Platform (UWP).

El prototipo permite visualizar en tiempo real los ángulos articulares y generar alertas vibratorias cuando se superan ciertos umbrales definidos por el profesional, fomentando la autonomía del paciente y la prevención de movimientos lesivos. Además, se analizan los aspectos anatómicos de la rodilla, los ejercicios habituales y el impacto psicológico durante la rehabilitación.

En conjunto, se propone una solución accesible para el seguimiento personalizado de la recuperación funcional, integrando electrónica, biomecánica y software en una única herramienta biomédica.

Palabras Clave: tecnología wearable, instrumentación biomédica, sensores, monitorización**.**

Abstract

This project is based on the study and practical application of wearable technology. It begins with a review of the state of the art, addressing the historical evolution of wearable devices, their growing economic impact, and their functional and anatomical classification, with a particular focus on applications in medical environments.

The practical section involves the design and implementation of a wearable goniometer prototype intended to monitor the range of motion of the knee during rehabilitation processes. The system integrates a processing unit, inertial sensors, a vibration motor for haptic feedback, and a Bluetooth BLE module, all embedded in a compact and low-cost device. Communication with the user is achieved through a graphical interface developed on the Universal Windows Platform (UWP).

The prototype enables real-time visualization of joint angles and generates vibration alerts when certain thresholds, defined by healthcare professionals, are exceeded. This promotes patient autonomy and helps prevent harmful movements. Additionally, the anatomical aspects of the knee, common rehabilitation exercises, and the psychological impact of the recovery process are also analyzed.

Overall, this work proposes an accessible solution for personalized monitoring of functional recovery, combining electronics, biomechanics, and software in a single biomedical tool.

Keywords: wearable technology, biomedical instrumentation, sensors, monitoring.

Agradecimientos

Se me hace complicado escribir estas palabras, no por falta de deseo, sino por el escepticismo que me genera pensar que este documento supone la finalización de una faceta de mi vida. Este proyecto es la máxima expresión de agradecimiento a todo aquel que me ha estado acompañando en estos seis años repletos de frustraciones, hastíos y, de vez en cuando, momentos dichosos.

Muchas gracias, José Manuel González, mi tutor, por darme alas en este trabajo. Considero que no se imagina lo importante que ha sido tener su apoyo. Tras conversaciones con varios profesores e infinidad de negativas, usted fue el único capaz de sentarse conmigo y escuchar el catálogo de ideas que le presentaba. Me dio su confianza y apoyo, algo que le agradeceré siempre que pueda. Espero que, aunque sea mínimo, sienta orgullo del trabajo que hemos conseguido.

Gracias a Íñigo del Río por estar y apostar por mí, incluso cuando ni yo me reconocía; te presentabas con una cerveza y me escuchabas. Gracias, Nicolás Calvo, por las tardes en Discord, las conversaciones incansables sobre qué personaje hacernos y por la fraternidad amistosa que hemos creado y que protegeré a lo largo de los años. Gracias a Alberto Velasco por ser mi confidente; sé que eres capaz de hacer grandes cosas. Gracias a Diego Delgado, por más comidas junto a ti. Gracias a vosotros, que habéis estado presentes en mis momentos más bajos y me habéis dado la confianza en momentos en los que se me hacía imposible mirarme.

De la carrera, agradecer a Sara Cruz. Tu aparición en mi vida dio sentido al ir a clases. No te me estreses tanto, cada cosa llega en su momento. Únicamente debemos crear las oportunidades y esperar pacientes.

A mi familia y a cada uno de sus integrantes, gracias por expresarme vuestro amor incondicional. Sé que empiezo una nueva etapa, pero no pienso ser menos que vosotros; tarde o temprano, os pienso superar. Me habéis enseñado el peso que es cargar el apellido Pallarés y voy a estar a la altura. Estoy profundamente orgulloso de pertenecer a la unidad familiar. Gracias por no darme la espalda.

Gracias, Isabel Raposo, la mujer que provoca que se me acelere el corazón. Por fin me tendiste la mano y no pienso rechazarla. Eres todo lo que está bien conmigo. Espero poder cumplir tus sueños.

Gracias, maestro Toriyama. Algún día podré acercarme a Piccolo.

“Hay que trabajar, hay que aprender, hay que comer, hay que descansar y también hay que jugar. Esas son las bases del entrenamiento del maestro Roshi para tener una buena condición”.

Tabla de contenido

Tabla de ilustraciones

Lista de acrónimos.

AR: *Augmented Reality*

ATM: *Automated Teller Machine*

BLE: *Bluetooth Low Energy*

CAGR: *Compound Annual Growth Rate*

CKC: *Closed Kinetic Chain*

CSSYs: *Core-Sheath Sensor Yarn*

DMP: *Digital Motion Processor*

FIFO: *First in, First Out*

GAP: Generic Access Profile

GATT: Generic Attribute Profile

HMD: *Head Mounted Display*

HP: *Hewlett Packard*

Hz: *Hercios*

I2C: *Inter-Integrated Circuit*

IDC: *International Data Corporation*

IMU: *Intertial Measurement Unit*

MEMS: *MicroElectroMechanical System*

MISO: *Master In, Slave Out*

MIT: *Massachusetts Institute of Technology*

MR: *Mixed Reality*

OA: *Osteoartritis*

OKC: *Open Kinetic Chain*

PWM: *Pulse Width Modulation*

SCL: *Serial Clock*

SDA: *Serial Data*

SPI: *Serial Peripheral Interface*

SS: *Slave Select*

ToF: *Time of Flight*

UUID: Universally Unique Identifier

UWP: *Universal Windows Platform*

VL: *Vasto Lateral*

VMO: *Vasto Medio Oblicuo*

VR: *Virtual Reality*

XR: *Extended Reality*

Estado del Arte

1.1 Introducción a la Tecnología Wearable

El dispositivo *wearable* es toda aquella electrónica integrada en la ropa y otros accesorios que pueden ser utilizados de manera cómoda. Dicha definición, engloba a aquellos instrumentos que pueden ser fácilmente cambiados e incluso versiones invasivas como los "*Smart-tattoos*" o micro-chips implantados en la piel.

Si se realiza la comparación con los móviles de hoy en día, su valor añadido radica en la oportunidad de monitoreo y escaneo del estado fisiológico del usuario, incluyendo la biorretroalimentación. Gracias a ello, abre nuevos horizontes a la mejora del bienestar del ser humano [1].

1.2 Marco Histórico.

Hoy en día, el dispositivo *wearable* se asocia con la idea de "inteligencia". No obstante, no deja de ser un término moderno. El objetivo primordial de estos, es mejorar la experiencia del usuario y con un mero ejemplo se puede entender lo que se trata de explicar: si se observan los relojes inteligentes, entre las funciones principales destacan recibir o rechazar llamadas o notificaciones de correos electrónicos, es decir, tener un móvil de menor tamaño. Sin embargo, es la evolución de los dispositivos lo que ha sido capaz de monitorizar patrones de sueño, el pulso cardíaco o el nivel de ejercicio. Por ello, como mera curiosidad, se propone un marco histórico de esta tecnología desde la primera concepción de estos hasta los más sofisticados [1].

Algunos autores [1,2,3] consideran el primer dispositivo *wearable* el llamado "Anillo de Ábaco", creado a principios del siglo XVII durante la dinastía Qing (véase Fig 1). Su función principal era la de ayudar al comerciante en la realización de cálculos. Estaba conformado por 10 cables paralelos englobados entre dos marcos con nueve cuentas en cada uno de ellos. Su tamaño era de 1.2 cm de largo y 0.7 cm de ancho.

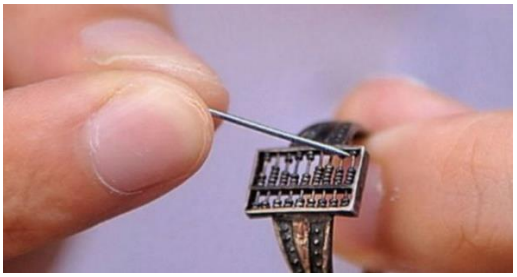


Figura 1: Anillo de Ábaco

Para los siguientes dispositivos, se tiene que dar un salto temporal importante hasta llegar al periodo entre la Primera Guerra Mundial y la Segunda. En estos tiempos, se inventó el primer dispositivo inalámbrico portable que permitía las comunicaciones, posteriormente llamado "walkie-talkie" en 1937 por Donald Hings (véase Fig 2. Izq) y patentado por Alfred J. Gross. En la cita [4], se puede leer el artículo hecho por "*Vancouver Sun*" en 1988 sobre Donald Hings.

Además de ello, en este periodo, se adaptaron relojes de pulsera, anteriormente empleados por las mujeres de la alta nobleza, que servían para la medición del tiempo exacto para los llamamientos de artillera y coordinación con la primera línea de batalla (véase Fig 2. Der).

Figura 2: Izquierda: Donald Hings en 1992 con el modelo C-58. Derecha: Reloj de Pulsera de Goldsmith de 1915 para soldados de trinchera

En 1960 de la mano de Morton Heilig, inventor y director de fotografía estadounidense, creó el primer dispositivo de realidad virtual llamado "*Sensorama*" (véase Fig 4.Izq). Utilizando como visor una pantalla estereoscópica en las que se reproducían imágenes tridimensionales, se permitía dar al usuario sensación de inmersión. Además, contaba con un sillón vibratorio, un generador de olores y un soplador de aire frío.

Al igual que sucedía con el "Anillo de Ábaco", multitud de autores consideran el siguiente invento como el primer dispositivo portable. En 1961 los investigadores del MIT *("Massachusetts Institute of Technology")* Edward O. Thorpe y Claude Shannon, el padre de la teoría de la información, concibieron un dispositivo de cronometraje dentro del zapato que podía predecir, con precisión, donde caería la bola en el juego de la ruleta (véase Fig 3). Si el lector me permite, me gustaría recomendar la lectura [5] en el que se explica como Edward O Thorpe se acerca a Claude Shannon e inventan este. Es cuanto menos curioso.

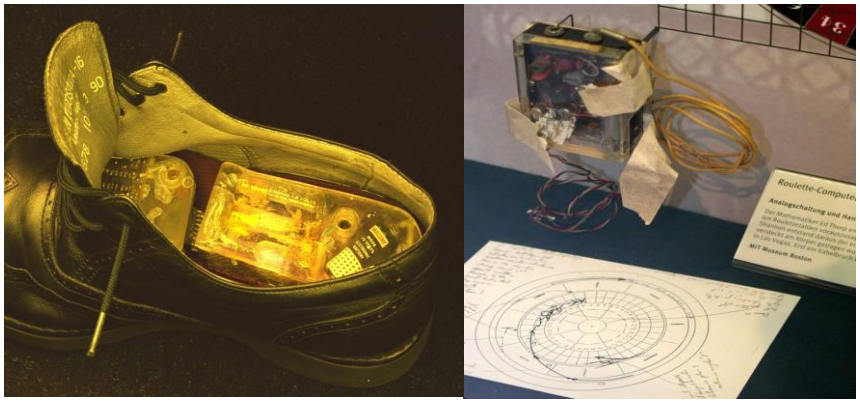


Figura 3: Izquierda: zapato con el receptor. Derecha: dispositivo transmisor

Rozando al término moderno de dispositivo *wearable*, se pueden encontrar invenciones tales la de Ivan Sutherland en 1968, padre de la computación gráfica, el cual crea el primer visor de realidad virtual y aumentada montado sobre la cabeza, la *"Espada de Damocles"* (véase Fig 4. Der). Aunque se quedase en un prototipo, cambia radicalmente la interacción hombre-ordenador.

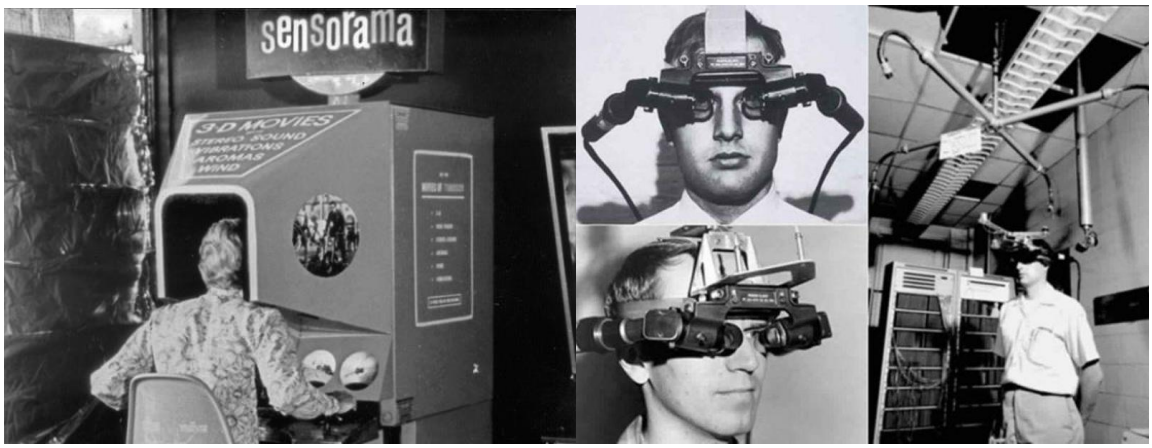


Figura 4: Izquierda: Sensorama. Derecha: la Espada de Damocles

En 1977, la compañía HP ("Hewlett Packard") crea el *"HP-01"* (véase Fig 5. Izq), siendo el primer reloj que hacía cálculos algebraicos.

Los finales del siglo XX se caracterizaron por inventos relacionados con la realidad virtual. De manera rápida, se pueden encontrar tales como el visor monocular llamado "*Private Eye*" vendido por *Reflection Technology* (véase Fig 5. Der) en 1989, considerado el padre de las "*Google Glass*", o el llamado "*Activa Badge*" como primer dispositivo que permitía la localización espacial mediante rayos infrarrojos dentro de infraestructuras de *"Olivetti Research"*. Resaltar, también, el "*mBracelet*", pulsera que permitía realizar transacciones financieras con el ATM ("*Automated Teller Machine*", cajero automático).

Cambiando de tercio, y casi próximo a la descripción de los tiempos modernos, a principios del siglo XXI, la división de ropa de "Levi" en colaboración con Philips crearon una chaqueta que permitía la interconexión entre aparatos electrónicos llamada *"Levi's ICD+ Jacket"* (véase Fig 5. Abj).

En 2003, sale al mercado el "*Fossil Wrist PDA*", creado por el ingeniero Donald Brewer y director de producto de "*Fossil*" Jeff Bruneau, el primer reloj inteligente, pudiendo realizar todas las funcionalidades que daba un PDA.

En 2006 Nike y Apple se aliaron para lanzar el *"Nike+iPod"*, el cual permitía medir y grabar la distancia recorrida, las calorías quemadas y velocidad y almacenarse en el *"iPod nano"*.

Finalmente, tras repasar por la superficie los inventos más importantes que han permitido la creación de los pilares de la tecnología presente, resaltar la iniciativa "*Android Wear*" en 2014 y la salida al mercado del primer "*Apple Watch*" en 2015.

Figura 5: Izquierda: HP-01, Derecha: "The Private Eye", Abajo: Levi's ICD+ Jacket

1.3 Estudio Económico.

A través de esta sección, el autor pretende resaltar el crecimiento continuo de la tecnología y el mostrar su aceptación, cada vez mayor, en la sociedad.

Según el estudio de mercado realizado por "*Transparency Market Research*" en 2021, el mercado de los dispositivos vestibles llegaría a tocar los 62.120,9 millones de dólares en 2025. Dicha estimación, se alinea con las predicciones de "*Cision PR*", la cual, mediante la medida CAGR ("*Compound Annual Growth Rate*"), estimó un crecimiento del 24.7% entre 2017 y 2025. Se estima que, para 2028 el mercado supere el techo de los 150 millones de euros [6].

Según [6], se puede observar un crecimiento continuo en el uso de estos dispositivos. Englobando a los dispositivos del mundo del deporte, se espera que los dispositivos conectados aumenten de 593 millones en 2018 a 1105 millones en 2025. En el caso de los pendientes, se elevó de 14.583 millones de USD a 32.724 millones desde 2019 a 2022. De menor crecimiento, los relojes inteligentes aumentaron su tasa de mercado desde los 18.501 millones de USD hasta los 21.758 USD de 2019 a 2020. Entre los dispositivos más aceptados, se encuentran las pulseras y los relojes teniendo una tasa de mercado del 51% en 2019.

Si se miran cifras por manufacturación, según el IDC ("*International Data Corporation*") y su estimación desde 2024 a 2028, se puede observar que los anillos y gafas inteligentes tuvieron un crecimiento del 88.4% y 73.1% respectivamente en 2024 y viéndose una disminución del 3.0% en los relojes inteligentes. Sin embargo, la joya de la corona se la lleva los pendientes, teniendo la mayor producción de estos en 2024 con un crecimiento del

10.2%. Si se observan los datos del 2028, se espera una estabilización de los productos, teniendo una tasa de crecimiento conjunta del 2.0% [7]. Para más información véase la figura 6.

Product Category	2024 Unit Shipments	2024 Growth	2028 Unit Shipments	2028 Growth	2024-2028 CAGR
Earwear	342.2	10.2%	399.0	2.5%	3.9%
Smartwatch	156.5	-3.0%	175.2	1.5%	2.9%
Wrist Band	35.2	7.8%	32.1	-2.0%	-2.3%
Rings	1.7	88.4%	3.1	5.8%	17.0%
Glasses	1.8	73.1%	2.3	6.3%	7.6%
Others	0.6	0.1%	0.7	1.1%	3.1%
Total	**537.9**	**6.1%**	**612.5**	**2.0%**	**3.3%**

Figura 6: Pronóstico Mundial de dispositivos vestibles por categoría de producto (envíos de unidades en millones)

Si se mira desde otra perspectiva, "*Mordor Intelligence*" estima que en 2024 el tamaño del mercado ronde los 84,23 millones de dólares en 2024 y para 2029 el mercado llegue a los 205,10 millones de dólares en 2029, presentando un CAGR de 19.48% [8].

Una comparación útil para entender lo grande que es dicho mercado, el "*Apple Watch*" vendió 31 millones de unidades en 2019, 10 millones más que las ventas realizadas por toda la industria relojera suiza [9].

En cuanto a las casas comerciales, destacan Apple teniendo el 18.2% de cuota de mercado, Xiaomi con 10.5%, Huawei con 9.6%, Samsung con el 9.3% e Imagine Marketing con el 5.3% (véase Fig 7) [10].

COMPAÑÍA	ENVÍOS Q1 2024	CUOTA DE MERCADO Q1 2024	ENVÍOS Q1 2023	CUOTA DE MERCADO Q1 2023	DIFERENCIA AÑO A AÑO
APPLE	20,6	18,2%	25,4	24,5%	-18,9%
XIAOMI	11,8	10,5%	8,2	7,9%	43,4%
HUAWEI	10,9	9,6%	6,3	6,1%	72,4%
SAMSUNG	10,6	9,3%	9,4	9,0%	13,0%
IMAGINE MARKETING	6,1	5,4%	6,4	6,2%	-4,8%
OTROS	53,1	46,9%	48,2	46,4%	10,1%
TOTAL	113,1	100,0%	104,0	100,0%	8,8%

Figura 7: Último informe de IDC Worldwide Quarterly Wearables Tracker. Datos de envíos por millon

Huelga decir que, dada la alta cantidad de dispositivos y de empresas consolidadas, la entrada al mercado de las nuevas se convierte en algo complicado. Por ello, además de tendencias económicas crecientes, se está observando una lucha empresarial por diferentes nichos, abogando por la hiperespecialización [11].

1.4 Clasificación y Ejemplificación.

La clasificación de los dispositivos *wearables*, puede hacerse de multitud de formas ya sea considerando la función que realizan, por el tipo de sensores que lo conforman o por el lugar de aplicación, por destacar algunas clasificaciones [1].

Sin embargo, por conveniencia y curiosidad de proponer ejemplos en el entorno médico, se sigue la clasificación por lugar de aplicación.

1.4.1 Dispositivos Montados en la Cabeza.

Principalmente, se centran en funcionalidades como la percepción y el control. Se engloban a todas las gafas AR/VR/XR/MR, dispositivos HMD e instrumentos relacionados con el audio. Los HMD presentan como fortaleza principal mostrar información con manos libres sin que el campo de visión del usuario se vea afectado [1].

Algunos ejemplos de instrumentos serían:

Las "***Google Glass***" tienen una pantalla óptica montada en la cabeza, un transductor de conducción ósea en la zona temporal, una CPU para ejecutar el sistema operativo Glass, un micrófono para el reconocimiento por voz y una grabadora de video. Además, cuenta con la capacidad de conexión vía Wi-Fi y Bluetooth. La cámara tiene la capacidad de tomar fotos de 5 megapíxeles y realizar videos de 720 píxeles de calidad [12,13]. Un ejemplo de aplicación sería:

- Uso para el **entrenamiento de la técnica quirúrgica en residentes de Neurocirugía**. Se enseñaban a los residentes tres videos: el primero en relación con la discectomía lumbar invasiva, el segundo mostrando el registro intraoperatorio de una craneotomía de emergencia para la evacuación de líquido epidural y un tercero grabando el estado postoperatorio de un paciente de Mongolia. En estos tres, sin la necesidad de estar presente el alumno, eliminando la posibilidad de interrumpir el flujo de trabajo, se mostraba como se preparaban las máquinas de imagen, el procedimiento o el posicionamiento de las herramientas [12].

En el trabajo [14], se presenta el empleo de las ***"Moverio BT-35E"*** para una **cirugía ortopédica**. En estos casos, dada la complejidad que puede llegar a tener la técnica quirúrgica, el cirujano ha de adoptar posiciones no ergonómicas para poder trabajar en el campo de operación, aumentando la posibilidad de desarrollar lesiones musculoesqueléticas, sin tener en cuenta los movimientos que ha de hacer para mirar las televisiones con las imágenes. Por ello, se utilizan las "*Moverio BT-35E"* para poder visualizar a tiempo real las imágenes por fluoroscopia y permitir al cirujano una posición más cómoda. Dichas gafas tienen la capacidad de tener conexiones HDMI y USB-C, presenta una pantalla Si-OLED de 1280x720 px (HD) y sensores de movimiento IMU *("Inertial Measurement Unit")* (véase Fig 8).

Figura 8: Izquierda: Entorno Quirúrgico, cirujano con las Moverio y pantallas de fluoroscopia detrás. Derecha: Visión del cirujano, gafas, receptor inalámbrico y batería

Como último ejemplo de aplicación, se propone el expuesto en [15]. Mediante el uso de las ***"HoloLens",*** se hace la **superposición de imágenes en formato DICOM en el paciente a partir de hologramas**. (véase Fig 9). De esta forma, se consigue un sistema de navegación en procedimientos neurológicos. Dichas gafas fueron concebidas para la combinación de AR y VR y por ello, presentan un sistema autónomo, pantalla holográfica, usando tecnología de ondas para proyectar imágenes, seguimiento de movimientos y gestos mediante cámaras de luz visible e infrarrojo, un sensor ToF ("*Time of Flight*"), una IMU y capacidad de tomar fotos de 8Mp y videos de calidad 1080p30 [16].

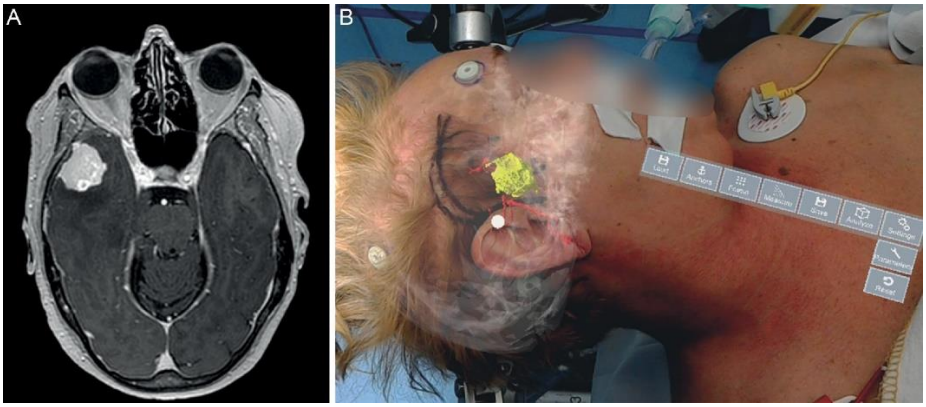


Figura 9: A: Imagen de Resonancia Magnética. B: Holograma mostrando la vasculatura y el tumor

1.4.2 Dispositivos colocados en el Cuerpo.

Están considerados como la nueva generación de dispositivos médicos de cuidado portables. Lo que los caracteriza es la capacidad que tienen de adaptarse a la piel y la flexibilidad que ofrecen permitiendo medidas continuas sin comprometer el movimiento natural y la comodidad del usuario [17]. Algunos ejemplos de aplicación serían:

- La **creación de un sensor en forma de parche** que permite la **medición de las formas de ondas del pulso cardíaco** [17,18]. Para ello, emplearon fotodetectores inorgánicos de 2x2 y espesor de 4 micras y un LED rojo, encapsulados en un elastómero (véase Fig 10). El sensor de pulso genera la luz

del LED y los fotodetectores reciben la luz reflejada en huesos y tejidos. La intensidad de luz reflectada de la sangre arterial va alternando por su carácter pulsátil mientras que la intensidad venosa y del resto de tejidos se mantienen constante. Gracias a las fluctuaciones, se representa el cambio de volumen de sangre arterial movida por el corazón, permitiéndose la medición de esta en varias partes del cuerpo tales dedos, uñas o antebrazos.

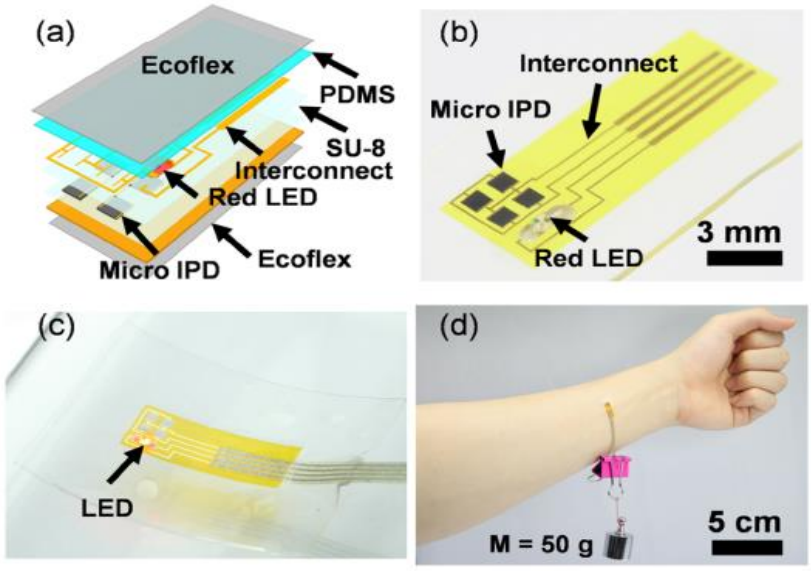


Figura 10: A. Visión Esquemática Aumentada del sensor. B. Visión Óptica. C. Sensor en una superficie curva. D. Sensor puesto en el antebrazo soportando una masa de 50 gramos.

- El **prototipado de parche que monitoriza el sudor** [17,19]. Se ha demostrado que el sudor es un fluido fácilmente accesible, el cual puede proporcionar información diagnóstica importante mediante su composición, ya que presenta proteínas y metabolitos asociados a enfermedades e infecciones. Por ejemplo: el sensado del lactato para saber que caminos energéticos se están llevando a cabo en la actividad física o el aumento de sustancias por causa del aumento del metabolismo anaeróbico [19]. En el estudio, se creó un parche (véase Fig 11) compuesto de un canal microfluídico de papel, con sensores flexibles de microagujas integrados y conectados a un potenciostato inalámbrico, el cual hace la transmisión mediante Bluetooth Smart basado en el circuito integrado Nrf51822 de "*Nordic Semiconductor*". En este, se dedicaron a probar la eficacia del sensor en la medición del pH y del lactato durante diferentes situaciones de deporte.

- **Diseño de una muñequera para monitorizar el movimiento y aplicar termoterapia en lesiones articulares** [20]. Para cumplir su cometido, a través de un proceso de trenzado, se fabricaron los hilos sensores de núcleo y cubierta (CSSYs) empleando nylon recubierto de negro de carbono, nylon plateado con plata y spandex elástico. Al torsionar dos CSSYs, se forma la estructura de doble hélice y obteniéndose hilos CSSY de doble capa optimizados, cuya función es la de actuar como sensores capacitivos de deformación (véase Fig 12). Con su posterior costura, se obtiene la prenda que aplica la termoterapia de manera eficiente y simultáneamente es capaz de seguir la lesión y recuperación.

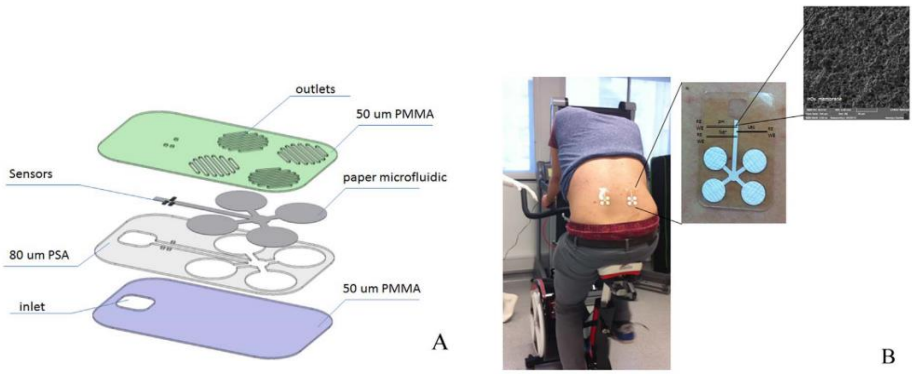


Figura 11: A. Representación Esquemática del chip. B. Sensor colocado en la espalda del deportista

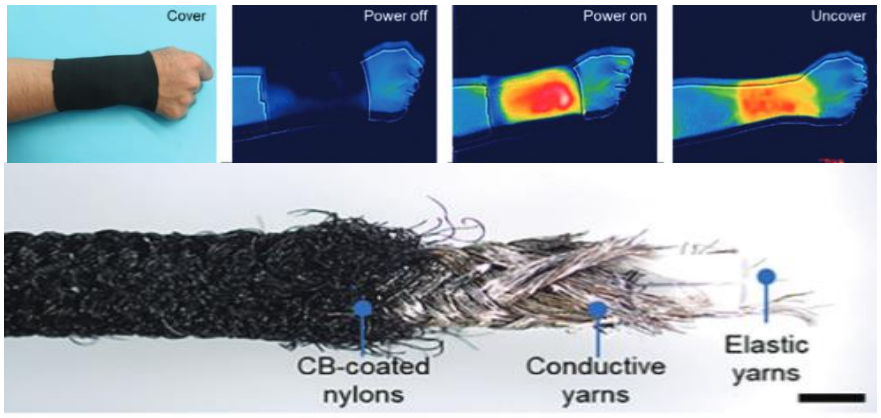


Figura 12: Superior: Rendimiento de calentamiento eléctrico del CSSYs. Inferior: Imagen microscópica del CSSYs

1.4.3 Dispositivos Colocados en la Muñeca.

Los dispositivos colocados en la muñeca, como pulseras y relojes inteligentes, se han convertido en herramientas clave para el monitoreo de diversas señales fisiológicas. Gracias a su diseño compacto y facilidad de uso, estos dispositivos permiten la medición continua de parámetros como frecuencia cardíaca, presión arterial, niveles de actividad física y sueño, entre otros. Además, con la incorporación de tecnologías como sensores ópticos, acelerómetros y giroscopios, pueden ofrecer un seguimiento en tiempo real, brindando a los usuarios una visión detallada de su salud y bienestar [17]. Algunos ejemplos serían:

- El ***dispositivo "GlucoWatch G2"*** (véase Fig 13. Izq) de la empresa "*Cygnus*" [20]. La medición de los niveles de glucosa se obtiene mediante iontoforesis inversa basada en la extracción de líquido intersticial a través de la piel. Para conseguir dicha funcionalidad, el reloj aplica una corriente eléctrica de 300 microamperios entre dos electrodos que contactaban la piel en la parte posterior del dispositivo.

- El ***empleo de un reloj inteligente para el análisis del temblor en estado de reposo de pacientes con Parkinson*** [21]. Se empleo el **"*Smartwatch 3*"** de la casa comercial de "*Sony*" (véase Fig 13. Der). Este, como características técnicas principales presenta una resolución de pantalla de 320x320px con conectividad Bluetooth 4.0 y Wi-Fi y sensores de luz ambiental, acelerómetro,

brújula y giroscopio. A través de los datos del giroscopio, analizaron las frecuencias desde 4 a 6 Hz por las que se caracteriza el temblor en reposo.

Figura 13: Izquierda: GlucoWatch G2. Derecha: Smartwacth 3

2. Anatomía de la rodilla.

Siendo la articulación más grande y compleja, la rodilla es una articulación de bisagra bicondílea sinovial entre los cóndilos femorales y tibiales y la rótula. Usualmente, estabilidad y movilidad son funciones incompatibles, se sacrifica una por otra. Sin embargo, gracias a la interacción de ligamentos y músculos y los movimientos de deslizamiento y balanceo, la rodilla cumple con ambas. Dada la pequeña estabilidad ósea, es la articulación que más lesiones sufre y por lo tanto, músculos y ligamentos sufren constantemente de tensiones y presiones elevadas [22].

Esta puede describirse a través de dos articulaciones bien diferenciadas: la tibiofemoral y la patelofemoral, efectuando tres funciones fundamentales [23]:

1. Transmisión de las fuerzas del fémur a la tibia.
2. Absorber y redistribuir el estrés generado durante los movimientos.
3. Permitir la marcha con gasto energético mínimo.

Gracias a las dos articulaciones anteriormente mencionadas, el movimiento de la rodilla puede ocurrir, fundamentalmente, en dos planos: el de la flexión y extensión y el de la rotación interna y externa. No obstante, el movimiento no se produce de manera independiente tal y como se refleja, por ejemplo, durante el balanceo en la marcha en el que la rodilla se flexiona para disminuir el recorrido de la extremidad inferior y generando una pequeña rotación. Otro ejemplo de la importancia de la interacción correcta es en la bipedestación. La rodilla se encuentra ligeramente flexionada para permitir las tres funciones mencionadas. El porqué de ello, radica en la fuerte asociación entre cadera, rodilla y tobillo y si se observase globalmente la anatomía del miembro inferior, dos tercios de los músculos que atraviesan la rodilla, también lo hacen en la cadera y tobillo [24].

En los siguientes apartados se hablará sobre ambas articulaciones, la importancia del control del rango de movimiento y su síntesis en las patologías comunes de la rodilla y los efectos psicológicos del daño en el paciente durante el proceso de rehabilitación.

2.1 Articulación Tibiofemoral.

La articulación tibiofemoral consiste en la combinación entre los cóndilos femorales convexos y los cóndilos tibiales, casi planos y de menor tamaño. Gracias a los primeros, se permiten movimientos en el plano sagital. La estabilidad de la articulación se consigue mediante las fuerzas y el contacto físico de músculos, ligamentos, la cápsula, meniscos y el peso corporal (véase Fig 14)[24].

Desde el punto de vista funcional y biomecánico, la articulación tibiofemoral se describe mediante seis grados de libertad: tres movimientos de rotación y tres movimientos de traslación. Importante a considerar, son los rangos de movimiento que, de media, puede realizar en un sujeto sano:

-Flexión-Extensión: -15 a 140 grados.

-Rotación en varo-valgo: -6 a 8 grados.

-Rotación interna-externa: 25 a 30 grados.

-Traslación antero-posterior: 10 a 15 mm.

-Traslación medio-lateral: 2-4 mm.

-Traslación superior-inferior: 2-5 mm.

Las disfunciones de rodilla causadas por lesiones o degeneraciones son una condición física prevalente que afecta a la vida diaria del paciente. Del mismo modo, debilidades o reconstrucciones quirúrgicas pueden desembocar en descompensaciones de la mecánica de la articulación, aumentando la prevalencia de diferentes patologías. A causa de ello, es necesaria la implantación de programas de rehabilitación robustos [24].

2.2 Menisco

El menisco lateral y el medial son estructuras fibrocartilaginosas semejantes a una media luna (véase Fig 15). La existencia de estos permite la transformación de la superficie articular de la tibia en cojinetes para los cóndilos convexos femorales, siendo la parte más importante la zona lateral de la tibia debido a la transición de plana a ligeramente convexa [3,4].

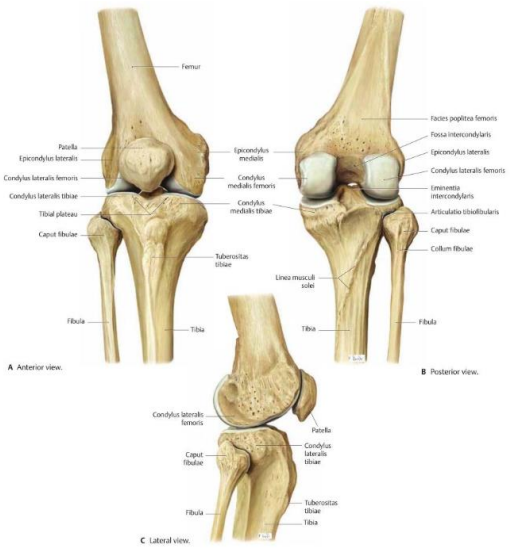


Figura 14: Visualización del componente óseo de la articulación tibiofemoral. A. Vista Anterior, B. Vista Posterior, C. Vista Lateral.

Los picos externos de cada menisco se encuentran unidos a la tibia y a la cápsula articular mediante los ligamentos coronarios. Estos últimos, están ligeramente sueltos para permitir rotaciones libres durante el movimiento. Además, la unión entre meniscos se consigue mediante el ligamento transversal.

De la misma importancia es la cantidad de uniones secundarias de diversos músculos alrededor de esta zona, como puede ser el cuádriceps, proporcionando mayor estabilidad al menisco.

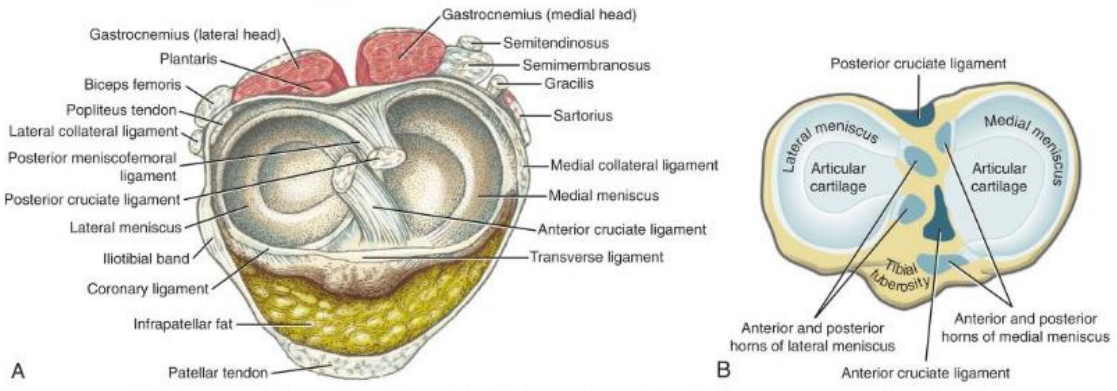


Figura 15:Visión Superior de: A. La superficie de la tibia, meniscos y otras estructuras. B. Superficie de la tibia con los puntos de intersección de los meniscos y los ligamentos cruzados.

2.2.1 Funciones.

Su función primaria es la reducción del estrés compresivo a lo largo de la articulación tibiofemoral mediante el aumento de la superficie. Dichas fuerzas, pueden superar entre 2.5 y 3 veces el peso corporal de la persona durante la marcha [25]. Pacientes los cuales hayan sufrido una menisectomía (extracción quirúrgica de la totalidad o una parte del

menisco), tienen mayor probabilidad de desarrollar enfermedades degenerativas articulares debido al aumento del desgaste [25].

Asimismo, se pueden observar funciones tales como: estabilización articular durante el movimiento, lubricación del cartílago articular, la propiocepción sobre la posición, velocidad y aceleración de la rodilla y guiar la artrocinemática de la rodilla [24,25].

2.3 Ligamentos colaterales mediales y laterales.

El ligamento colateral medial es una estructura plana que atraviesa la zona medial de toda la articulación, presentando sus inserciones en los cóndilos mediales femorales y tibiales (véase Fig 16). De la misma forma, se pueden observar algunas fibras insertándose en el menisco medial y provocando que, si hay un exceso de tensión en el ligamento, probablemente, este se desgarre.

Atendiendo al autor, este se puede dividir en una zona superficial consistente de fibras paralelas planas e insertándose debajo de la tuberosidad tibial y otra profunda de fibras más pequeñas y oblicuas unidas al menisco.

En cuanto al ligamento colateral lateral, consiste en fibras redondas que atraviesan, casi, verticalmente el epicóndilo del fémur y la cabeza de la fíbula y se inserta en el cóndilo lateral del fémur y se dirige hacia abajo hasta el peroné (véase Fig 5).

2.3.1 Funciones

Ambos, en común, limitan el movimiento excesivo en el plano frontal y el movimiento lateral. Durante la flexión, ambos ligamentos se encuentran parcialmente relajados y durante la extensión se van tensando, llegando al límite en la posición de bloqueo. El medial otorga estabilidad medial y previene de lesiones frente golpes laterales.

2.4 Ligamentos Cruzados Anteriores y Posteriores.

El concepto cruzado se da por la relación espacial entre ambos ligamentos y la apertura del fémur a medida que lo cruzan (véase Fig 16, 17 y 18). Estos, son intracapsulares y se hallan recubiertos del líquido sinovial.

De manera común, ambos son estrechos y fuertes permitiendo la estabilidad multiplanar y son los elementos más resistivos frente a movimientos anteroposteriores y las fuerzas de corte que se generan por la tibia y fémur. Presentan mecanorreceptores que

proporcionan retroalimentación, limitando la activación del musculo para evitar daños por la tensión generada en el ligamento anterior cruzado y evitar la hiperextensión

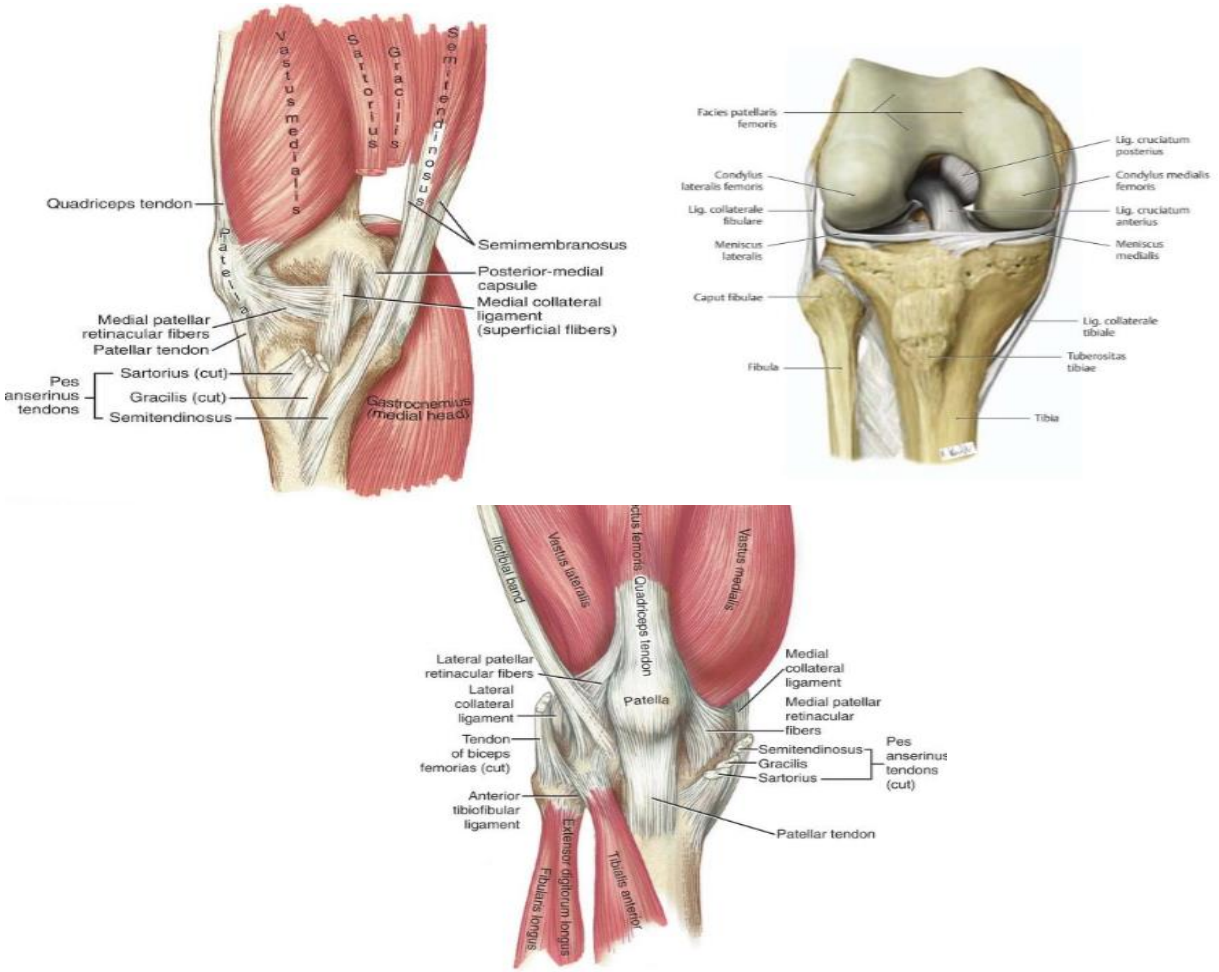


Figura 16: Izquierda: Visión Medial de la rodilla mostrando músculos y tejidos conectivos. Derecha: Visión Anterior sin la cápsula articular ni la rótula. Inferior: Visión Anterior

2.4.1 Ligamento Anterior Cruzado.

La tensión y orientación de este cambia a medida que la rodilla se flexiona y extiende e impedir la hiperextensión (véase Fig 17). Aunque haya fibras que permanezcan tensas en todo el rango del movimiento en el plano sagital, la gran mayoría aumentan la tensión, a medida que la rodilla alcanza el grado máximo de extensión. Durante los últimos 50-60 grados de extensión, la fuerza generada por el cuádriceps permite el movimiento anterior de la tibia y con ello, accionando el ligamento. Dicha tensión, permite evitar el desplazamiento posterior del fémur sobre la tibia o la tibia anteriormente sobre el fémur

Algo curioso es que se puede definir al cuádriceps como antagonista, ya que en ángulos pequeños de flexión, el músculo estrecha a las fibras de esta por la fuerza de contracción producida.

2.4.2 Ligamento Posterior Cruzado.

De menor grosor que el anterior, generalmente se puede definir mediante dos paquetes de fibras principales: uno anterior-lateral, formando la mayor parte del ligamento, y otro posteromedial (véase Fig 18).

A medida que la rodilla se flexiona, el ligamento se tuerce y cambia tanto su orientación como longitud. A diferencia de su homónimo, evita el desplazamiento anterior del fémur sobre la tibia o la tibia posteriormente sobre el fémur.

Sus funciones no son totalmente conocidas, pues la incidencia en las lesiones es bastante pequeña. Sin embargo, se sabe que las fibras se mantienen totalmente tensas durante la flexión y extensión, siendo mayor conforme aumenta la flexión.

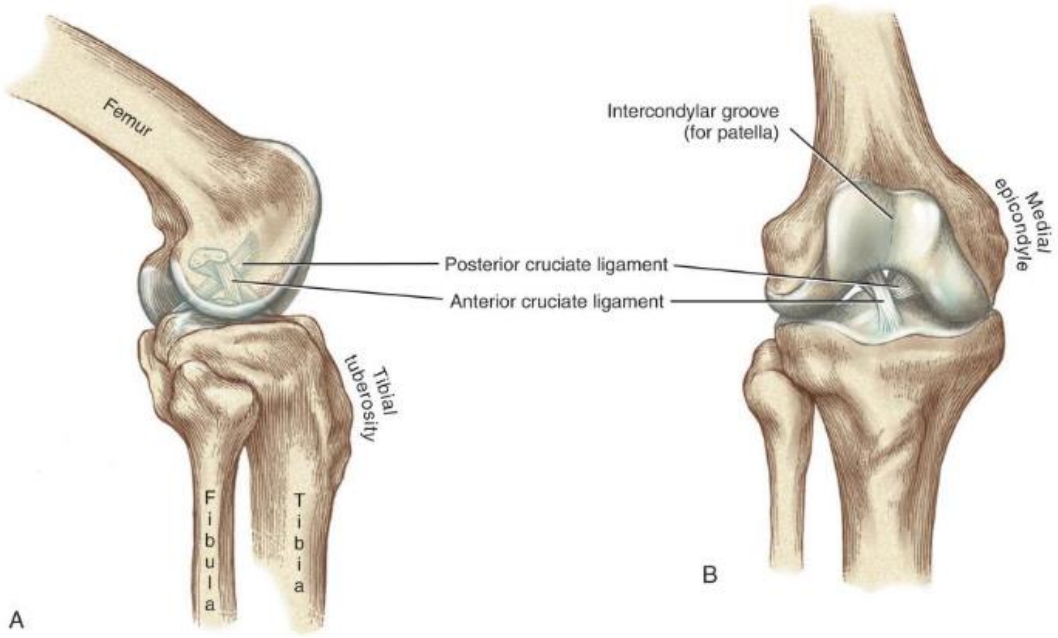


Figura 17: Visión del Ligamento Cruzado Posterior y Anterior. A. Visión Lateral. B. Visión Anterior.

2.5 Articulación Patelofemoral.

La articulación patelofemoral está formada por la parte articulada de la rótula (patella), abrazada por el tendón rotuliano y recubierta de cartílago articular para disminuir la fricción, y la tróclea femoral (véase Fig 16 y 18). Como estabilizadores locales, se tiene la fuerza producida por el cuádriceps, las superficies articulares y la sujeción pasiva de las fibras circundantes y de la cápsula. Por norma general, a medida que la rodilla se flexiona y se extiende, se genera un movimiento de deslizamiento de la articulación.

Durante la rotación tibia-femoral, la rótula se desliza respecto de la tróclea fija y a causa de sus uniones óseas a la protuberancia tibial, en movimiento de extensión, esta sigue a la tibia. Si se presentan rotaciones femoral-tibial, la tróclea gira con respecto a la rótula.

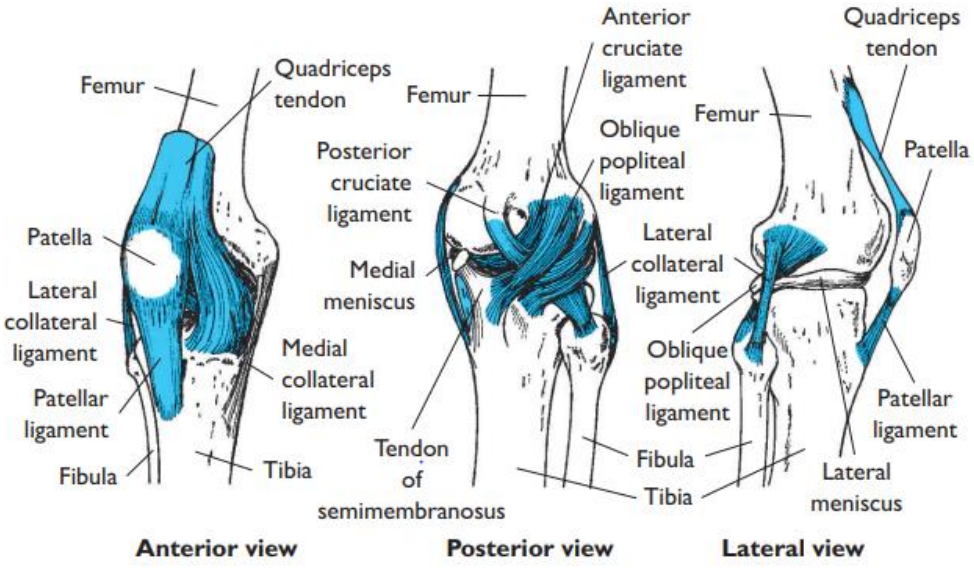


Figura 18: Visualización de todos los ligamentos de la rodilla.

2.5.1 Funciones.

La rótula permite incrementar el ángulo de tracción del tendón del cuádriceps, mejorando la mecánica de dicho músculo durante la extensión. Además, presenta la acción de centralizar la tensión divergente de este, transmitida al tendón rotuliano, y disminuir las tensiones de rozamiento de la articulación, ya que aumenta la superficie de contacto entre el fémur y el tendón rotuliano. Finalmente, como últimas funciones primordiales, otorga protección anterior a la rodilla y protege al cuádriceps frente a fricciones con los huesos adyacentes.

Cuando se presentan 135 grados en flexión, la rótula contacta con la tróclea en su parte superior y a medida que se acerca a la posición máxima, esta descansa sobre la fisura.

2.6 Importancia en la Limitación del Movimiento.

Los ejercicios de rehabilitación se dividen en dos categorías: *"Open Kinetic Chain"* (OKC) y *"Closed Kinetic Chain"* (CKC). Se definen como cadenas cinéticas puesto que todo el segmento de la extremidad inferior se puede considerar una vara rígida. La definición de abierta o cerrada se da atendiendo al segmento distal. Por ello, los ejercicios OKC se dan cuando dicho segmento tiene movimiento libre en el espacio, sin soporte de peso, y en los CKC se encuentra limitada de alguna forma [28,29,30].

2.6.1 Ejercicios OKC.

Esta categoría de ejercicios permite el libre movimiento de la articulación distal sin soporte de peso. Algunos ejemplos de estos serían: extensiones de pierna sentado, extensión terminal de rodilla, curl de femoral y elevación de gemelos (véase Fig 19).

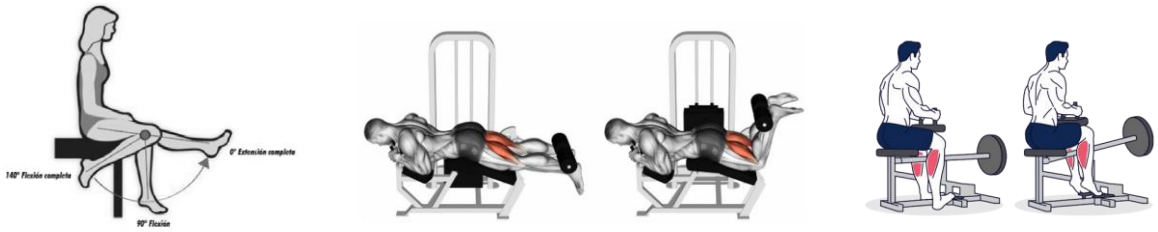


Figura 19: Izquierda: Extensión de Rodilla. Medio: Curl de Femoral. Derecha: Elevación de gemelos

Entre los beneficios encontrados de esta modalidad destacan: mayor actividad muscular aislada, aumento de la fuerza y del rango de movimiento promoviendo patrones de marcha normales [28].

Se ha demostrado que en 90º de flexión, las fuerzas del cuádriceps son la mitad que las observadas a 0 grados cuando la fuerza de la gravedad actuaba como única perturbación durante extensiones de rodilla sentado. A la vez, se vio que las fuerzas de dicho músculo se duplicaron cuando la rodilla se flexionó con una carga resistiva. Por ello, se concluyó que, junto con otros estudios, la fuerza del cuádriceps pierde su habilidad de generar momentos de extensión a medida que el ángulo de la articulación aumenta. [28,29]

Siguiendo con la misma dinámica, se observó que las fuerzas generadas por el ligamento cruzado anterior se reducen considerablemente cuando la articulación se encuentra en 30º de flexión y es, aproximadamente, cero en 60º y por ello, ejerciendo la fuerza máxima cuando la rodilla se encuentra totalmente en extensión. No obstante, las fuerzas exhibidas por el ligamento posterior cruzado son a la inversa, teniendo la mayor tensión cuando la rodilla se encuentra flexionada 90º.

Por consiguiente, se recomienda que para pacientes que padezcan reconstrucciones o lesiones del ligamento anterior cruzado, no se realicen ejercicios con todo el rango de movimiento. Si se desea realizar rehabilitación en el que haya extensiones de rodilla, se deben tener rangos de flexión entre 0 y 60 grados para la minimización del posible estrés generado en el ligamento posterior cruzado.[29]

2.6.2 Ejercicios CKC.

A diferencia que los anteriores, la articulación distal se encuentra fija y por ende, limitando el posible movimiento generado por esta. Los ejercicios que destacan de esta modalidad son: las sentadillas, las zancadas o la prensa de pierna (véase Fig 20). En este caso, se promueve la participación conjunta de todos los músculos participantes en el movimiento, permitiendo mayor estabilización y control de la articulación [28].

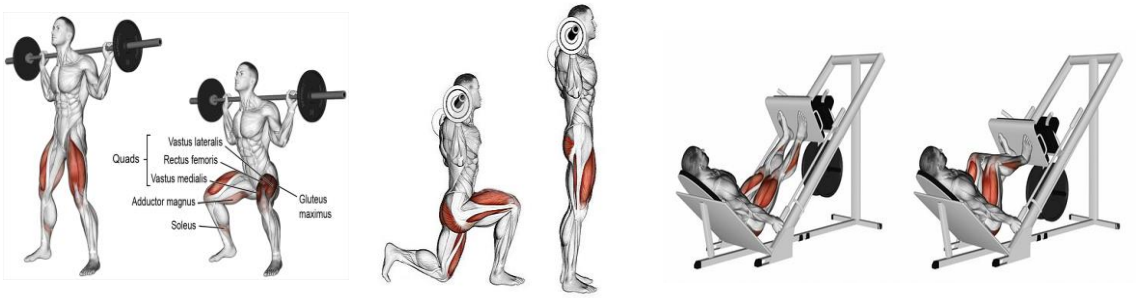


Figura 20: Izquierda: Sentadilla; Medio: Zancada; Derecha: Prensa de Pierna

Dichos ejercicios, suelen presentar contracciones simultáneas del cuádriceps y de los isquiotibiales. Gracias a ello, suelen ser aconsejables para la estabilización de la articulación y protección del ligamento anterior cruzado [8].

Se demostró que en ejercicios de sentadilla con múltiples ángulos de flexión, las fuerzas generadas por el cuádriceps incrementaban significativamente con flexiones entre 20º y 90º, siendo el pico en el segundo. Adicionalmente, las fuerzas generadas por el ligamento anterior cruzado presentan un pico en 60º y posteriormente, se vieron reducidas al llegar a 90º.

Por ello, se desestima la realización de ejercicios de sentadillas con ángulos y momentos articulares mayores y con pesas en las manos por el posible aumento en la compresión de los ligamentos [29].

2.6.3 Rehabilitación de la Articulación Patelofemoral.

En este caso, diferentes ejercicios pueden derivar a diferentes fuerzas reactivas de la articulación patelofemoral.

En general, para ejercicios de extensión OKC, la fuerza ejercida por el cuádriceps y el estrés generado son máximas cuando la rodilla se encuentra totalmente extendida. Sin embargo, para ejercicios CKC es al contrario, hallándose en flexión máxima. Cuando la rodilla se encuentra flexionada en 90º en OKC, las tensiones en la rótula son las mínimas y en el momento en el que se sobrepase, las fuerzas aumentan, considerablemente, por la pérdida del área de contacto entre rótula y tróclea. Inversamente, en ejercicios CKC, en 20º, la tróclea pierde el área de contacto.

Los pacientes que muestran incomodidad en la articulación pueden beneficiarse tanto de ejercicios CKC como OKC, siempre y cuando se hagan en un rango de movimiento libre de dolor.

En consecuencia, se recomienda que los ángulos de flexión en OKC se encuentren en el rango 90º-50º y 20º-0º. Para CKC, se recomiendan flexiones en el rango 0º-45º, principalmente, porque si se hacen de manera correcta, la carga es soportada, mayoritariamente, por el cuádriceps. [29,30].

2.6.4 Aplicación de los ejercicios para la osteoartritis de rodilla.

La osteoartritis (OA) es una condición degenerativa de las articulaciones sinoviales. La OA ha sido definida como un síndrome clínico, el cual se caracteriza por incomodidad articular, disminuyendo el bienestar del paciente.

En el caso de la articulación de la rodilla, esta se caracteriza por dolor al soportar peso, sensibilidad, restricción en la movilidad, crepitación, derrames intermitentes y diferentes grados de inflamación. Se han encontrado evidencias que señalan el fortalecimiento de la musculatura de alrededor y al ejercicio aeróbico como atenuadores del efecto de la enfermedad.

En este caso, tanto ejercicios OKC como CKC son igualmente efectivos para el fortalecimiento del cuádriceps. Sin embargo, históricamente, se han preferido ejercicios OKC por la sospecha de que los CKC pueden acelerar el desgaste articular [29].

2.6.5 Aplicación de ejercicios después de la reconstrucción del ligamento cruzado anterior.

Las lesiones de ligamento cruzado anterior suelen provocarse a causa de la rotación del fémur con el pie en el suelo y la rodilla cerca de la extensión máxima. Deportes tales como el beisbol o el baloncesto, pueden generar momentos rotacionales y fuerzas en varo o valgo lo suficientemente grandes para romper el ligamento. Tras una reconstrucción quirúrgica, los pacientes suelen experimentar debilidad, pérdida de masa de los extensores, reducción del rango de movimiento y propiocepción disminuida [23,26].

La realización de ejercicios CKC permite poner menos carga en el ligamento al igual que la disminución en la incomodidad en el movimiento y la laxitud de la rodilla. La razón de ello, es que la contracción del cuádriceps, isquiotibiales y músculos gastrocnemios reducen las tensiones de corte entre la tibia y el peroné, aumentando la compresión articular y desembocando en mayor estabilidad articular y protección del injerto [28,29].

2.6.6 Aplicación de ejercicios con síndrome del dolor patelo-femoral.

Es uno de los trastornos musculoesqueléticos más comunes. Se caracteriza por dolor anterior durante y después de la actividad física, sobre todo en aquellas en las que la flexión de rodilla es fundamental. Algo a considerar es que, el vasto medio oblicuo (VMO) y el vasto lateral (VL) son los dos músculos principales que dan apoyo a la rótula durante la extensión dinámica y se sospecha que dicho síndrome surge por un cambio en la ratio de contracción, especialmente la debilidad del VMO, generando el movimiento patológico de la rótula, inflamación, incomodidad y deterioro rápido del cartílago [23,26].

Por ello, los ejercicios han de enfocarse al fortalecimiento del VMO, siendo los más empleados los OKC, dado que el fortalecimiento del cuádriceps es mayor entrenando a 60º de flexión que en 30º o 90º. De la misma forma, durante extensiones de rodilla el VMO es más activo en el rango de movimiento entre 60º y 90º, siendo máximo en el primero [29].

Por ello, ejercicio CKC, con ángulos de flexión entre 15 y 90 grados, permite un entrenamiento más exhaustivo del vasto medio oblicuo [29].

2.6.7 Efectos Psicológicos en la rehabilitación.

Los efectos psicológicos, ya sean respuestas emocionales o cognitivas, juegan un papel importante tanto en los resultados como en el comportamiento del paciente durante la rehabilitación. La hipótesis principal es que el estrés y la depresión afectan globalmente tanto a la salud mental como a la física y el apoyo social que reciben provocan una fuerte modulación entre estos [31]. De la misma forma, en la literatura de la rehabilitación de la extremidad inferior, se pueden encontrar trabajos que relacionan la severidad de los síntomas y la capacidad de recuperación de la cirugía con dichos efectos [32]. Además de ello, se sospecha que una de las posibles causas del dolor persistente no explicado y pobre recuperación de las funcionalidades preoperatorias es la angustia psicológica [33].

Se han hallado evidencias de que el miedo al dolor, es la respuesta más común entre los pacientes que dejan el proceso de rehabilitación. Pacientes que experimentan dolor recurrente con ciertos movimientos provoca, usualmente, lo llamado “kinesiofobia” o evitación activa del movimiento provocada por el pánico al dolor recurrente o recaída a una posible nueva lesión [31].

Dentro de toda la maraña de creencias del paciente, las más destacables son: la autoeficacia, miedo al movimiento, con asociación estadísticamente significativa entre la intensidad del dolor y miedo al movimiento con el nivel de funcionamiento motor en las actividades diarias, y la catastrofización del dolor, acentuado por la debilidad muscular y la disminución en la tolerancia en el ejercicio desembocando en una sensación de abandono por la falta de control que tiene el paciente a la hora de realizar vida diaria [31,33].

Huelga decir que una parte importante del proceso es la concepción de la enfermedad. Tal y como se demuestra en [34], un factor importante en la recuperación temprana y seguimiento dentro del proceso de la rehabilitación es el conocimiento de la patología y del proceso quirúrgico. Pacientes que creían que la patología no iba a afectar apenas a sus vidas, obtienen, comúnmente, mejores resultados funcionales. Con ello, únicamente, se pretende destacar la necesidad de educar al paciente para obtener mejores resultados.

Se ha intentado fundamentar porqué el daño y sus derivados son agravantes de la patología. Sin embargo, de la misma forma, se puede hacer con situaciones prequirúrgicas. Se ha demostrado que la depresión y ansiedad preoperatoria desembocan en sensaciones de gran dolor e incluso presentado en cuadros clínicos sin causa aparente [32].

3. Descripción del Hardware.

Bajo el pretexto mencionado en los apartados anteriores, se pretende crear un prototipo de goniómetro de bajo costo el cual presente las siguientes funcionalidades:

- **Visualización de los ángulos en los movimientos de rodilla** del paciente y por ende, **conocimiento continuo del rango de movimiento**. Además de ello, si se

generase dolor, gracias a la visualización constante a través de la interfaz, podría ser limitado.

- Mediante la **selección del ángulo máximo** por parte de los profesionales y **vibradores como controladores**, permitir que el usuario sea consciente de sus límites. Al igual que, gracias a ello, generar una pequeña independencia del paciente en ejercicios en los que el rehabilitador no participa activamente.

Para ello, el prototipo estará compuesto por: un Arduino nano como controlador, un vibrador, una IMU de 6 ángulos de libertad, un acelerómetro y un módulo bluetooth para conseguir la interacción Arduino-Interfaz.

3.1 Arduino Nano.

Se escoge dicho controlador por la dimensión que presenta,45 x 18 mm, siendo el de menor tamaño entre los disponibles (véase Fig 21). Se desea que el prototipo sea lo más pequeño posible, adaptándose a la anatomía de la rodilla y no generar molestias al usuario. Además, destacar el bajo coste de compra y la alta disponibilidad de este en el mercado tanto de modelos originales como réplicas.

Otra de las razones por las que se selecciona un controlador de la familia Arduino, es por la cantidad de soporte, información y bibliotecas publicadas por la comunidad, desembocando en una implementación e integración de los diferentes componentes lo más sencillo posible.

Entre las características a mirar, destacan [35]:

- **Rango de voltaje de entrada entre 7-12V**. Aunque su voltaje de operación sea de 5 V, conseguido por la presencia de un regulador de voltaje interno, si se quisiera incorporar una batería para alimentar a todos los componentes, sería posible. Sin embargo, se descarta la posibilidad de alcanzar el límite superior para evitar posibles sobrecalentamientos y daños físicos.
- **Presencia de pines PWM** (“*Pulse Width Modulation*”), 3,5,6,9,10,11. El vibrador seleccionado es modulado por PWM. Por ello, se podría variar su frecuencia de vibración y por consiguiente, la fuerza con la que cumple su función; se puede modificar a través de dichos pines con una simple línea de código.
- Presenta pines para **conexión I2C** (“*Inter-Integrated Circuit*”), A4 (SDA , “*Serial Data*”) y A5 (SCL, “*Serial Clock*”) y **SPI** ,(“*Serial Peripheral Interface*”), 10 (SS , *“Slave Select”),* 11(MOSI, “*Master Out, Slave In*”), 12(MISO *, “Master In, Slave Out”*) 13 (SCL, *“Serial Clock”*). Aumentando la flexibilidad de configuraciones y empleo de sensores.

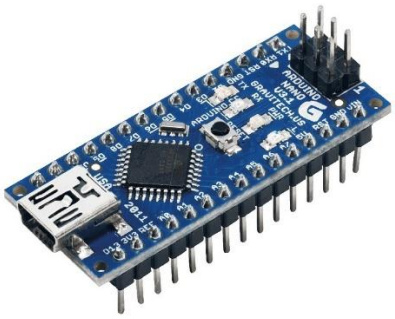


Figura 21: Arduino Nano.

3.2 MPU, 6050

El MPU 6050 (véase Fig 22) combina tanto acelerómetro como giroscopio, permitiendo tener 6 ángulos de libertad y mayor fiabilidad a la hora de tomar las medidas. Debido a ello, se cumplen 2 funcionalidades:

- Gracias a la presencia de un MEMS interno ("*MicroElectroMechanical System*"), se permite conocer la aceleración de la IMU respecto a la gravedad y a partir de esta, conocer tanto la aceleración como el desplazamiento.
- De la misma forma, pero en el caso del giroscopio, permite medir la velocidad angular usando el efecto Coriolis.

De esta componente, destacar las siguientes características [36]:

- Es **económico** y **ampliamente usado**. Por ello, se tienen un abanico de aplicaciones y con ello, bibliotecas, grande.
- **Emplea conexiones I2C.** Ello, facilita la conexión a microcontroladores y en el caso en el que se quisieran mediciones más exactas, se puede conectar un magnetómetro e integrar la información con las mediciones del acelerómetro y giroscopio.
- Presenta **rangos amplios para la detección del movimiento**, los cuales son programables por el usuario. Para el acelerómetro: ±2g, ±4g, ±8g y ±16g. Para el giroscopio: ±250, ±500, ±1000 y ±2000 º/sec (dps). Permitiendo la detección tanto de movimientos lentos como rápidos a gusto del usuario.
- Tiene un **buffer FIFO** (*"First in, First Out"*) de 1024 bytes. Asimismo, consigue **ahorrar energía** mediante lecturas por ráfagas del controlador y entrando a modo ahorro de energía cuando se hace la adquisición de medidas.
- El **acelerómetro** está conformado por **3 masas independientes**, induciendo el movimiento de manera independiente y por tanto, mayor capacidad discriminatoria. Gracias a su arquitectura, se disminuyen **las variaciones de fabricación**, homogeneizando medidas entre diferentes chips, y **menor sensibilidad a la deriva térmica**. Además de ello, cuenta con un **sensor medidor de la temperatura** del chip.
- Para la realización de las medidas del **giroscopio**, el MPU está conformado por **tres giroscopios MEMS**, uno por cada eje. La señal generada por la vibración es amplificada, demodulada y filtrada.
- Finalmente, presenta un chip llamado ***"Digital Motion Processor"*** **(DMP),** aliviando la carga computacional del controlador y simplificando la

arquitectura del software. El objetivo es, de manera automática, fusionar y procesar la información proveniente del acelerómetro, giroscopio y de un posible tercer sensor y emplear algoritmos para presentar la información, por ejemplo, en cuaterniones a medida que se encuentre disponible.

Por todas estas razones, se decidió emplear el MPU 6050: fácil de implementar, como se verá con posterioridad, posibles ruidos tales los de alta frecuencia se hayan minimizados gracias a softwares internos y a su arquitectura, posibilidad de adición de más sensores y costo reducido.

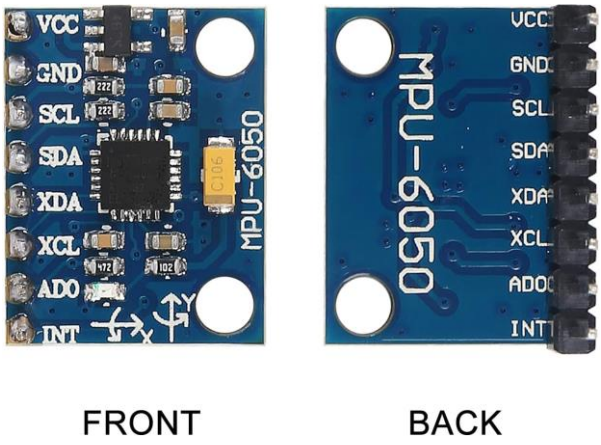


Figura 22: MPU-6050 vista anterior y posterior.

3.3 Motor Vibrador PWM.

Para la **generación de alertas**, se decide emplear **un motor vibrador PWM** (véase Fig 23). Por definición, es un dispositivo generador de efectos vibratorios en el momento en el que la entrada PWM es alta. Si no se modifica su configuración, la velocidad de giro es de 9000 rpm ("revoluciones por minuto"). Su **rango de funcionamiento es de 3.3V a 5.3V**, **mayor voltaje implica mayor rpm** [37].

Aunque se pueda pensar que es contradictorio el empleo de un módulo de vibración con el funcionamiento del MPU 6050, en las siguientes secciones se entenderá cómo se aprovechan las derivas generadas.

Se antepone su uso a un módulo generador de sonidos. En primera instancia, porque se puede dar la situación en la que el sonido sea demasiado bajo en comparación con el entorno. Asimismo, hay que considerar el sonido empleado. No se desean sonidos agudos que puedan inducir ansiedad o estrés en el usuario y atendiendo a la edad del paciente, no ser audibles.

Por ello, se propone emplear vibraciones. De uso más cómodo y con contacto, casi, íntimo con la piel, el usuario es capaz de detectar perfectamente dichas derivas en el movimiento. A través de la experimentación y del fin, se puede introducir en la programación un rango amplio de vibraciones, pues, únicamente, habría que variar la cantidad de voltaje que llega al dispositivo.

Figura 23: Módulo Vibratorio PWM

3.4 Módulo DSD-TECH HM-10.

Para poder realizar la **conexión Arduino-Ordenador**, se eligió el **módulo bluetooth HM-10** (véase Fig 24). **La principal característica** de este módulo es la posibilidad **de funcionar como maestro o esclavo**, fundamental para la aplicación. Cuando se encuentre como maestro, enviará indicaciones al Arduino para saber el modo de operación y en el momento en el que sea esclavo, se dedicara a enviar la información generada por el MPU y procesada por el controlador al ordenador.

Mediante comandos UART y desde la consola de Arduino, se puede **configurar diferentes parámetros del módulo** tales la tasa de Baudios, si se desea la conexión automática o el PINCODE para permitir la vinculación. El **voltaje de trabajo del módulo es 3.3V**, siendo posible su alimentación mediante un pin de Arduino.

Para **la señalización de la conexión**, dispone de un **led rojo**, el cual parpadeará si no hay un dispositivo vinculado y será permanente en el momento en el que exista esta. Algo a considerar. es que emplea **la tecnología BLE** *("Bluetooth Low Energy"*), protocolo creado por la empresa *"Bluetooth SIG"* para dispositivos de salud, fitness y *beacons*. Este, permite tener el **mismo alcance que el protocolo normal, pero de consumo menor** [38,39].

3.5 Acelerómetro MMA8451

Hay ciertos ejercicios en los cuales se necesita hacer una medición de cadera para conseguir una buena práctica de estos. Por ello, como segunda unidad de medición se decidió usar el MMA8451 de la casa comercial "*Adafruit* "(véase Fig 25).

Como características principales presenta [40]:

- **Resolución de ±2g, ±4g y ±8g** con un **filtro paso alto** incluido en este.
- **Conexión** se realiza mediante **I2C**.
- Tiene la **capacidad de examinación automática**. Para la calibración de este, el MMA8451 se puede colocar en estado estacionario y cuando se activa la función "self-test" induce una fuerza electroestática de actuación.

- Tiene **dos funcionalidades interesantes** para este ámbito. El modo **"Wake-Sleep"** en el que, el acelerómetro pasa a estado dormido, disminuyendo considerablemente la tasa de muestreo y posteriormente, ha estado despierto aumentando dicha tasa. Esta funcionalidad, se puede combinar con el **sensor de movimiento** incluido en el MMA. Para este último, se selecciona un umbral de aceleración que, si es superado, realizará una función u otra como aumentar la sensibilidad. Ambas cosas fusionadas, permiten que el consumo del dispositivo disminuya, aumentando la vida útil de las posibles baterías a incluir.

Estas son las características consideradas para seleccionar el MMA8451 como acelerómetro auxiliar: bajo consumo, uso generalizado e implementado en Arduino por la comunidad y los fabricantes.

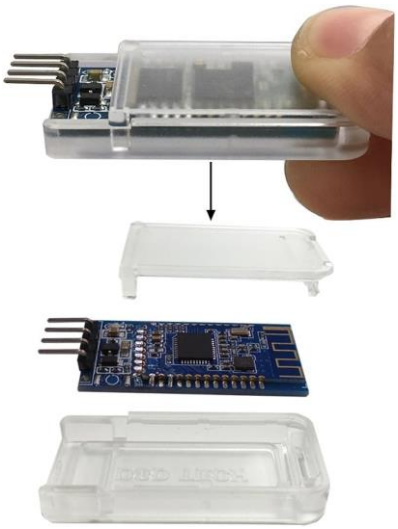


Figura 24: DSD-TECH HM-10

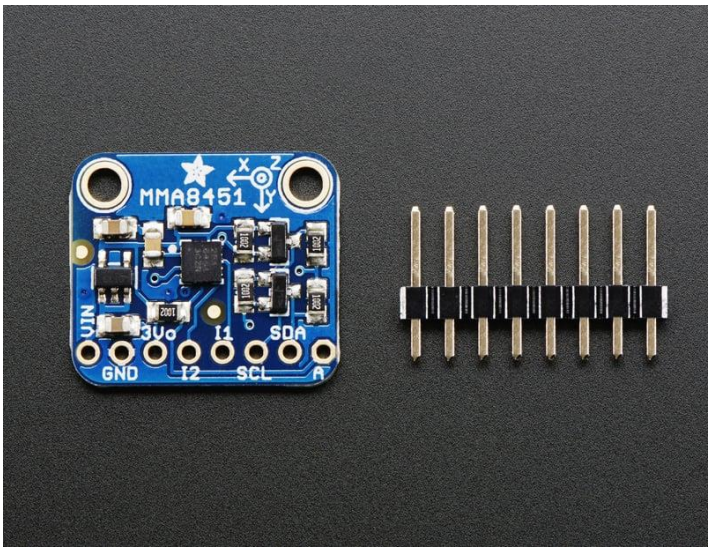


Figura 25: Adafruit MM8451

3.6 Montaje del Dispositivo

El prototipo final se puede verse en la Fig.26. Este, sería el módulo principal y el que se colocaría en el paciente para obtener las mediciones de la rodilla.

Como se puede observar, se pretende tener el vibrador lo mas cercano a la rodilla mientras que el acelerómetro y el módulo Bluetooth queden en la zona exterior del montaje. El tamaño es de 5.5 x 4.5 mm, pudiéndose minimizar más. Resaltar el uso de un tipo diferente de cables que permitan adaptarse a las vibraciones y no sean totalmente rígidos.

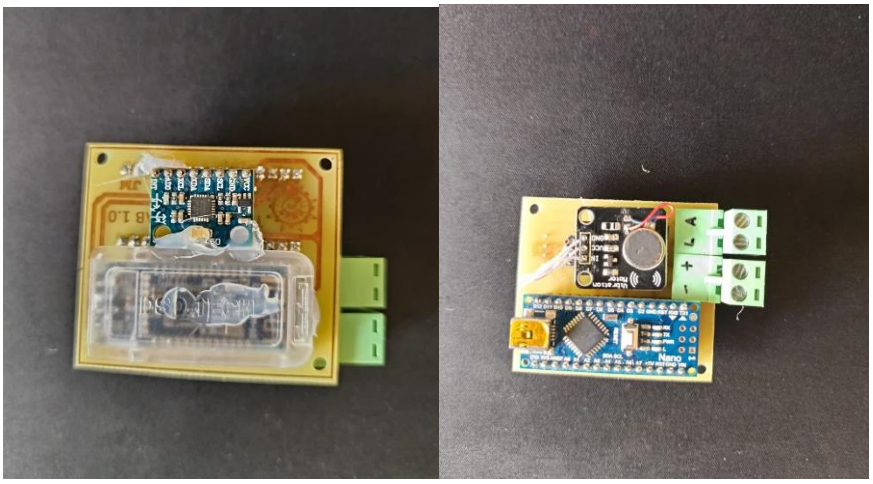


Figura 26: Modulo principal. Izquierda: Visión Anterior Derecha: Visión Posterior

En la Fig.27, se muestra el módulo complementario. De menor tamaño, solamente se presenta el acelerómetro MMA8451

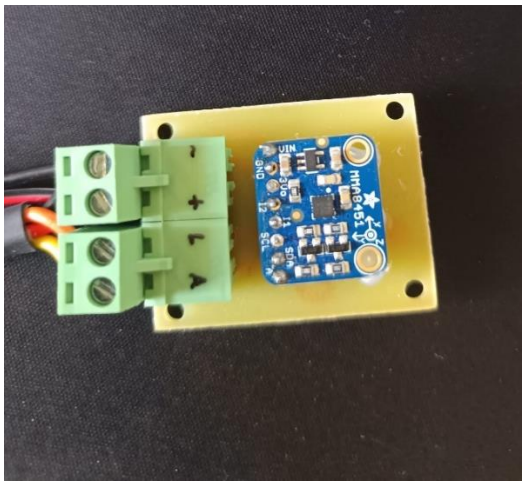
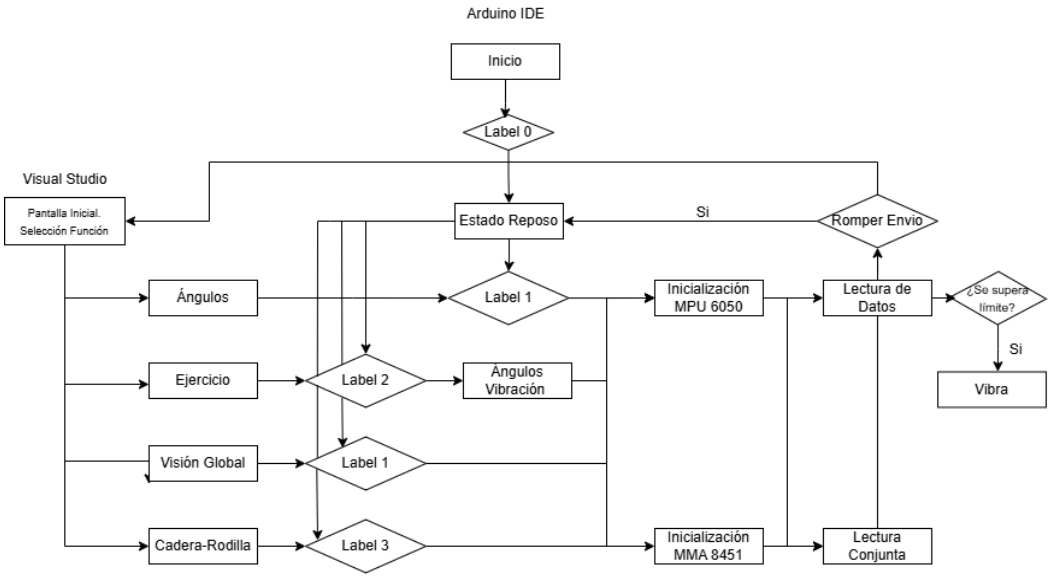


Figura 27: Módulo Auxiliar.

El diseño de la PCB se realizó mediante el software Proteus.

4. Descripción del software

En esta sección, se explicará la programación del software tanto la del Arduino IDE como la de Visual Studio, detallando sus peculiaridades. En el siguiente diagrama de bloques, se puede tener una visión global de cómo funciona la conexión entre el Arduino IDE y Visual Studio:



4.1 Arduino IDE.

La **declaración de bibliotecas** es la siguiente:

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
#include "SoftwareSerial.h"
#include "Adafruit_MMA8451.h"
#include "Adafruit_Sensor.h"
```

“I2Cdev”, “Wire.h” y “SoftwareSerial” permiten la correcta comunicación entre Arduino y los diferentes componentes planteados para el prototipo. La necesidad de emplear tres bibliotecas en apariencia heterogéneas entre sí, pero con finalidades comunes radica en que las bibliotecas “MPU6050.h” y “Adafruit_MMA8451.h” emplean diferentes funciones para la inicialización de la conexión entre componentes. Siendo un caso especial “SoftwareSerial”, el cual permite realizar las conexiones vía Bluetooth.

En relación con las bibliotecas de los hardware, resaltar la necesidad de leer el código fuente, pues la forma de tomar las medidas son diferentes y por ello, su tratamiento, también, lo es.

Antes de la visualización y explicación de la programación, se **expone su funcionamiento**:

- Se **plantean 4 casos**:
    1. **“label 0”:** reposo del Arduino sin medidas.
    2. **“label 1”:** para medir únicamente el movimiento natural y graficarlo en la interfaz.
    3. **“label 2”:** para medir y visualizar los ángulos y si se sobrepasan ciertos límites, poner en marcha el vibrador,
    4. **“label 3”**: con el mismo funcionamiento que “label 1”, pero funcionando el acelerómetro auxiliar.
- En **la interfaz**, se seleccionará uno de los casos y será recibido por el Arduino, vía Bluetooth, el número característico del “label”. Para lograrlo, se hace el llamamiento a la función “Vibración_bluetooth” la cual, mientras el módulo HM-10 se encuentre disponible, sin recibir ni enviar información, permite al controlador hacer lecturas continuas del buffer del módulo. Mediante esta, se permite el envió de información de los sensores como la recepción de la funcionalidad escogida.

Siendo este el funcionamiento general del código, se propone la explicación de los diferentes casos, teniendo como **estructura de control principal** la función **“switch…case”,** la cual permite mediante la declaración de “case(var)”, entrar en un modo de operación u otro.

4.1.1 Case 0

El Arduino se encuentra esperando un comando, no hay demanda de datos y el vibrador esta apagado.

```
case label0:
      Vibracion_bluetooth();
      digitalWrite(vibrador, LOW);
      break;
```

4.1.2 Case 1

Definida como la **medición simple de los ángulos**. Es la propuesta para que el paciente describa un movimiento lento y enviar dicha información a la interfaz.

```
case label1:
      Vibracion_bluetooth();
      MPU_Tilt();
      BT.print(accel_ang_x_6050);
      BT.print("M");
      BT.print(accel_ang_y_6050);
      BT.print("M");
      BT.print(accel_ang_z_6050);
      delay(100);
      break;
```

Se hace **llamamiento a la función "MPU.Tilt()",** la cual permite la extracción de la información procesada del MPU 6050. Se define como:

```
Void MPU_Tilt()
{
  mpu.getAcceleration(&ax, &ay, &az);
  mpu.getRotation(&gx, &gy, &gz);

  dt = (millis()-tiempo_prev)/1000.0;
  tiempo_prev=millis();

  float gyro_rate_x = gx / 131.0;
  float gyro_rate_y = gy / 131.0;
  float gyro_rate_z = gz / 131.0;

  accel_ang_x_6050 = atan(ax / sqrt(pow(ay, 2) +
pow(az, 2))) * RAD_TO_DEG;
  accel_ang_y_6050 = atan(ay / sqrt(pow(ax, 2) +
pow(az, 2))) * RAD_TO_DEG;


  angle_x_6050 = 0.98 * (accel_ang_x_6050 +
gyro_rate_x * dt) + 0.02 * accel_ang_x_6050;
  angle_y_6050 = 0.98 * (accel_ang_y_6050 +
gyro_rate_y * dt) + 0.02 * accel_ang_y_6050;
  angle_z_6050 += gyro_rate_z * dt;

  accel_ang_x_6050 = angle_x_6050;
  accel_ang_y_6050 = angle_y_6050;
  accel_ang_z_6050 = angle_z_6050;
```

En este caso, lo primero que se calcula es el ángulo de inclinación de los ejes X e Y mediante las medidas del acelerómetro y del eje Z mediante el giroscopio, al no tener magnetómetro. Posteriormente, se fusionan con las del giroscopio mediante el empleo de un filtro complementario. Mediante este último, mediciones largas serán descritas por el acelerómetro, no acumula error con el tiempo, y mediciones cortas serán descritas por el giroscopio, robusto frente a ruido y a aceleraciones bruscas.

Gracias a este planteamiento, el sensor se puede llevar durante un espacio corto de tiempo, por ejemplo, con series de ejercicios y reposo o se puede emplear para un monitoreo continuo, como podría ser la marcha.

4.1.3 Case 2

Este bloque de programación tiene sus particularidades respecto a los anteriores. La programación sirve para que los vibradores entren en acción tras superar cierto límite impuesto por el usuario.

```
case label2:
      if (millis() - tiempo_muestras >= 150)
      {
        Vibracion_bluetooth();
        MPU_Tilt();
        BT.print(accel_ang_x_6050);
        BT.print("M");
        BT.print(accel_ang_y_6050);
        BT.print("M");
        BT.print(accel_ang_z_6050);
        if (millis() - tiempo_vibrador >= 2000)
        {
          tiempo_vibrador = millis();

          if (vibrador_y < accel_ang_y_6050)
            {

              float valor_x = 90;
              int accel_ang_y_6050_ant = -50;
              int accel_ang_z_6050_ant = -50;

              delay(10);
              BT.print(accel_ang_y_6050_ant);
              BT.print("M");
              BT.print(valor_x);
              BT.print("M");
              BT.print(accel_ang_z_6050_ant);

              digitalWrite(vibrador, HIGH);

            }
         }
        else
        {
          digitalWrite(vibrador, LOW);
        }

        tiempo_muestras = millis();

      }
      break;
```

La **razón de introducir condicionantes "if",** comparando el reloj del Arduino con la variable "tiempo_vibrador", es para **evitar** que haya **desbordamiento en la interfaz** (en el Case 2, se emplea "delay") y se consigue el envío de una muestra cada 150 ms.

Se asume que no se precisa del rastreo continuo de las muestras, dada la mínima necesidad de medir variaciones pequeñas de ángulos y con este tiempo se da el correcto funcionamiento del prototipo.

La siguiente comparación de tiempos, es para **evitar el funcionamiento ininterrumpido del vibrador**; habría que experimentar con pacientes para saber el correcto espaciado temporal.

Otra de las hipótesis en las que se fundamenta el espaciado entre muestras, es el movimiento lento del paciente en rehabilitación y por tanto, el descarte de movimientos bruscos que puedan llegar a afectar negativamente al proceso.

En secciones anteriores, se comentó que los vibradores pueden perjudicar a las medidas del acelerómetro. No obstante, este efecto se convierte en ventaja en el presente trabajo. Si se imagina el caso en el que el rehabilitador no escuche el vibrador o el paciente no comunique la activación de este, se obliga a que el "valor_x" aumente a 90º, mostrando en qué eje se ha producido la vibración mientas que las dos restantes se vuelven negativas para lo contrario. Únicamente se expone, como ejemplo explicativo, el caso en el que el ángulo en el eje Y supere el fijado por el rehabilitador. Se debería seguir el mismo procedimiento en el resto de los casos posibles.

4.1.4 Case 3

Corresponde a la **combinación de medidas** entre el acelerómetro principal y el auxiliar.

```
case label2:
      Vibracion_bluetooth();
      ADA_Tilt();
      MPU_Tilt();
      float diff_x = accel_ang_x_6050 - accel_ang_x_MMA;
      float diff_y = accel_ang_y_6050 - accel_ang_y_MMA;
      float diff_z = accel_ang_z_6050 - accel_ang_z_MMA;

      BT.print(diff_x);
      BT.print("M");
      BT.print(diff_y);
      BT.print("M");
      BT.print(diff_z);
    break;
```

Destaca el empleo de variables tipo "float" y se debe a la manera en la que están definidas los datos en crudo del acelerómetro "Adafruit_MMA8451". Para el MPU, se definen tres variables independientes, asociándose cada una de ellas a un eje. No obstante, en el "Adafruit" se tiene una estructura de datos a la cual se ha de acceder de manera independiente.

Mediante esta explicación, se tiene una visión global del comportamiento del prototipo. Sin embargo, aún quedan dos preguntas por responder.

Tras cada envío de valor, ¿por qué se manda una M?

La respuesta se debe a cómo Visual Studio trata los datos ASCII. Se probó de varias maneras, por ejemplo, por cada envío hubiese un salto de línea o se generase un espacio. Sin embargo, por cómo se plantea el tratamiento de datos para su visualización, se explica posteriormente, y las funciones disponibles para su lectura, esas y otras opciones fueron descartadas por incompatibilidades.

Por ello, lo más sencillo y efectivo de realizar era enviar un dato que fuera estático y el programador tuviera consciencia de ello. Además, se permite solucionar el gran problema de las variaciones en decimales. Se trata de la misma forma si se mide un 1, 1.111 o 20.432 y gracias a ello, las listas que almacenan datos en Visual Studio pueden variar de tamaño acorde a las entradas y no se generan problemas de lectura.

¿Cómo se realiza el envío o lectura de los datos vía bluetooth?

El envío de estos se realiza de manera bastante sencilla gracias a la biblioteca "Software.Serial" con la siguiente línea de comando:

```
BT.print(accel_ang_x_6050);
```

Sin embargo, para la lectura de datos enviados por la interfaz, limite de los vibradores y la funcionalidad, se aplica lo siguiente:

```
void Vibracion_bluetooth()
{
  while (BT.available())
    {

      buffer[bytesRead++] = BT.read();
    }

    if (bytesRead == 8)
    {
      processData(buffer, bytesRead);
      bytesRead = 0;
      memset(buffer, 0, sizeof(buffer));

    }
    else if (bytesRead == 1)
    {
      int commandValue = (int)strtol(buffer, NULL, 16);
      var = (State)commandValue;
      bytesRead = 0;
      memset(buffer, 0, sizeof(buffer));
    }
}
```

El bucle “while” sirve para que siempre que el módulo se encuentre inactivo, haga lecturas de su buffer. Gracias al espaciado entre muestras, se evita que no haya problemas con esto último. De esa forma, se **plantean dos opciones**:

- Si la **lectura de bytes llega a 8**, se introduce en la función “processData” y se hace limpieza de la memoria. Si dicha cantidad es la que se lee, se asume que, por diseño, los datos que hay son límites del vibrador.
- Si **la lectura es 1**, implica que se ha escogido una funcionalidad y el valor de entrada se convierte a decimal y la variable “var”, variable de estado, toma el valor recibido.

La función “processData” es la siguiente:

```
void processData(char* data, int length)
{
  if (length == 8)
  {
    char tempX[3] = {data[0], data[1],
'\0'};
```

```
    char tempY[3] = {data[3], data[4],
'\0'};
    char tempZ[3] = {data[6], data[7],
'\0'};

    vibrador_x = atof(tempX);
    vibrador_y = atof(tempY);
    vibrador_z = atof(tempZ);
  }
```

La variable buffer almacena 8 valores: 6 asociados a los límites de ejes para la vibración y 2 caracteres que son comas o espacios. Mediante el uso de la función “atof”, transforma variables caracteres a numéricos, se obtienen los valores asociados a cada uno de los límites. Se escoge el tipo de variable “char”, ya que Arduino permite tratar a los bytes como caracteres y la función “atof” sirve como traductor del código ASCII.

Cabe mencionar que independientemente de lo que se reciba, al ser “tempX” y sus homónimos variables char, estos se transforman a código ASCII. Es decir, en el caso en el que se recibiese un carácter “A”, este se almacenaría como 65, valor numérico, y “atof” lo traduciría para permitir su tratamiento. Sin embargo, han de ser “char” por comportamiento interno de la función.

4.2 Desarrollo en la Plataforma Visual Studio.

Antes de comenzar con la explicación de los detalles del código, se emplea un espacio para responder a qué es UWP (“*Universal Windows Platform*”)

, la tecnología BLE y biblioteca asociada y conceptualizar la metodología empleada.

4.2.1 ¿Qué es la plataforma Universal de Windows?

La Plataforma Universal de Windows, UWP, es una API presente en Windows 10 y Windows 11. Este, provee un conjunto de librerías que facilitan el desarrollo de aplicaciones para aquellos dispositivos, ordenadores, Xbox, móviles o wearables, que tengan instalado el sistema operativo Windows [41,42].

Las **características de aplicaciones** desarrolladas con UWP [42], destacan por:

- **Declaración de los recursos y datos de dispositivos** a los que se accede y con ello, la autorización del usuario.
- Emplea **funcionalidades específicas del dispositivo** y **adapta la interfaz** a tamaños de pantalla y resoluciones diferentes.
- Si la aplicación **emplea las API principales**, esta podrá ser ejecutada por **cualquier dispositivo de Windows**.
- Se puede **publicar en Microsoft Store,** provocando la instalación de esta en PC, tabletas, Xbox, HoloLens, Surface Hub y dispositivos IoT.
- **Diversidad en los códigos de programación empleados**. Para la escritura de código se puede emplear C#, C++, Visual Basic y Javascript. Para la creación de la interfaz de usuario, WinUI, XAML, HTML o DirectX.

Para la creación de la aplicación, se ha seleccionado C# como código de programación y XAML para la creación de la interfaz.

4.2.2 Bluetooth Low Energy en UWP apps.

Bluetooth Low Energy (BLE) es un estándar que define protocolos para el descubrimiento y comunicación entre dispositivos de bajo consumo energético, tales sensores, relojes inteligentes o dispositivos médicos.

Las **características** más llamativas del protocolo [43,44**] son las siguientes**:

- **Bajo consumo de energía**: al revés que el Bluetooth clásico el cual, mantenía continuamente la conexión con el dispositivo y por tanto, la transmisión de información, BLE lo hace de manera periódica, consumiendo menor energía.
- **Inmediatez**: realiza las conexiones de manera más rápida y la transmisión de información se realiza de manera que la latencia puede ser inferior a 3 ms.
- **Menor coste**: complejidad menor al usar menor cantidad de chips.
- **Amplias Compatibilidades**.
- **Mayor Alcance**: con las nuevas versiones, el rango de transmisión tiene un alcance teórico al aire libre de 100 metros
- **Mayor Seguridad**: es compatible con multitud de formas de emparejamiento tales como método de emparejamiento de claves de acceso, método de emparejamiento fuera de banda o el método de emparejamiento Just Works.

A grandes rasgos, el descubrimiento de dispositivos se realiza mediante el protocolo GAP ("*Generic Access Profile*") y la comunicación mediante el protocolo GATT ("*Generic Attribute Profile*").

4.2.3 Generic Access Profile

El protocolo GAP dicta cómo los dispositivos interactúan unos con otros a bajo nivel, fuera de la pila de protocolos. Además de ello, se puede considerar como la capa de control superior del BLE, especificando como se llevan al cabo los procedimientos de control, como se hace el descubrimiento, la conexión y el establecimiento de seguridad permitiendo de esta forma la interoperabilidad y el intercambio de datos entre dispositivos de casas comerciales distintas [45].

En este caso, se definen **cuatro roles** que el dispositivo puede adoptar [45]:

- ***Broadcaster***: optimizado para aplicaciones cuyo objetivo es la transmisión de datos regularmente. En la práctica, este rol es asignado a aquellos dispositivos capaces de transmitir y recibir información. El broadcaster envía datos en paquetes de publicidad (“*advertising packets”)* en lugar de paquetes de datos de conexión *(“connection data packets”*) y la información esta disponible para cualquier dispositivo oyente.
- ***Observer***: optimizado para aplicaciones cuya finalidad es recolectar datos.
- ***Central***: es el rol que permite la conexión de dispositivos a la red y siempre es el que inicia la conexión. Se requiere que este rol lo adquieran dispositivos con CPUs potentes para mantener la conexión entre múltiples dispositivos.
- ***Peripheral***: emplea los paquetes de publicidad para que los dispositivos centrales lo encuentren y establecer la conexión. A diferencia del rol central, este se centra en tener una o pocas conexiones por la limitación de recursos.

4.2.4 Generic Attribute Profile

El GATT se construye sobre el ATT *(“Attribute Protocol”)* y añade una jerarquía y un modelo de abstracción de datos. Define como los datos se organizan y se intercambian entre aplicaciones. Se introduce el concepto de servicio, consistente de una o más características, definidas como la unión de una pieza de datos del usuario junto con metadatos (información descriptiva sobre un valor como el nombre visible, las unidades o propiedades). Mediante este se definen **dos roles**, compatibles con los mencionados en el GAP [45,46,47]:

- **Cliente**: definido como el que envía la petición al servidor y recibe respuestas de este. En el momento en el que se cumple el descubrimiento, puede comenzar con la lectura y escritura de atributos.
- **Servidor**: recibe las peticiones del cliente y devuelve respuesta. Su función principal es la de almacenamiento y accesibilidad de la información para el cliente, organizada en atributos.
-

Además de ello, a cada dispositivo se le asocia un UUID *(“Universally Unique Identifier”) ,* numero de 16 bytes, el cual sirve como identificador [46].

Atributos.
El atributo es el tipo de dato más pequeño definido por el GATT. Son piezas de información que pueden contener datos relevantes sobre la estructura [46].

Servicios.
Los servicios agrupan atributos conceptualmente relacionados en una sección común del servidor GATT. Por ello, se pueden definir como una sucesión de definiciones del servidor, empezando cada una de ellas con un único atributo que marca el comienzo del servicio [45].

Características.
Se pueden entender como almacenadores de información del usuario. Siempre incluyen dos atributos: la declaración de la característica (información del usuario actual) y el valor de esta (es un atributo completo que contiene la información del usuario) [45].

4.2.5 Explicación del Código.

Para comenzar con el descubrimiento de dispositivos cercanos, hay que activar el bluetooth del ordenador e inicializar la clase DeviceWatcher:

```
void InitializeDeviceWatcher()
   {
       string[] requestedProperties = {
"System.Devices.Aep.DeviceAddress",
"System.Devices.Aep.IsConnected" };
       DeviceWatcher deviceWatcher =
           DeviceInformation.CreateWatcher(

BluetoothLEDevice.GetDeviceSelectorFromPairingSt
ate(false),requestedProperties,
DeviceInformationKind.AssociationEndpoint);

       deviceWatcher.Added +=
DeviceWatcher_Added;
       deviceWatcher.Removed +=
DeviceWatcher_Remove;
       deviceWatcher.Start();
   }
```

El “device watcher permite enumerar dinámicamente todos aquellos dispositivos que haya alrededor, informando de su descubrimiento, eliminación o cambio en la posición una vez que se haya completado el proceso. Si este se ha completado positivamente, se comenzará a recibir lo que Visual Studio denomina “DeviceInformation”, entendiéndose como las propiedades que caracterizan a cada dispositivo presente en la lista [48].
Se definen funciones que permiten crear listas dinámicas y que podrán ser leídas en la sección anexos.
Una vez que se obtienen todos los dispositivos, el usuario ha de buscar el nombre “DSD TECH”, identificador del módulo. Si no fuera así, la conexión no se realizaría y se generaría un mensaje de error (véase Fig 27).

La idea de que el usuario tenga que realizar dicha elección, radica en que se quiere que se tenga consciencia de si el dispositivo está conectado o no, en vez de hacerlo de manera automática. Se intentan paliar errores derivados de los automatismos de la rutina y errores debidos a fallos de conexión o alimentación del dispositivo.

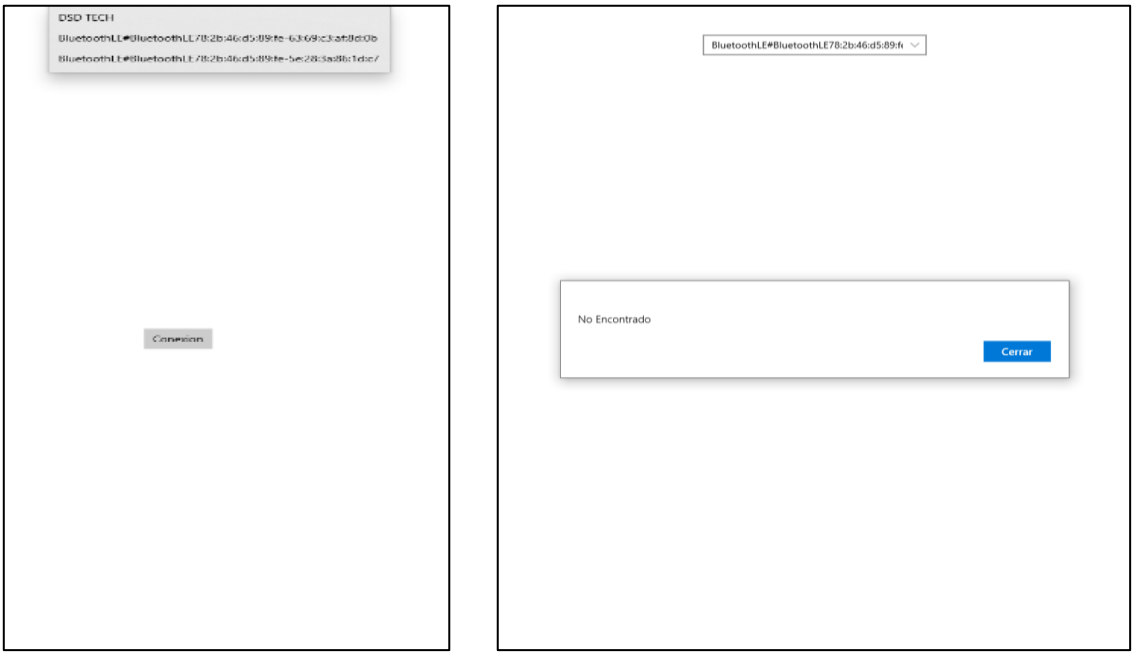


Figura 28:Izquierda: Lista de Dispositivos de mi alrededor. Derecha: Mensaje de error al seleccionar dispositivo incorrecto

Asumiendo que la conexión se ha realizado correctamente, se muestra paso a paso las verificaciones que se realizan.

En primera instancia, mediante el "id" del dispositivo, obtenido del objeto "BluetoothLEDevice", se conocen todos los servicios, atributos, uuids y estado de conexión del dispositivo conectado. Gracias a ello, se pueden conocer los servicios que se tienen:

```
BluetoothLEDevice bluetoothLeDevice = await
BluetoothLEDevice.FromIdAsync(DSD_TECH.Id);

GattDeviceServicesResult result = await
bluetoothLeDevice.GetGattServicesAsync();

if (result.Status == GattCommunicationStatus.Success)
{
    var services = result.Services;
    Debug.WriteLine(services);
    foreach (var servicios in services)
    {
        Debug.WriteLine(servicios.Uuid);
    }
```

Se hace busca del servicio deseado:

```
foreach (var service in services)
    if (service.Uuid.ToString("N").Substring(4, 4) ==
DSD_TECH_UUID)
    {
        Debug.WriteLine("Se ha encontrado el servicio");

        GattCharacteristicsResult
caracteristica_servicio_resultado = await
service.GetCharacteristicsAsync();
```

Como se desean realizar operaciones de lectura, se obtienen funciones de la característica seleccionada y se comienza con la lectura.

```
if (properties.HasFlag(GattCharacteristicProperties.Read))
{
    GattCommunicationStatus status = await
_caracteristica.WriteClientCharacteristicConfigurationDescriptor
Async(

GattClientCharacteristicConfigurationDescriptorValue.Notify);
    //GattReadResult result2 = await
caracteristica.ReadValueAsync();
    //Debug.WriteLine(result2);
    if (result.Status == GattCommunicationStatus.Success)
    {
        DispositivoComboBox.Visibility = Visibility.Collapsed;
        BotonConexion.Visibility = Visibility.Collapsed;
        //CuadroSalidaDatos.Visibility = Visibility.Collapsed;

        Angulos_.Visibility = Visibility.Visible;
        Modulo_Vibración.Visibility = Visibility.Visible;
        Global.Visibility = Visibility.Visible;

        Fusion_acelerometros.Visibility = Visibility.Visible;


    }
```

Como se podrá observar a lo largo del código, hay una cantidad considerable de condicionantes para aumentar la robustez del código y evitar la conexión con dispositivos no deseados. Varios de estos, se han conseguido mediante la experimentación. Por ejemplo, el UUID, no está dentro del SIG, el dispositivo no está estandarizado, se ha conseguido mediante prueba y error y visualizado por consola todos los datos que describen al dispositivo.

Una vez conseguida la conexión, queda la selección de la funcionalidad del prototipo. Solo se realiza la explicación de la función “Ángulos”, pues la estructura del resto de funcionalidades es la misma.

Tal como se explicó en la implementación en Arduino, la selección de las funcionalidades se realiza mediante el envío de numero de enteros. Para poder cumplir con ello, se genera la clase “byte” y “DataWriter” para permitir la transmisión de información al dispositivo. Se

hace una espera de 30 segundos, para evitar posibles “bugs” y se obtiene la respuesta de este por consola. Finalmente, si el resultado es “True”, se comenzará con la lectura de datos.

```
      async public void Angulos(object sender,
RoutedEventArgs e)
      {
          byte[] bytes =
Encoding.ASCII.GetBytes("1");


          var writer = new DataWriter();
          writer.WriteBytes(bytes);
          await Task.Delay(30);
```

```
          GattCommunicationStatus result = await
_caracteristica.WriteValueAsync(writer.DetachBuffer
());
_caracteristica.ValueChanged += Lectura_Datos;

      }
```

La función “Lectura_Datos” lee el buffer del dispositivo bluetooth. Posteriormente, se obtiene una lista de índices cuya finalidad es encontrar donde esta el número 77, carácter “M” en código ASCII, y se transforma a una lista, consiguiendose la división entre eje X, eje Y y eje Z.

Una vez obtenido, se hace la traducción desde código ASCII a formato “string”. Se han probado múltiples funciones para la lectura de bytes y sin embargo, ninguna fue efectiva con lo que se buscaba.
Por consiguiente, aunque parezca un gasto de recursos el paso de “string” a “float”, se puede realizar gracias a que, experimentalmente, se ha conseguido el tiempo entre muestras que evita el desbordamiento del código.
Para dar la sensación de movimiento en las gráficas, explicadas con posterioridad, se elimina continuamente el valor en el índice 0

```
private void Lectura_Datos(GattCharacteristic sender,
GattValueChangedEventArgs args)
        {

            var reader = DataReader.FromBuffer(args.CharacteristicValue);

            byte[] input = new byte[reader.UnconsumedBufferLength];


            reader.ReadBytes(input);
            List<int> indices = input
                                .Select((value, index) => new { value, index
})
                                .Where(x => x.value == 77)
                                .Select(x => x.index)
                                .ToList();

            string x_ASCII = Encoding.ASCII.GetString(input, 0,
indices[0]);
            float.TryParse(x_ASCII, NumberStyles.Float,
CultureInfo.InvariantCulture, out float x_medida);

            string y_ASCII = Encoding.ASCII.GetString(input, indices[0] +
1, indices[1] - indices[0] - 1);
            float.TryParse(y_ASCII, NumberStyles.Float,
CultureInfo.InvariantCulture, out float y_medida);

            string z_ASCII = Encoding.ASCII.GetString(input, indices[1] +
1, input.Length - (indices[1] + 1));
                float.TryParse(z_ASCII, NumberStyles.Float,
CultureInfo.InvariantCulture, out float z_medida);

            x_medidas.RemoveAt(0);
            x_medidas.Insert(x_medidas.Count, x_medida);

            y_medidas.RemoveAt(0);
            y_medidas.Insert(y_medidas.Count, y_medida);

            z_medidas.RemoveAt(0);
            z_medidas.Insert(z_medidas.Count, z_medida);


        }
```

Algo a resaltar es que, las listas están definidas de la siguiente forma:

```
private List<float> x_medidas = Enumerable.Repeat(0.0f, 21).ToList();
private List<float> y_medidas = Enumerable.Repeat(0.0f, 21).ToList();
private List<float> z_medidas = Enumerable.Repeat(0.0f, 21).ToList();
```

Son listas fijas de 21 números y por ende, lo que se mostrara en la interfaz. Dicho hecho tiene que ver con la biblioteca “Canvas”, biblioteca para la creación de gráficas en UWP. El funcionamiento base de esta, es la división de la pantalla en rejillas de espacio fijo. Por ejemplo: si se tiene una pantalla 1920 x 1080 px, y se quiere realizar 3 graficas, se tendrá por cada un espacio de dibujo de 640x360, sin contar posibles solapamientos.

Por consecuente, al tener un espacio reducido, el aumento en la cantidad mostrada puede producir que haya cambios bruscos no compatibles con el movimiento.

Si se considera el caso contrario, listas de menor tamaño, posiblemente produzcan un desbordamiento en la interfaz y la aplicación se cierre. Por ello, además de considerar la resolución de los valores, hay que ver cuál es la tasa de refresco de las gráficas, siendo este invariable por el usuario y adaptarse a esta.
Por ende, el tamaño 21 se ha conseguido a través de la experimentación.

Si se quisiera comenzar a programar un programa similar, se recomienda encarecidamente el video [49].

4.2.6 Explicación de la Interfaz.

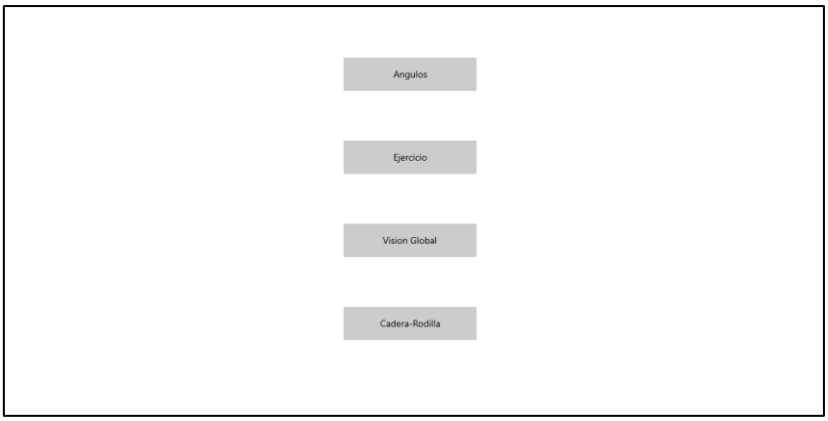
La pantalla principal tendría la siguiente forma:

En esta, se puede observar un botón y la estructura “ComboBox”:

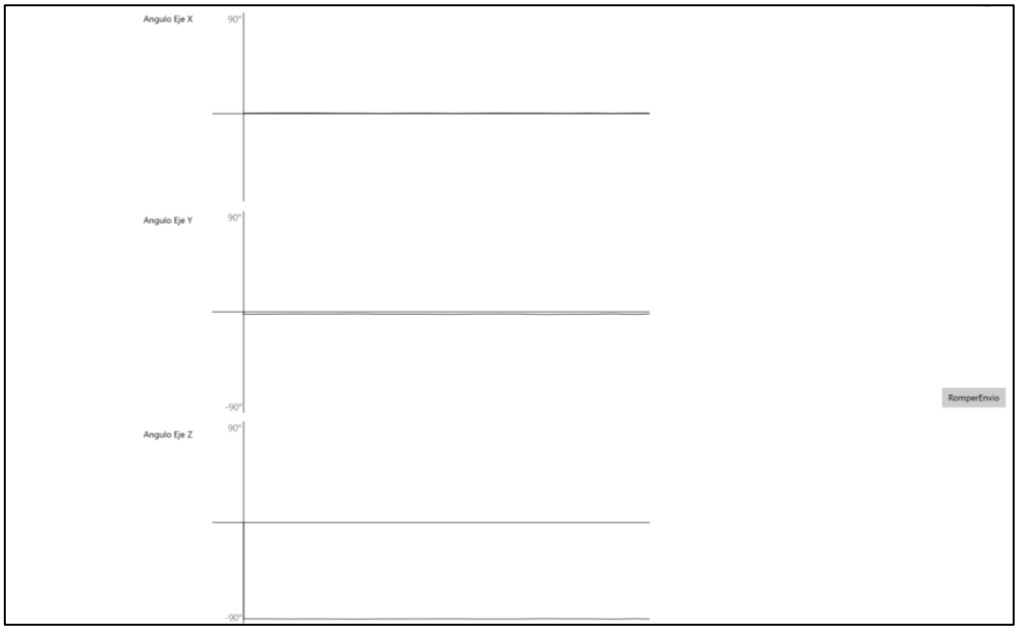
```
<!---Creacion del Primer Bloque de la Interfaz-->
<ComboBox x:Name="DispositivoComboBox" ItemsSource="{x:Bind
DispositivosAnadidos2}" DisplayMemberPath="DisplayName"
          SelectedValuePath="Id"
          HorizontalAlignment="Center"
          VerticalAlignment="Top"
          Width="300"
          Margin="20"/>
<Button  x:Name = "BotonConexion" Content="Conexion"
Click="Conexion_Click" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
```

“ComboBox” permite la interacción activa del usuario y además, se actualiza correctamente frente a adiciones o eliminaciones de dispositivos de alrededor.

Una vez encontrado el dispositivo, se muestran los cuatros botones con las cuatro funcionalidades mencionadas



Si se presiona el botón ángulos, este sería su aspecto:

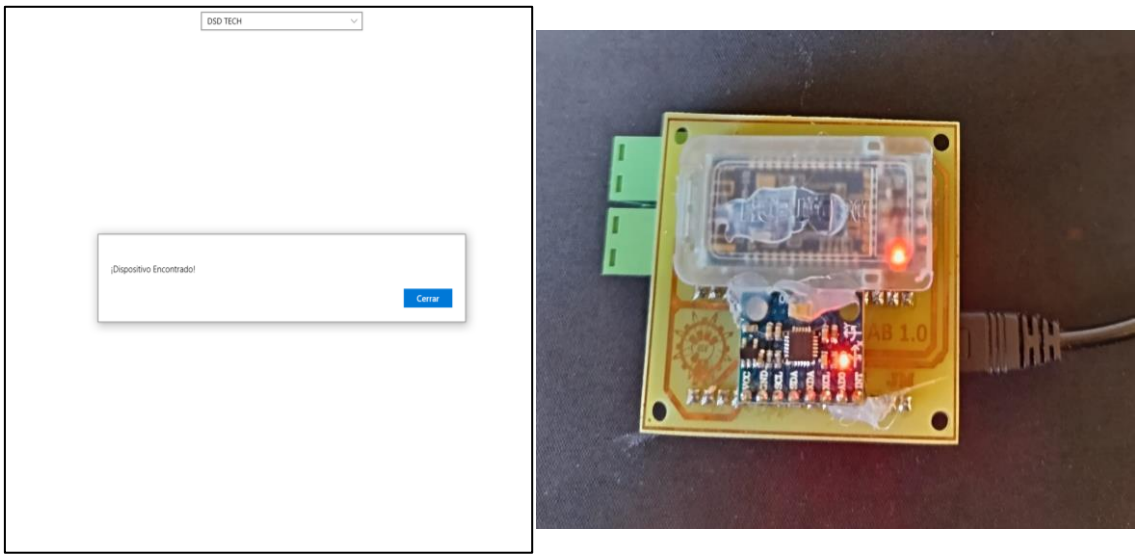


5. Funcionamiento del Prototipo.

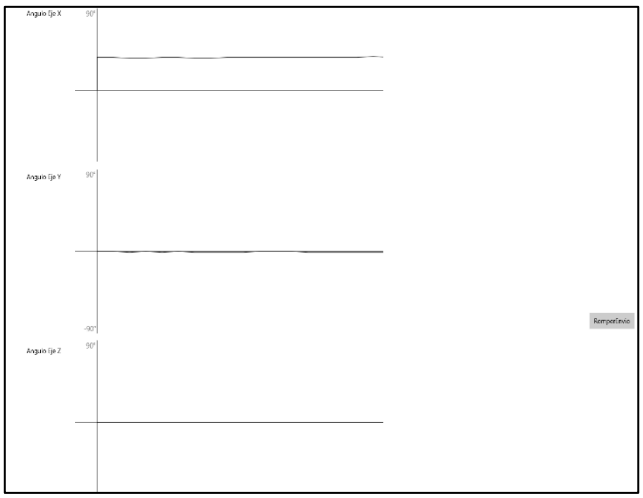
Tras explicar el hardware y sotfware, este apartado esta pensado para mostrar mediante imágenes el funcionamiento del prototipo.

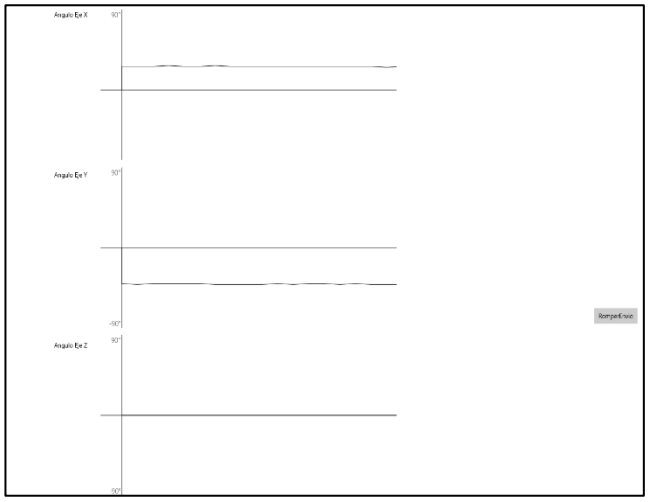
En las imágenes inferiores, se puede observar la pantalla inicial, aún no se ha hecho conexión con el dispositivo. En la imagen de la derecha, se ha intentado capturar el parpadeo del DSD, el cual señaliza la ausencia de conexión con el dispositivo:

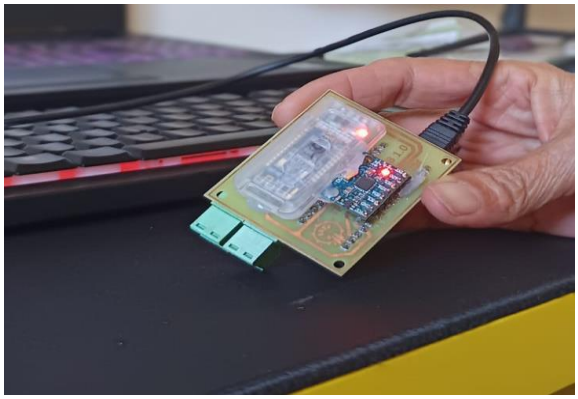
El dispositivo se ha encontrado. La luz del DSD permanece constante señalizando la conexión y aparece un mensaje: ¡Dispositivo Encontrado!:



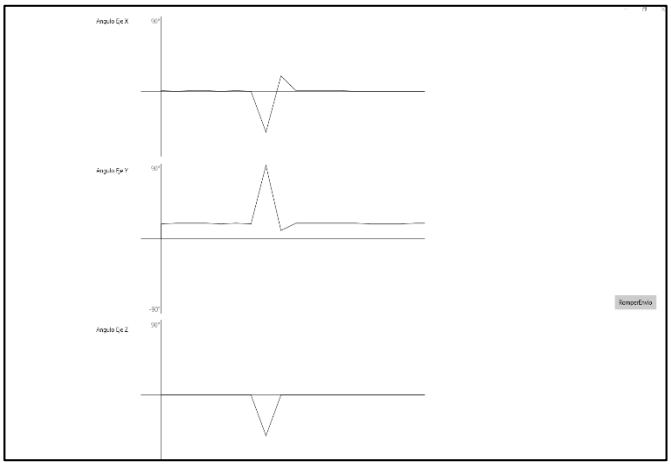
Se muestra la funcionalidad de “Ángulos”, solo moviendo el dispositivo:

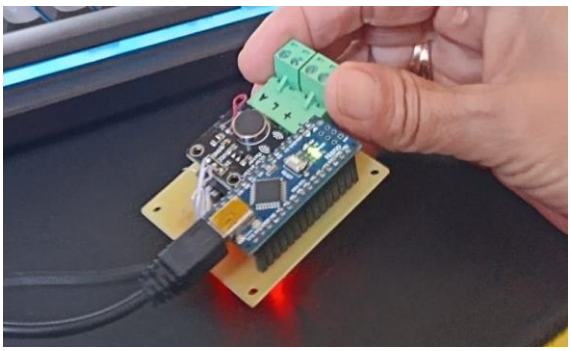






En las siguientes imágenes, se muestra un ejemplo con los vibradores activos:





6. Limitaciones

Hay una serie de limitaciones que habría que considerar en líneas futuras de trabajo:

- **Cambio del controlador:** a través de la experimentación se han enfrentado a varios problemas causados por el uso de Arduino Nano. Entre estos, destaca la poca capacidad de almacenamiento que tiene, provocando que la programación se adapte a ella y se tengan que hacer tareas de optimización exhaustivas. Sorprende que un programa complejo como el que se ha realizado sea capaz de funcionar sin problemas.
- **Posibilidad de mayor miniaturización del prototipo:** si se emplease un ESP32, el módulo DSD podría ser eliminado. Se eliminarían posibles fallos con los comandos UART y posiblemente, el envió de datos mediante Bluetooth sería más fácil.
- **Uso de Silicona:** principalmente por estética. Se debería de estudiar otras formas de sujeción de los elementos que no interfieran con el correcto funcionamiento del dispositivo.
- **Uso de otras IMUS:** se podría tener un dispositivo más robusto con la inclusión de una IMU de nueve grados de libertad.
- **Empleo de otro software de diseño de interfaz:** el objetivo de emplear UWP es porque se tienen las bibliotecas de Bluetooth accesibles. Sin embargo, el diseño mediante Canvas, biblioteca de la interfaz, es bastante complicada y limitada en recursos. Por ejemplo, un fallo que tiene es la tasa de refresco de la pantalla y aunque no sea necesaria una precisión en los ángulos, es decir en el envió de muestras, se tiene que limitar esta última para que no se cierre la aplicación.

Es importante pensar que las limitaciones han sido impuestas por el deseo de diseñar un prototipo lo más barato posible y a la vez funcional. Aunque se presencien limitaciones considerables, el prototipo funciona y supera las expectativas.

7. Conclusiones

El objetivo principal del presente trabajo era proporcionar un análisis panorámico de la tecnología, hoy presente, de la tecnología wearable en aplicaciones médicas.

Los objetivos secundarios corresponden a:

1. Búsqueda y análisis de aplicaciones potenciales.
2. Diseño de hardware.
3. Diseño de software.
4. Valoración del comportamiento del equipo.

Se considera que los objetivos han sido cumplidos y superado las expectativas. Se presenta la evolución histórica de los dispositivos, su impacto económico y se exploran aplicaciones diversas en el ámbito médico.

Posteriormente, el prototipo se pudo crear gracias a la colaboración del equipo de rehabilitación del Hospital Clínico de Valladolid. Gracias a ello, se pudo obtener retroalimentación importante que le dio al forma final de proyecto. Por ello, se considera que el dispositivo puede ser aplicado a la práctica clínica. Los componentes finales se

seleccionaron mediante prueba y error y considerando cual es el que mayor potencial tiene.

Por consecuente se concluye que el dispositivo cumple con las funciones propuestas, es una solución económica y fácil de usar. Permite la disminución de la carga de trabajo de los profesionales a la vez que fomenta la independencia del paciente, intentando eliminar la barrera que podría suponer la presencia de dolor.

8. Bibliografía.

[1] Ometov, A., Shubina, V., Klus, L., Skibińska, J., Saafi, S., Pascacio, P., Flueratoru, L., Gaibor, D. Q., Chukhno, N., Chukhno, O., Ali, A., Channa, A., Svertoka, E., Qaim, W. B., Casanova-Marqués, R., Holcer, S., Torres-Sospedra, J., Casteleyn, S., Ruggeri, G., . . . Lohan, E. S. (2021). A Survey on Wearable Technology: History, State-of-the-Art and Current Challenges. *Computer Networks*, *193*, 108074. https://doi.org/10.1016/j.comnet.2021.108074

[2] Zolfagharifard, E. (2014, 19 marzo). Is this the first wearable computer? 300-year-old Chinese abacus ring was used during the Qing. . . *Mail Online*. https://www.dailymail.co.uk/sciencetech/article-2584437/Is-wearable-computer-300-year-old-Chinese-abacus-ring-used-Qing-Dynasty-help-traders.html

[3] Engineering, I. (2017, 4 marzo). The Smart Ring: From the 17th Century Wearable Abacus to Today. *Interesting Engineering*. https://interestingengineering.com/innovation/smart-ring-17th-century-wearable-abacus-today

[4] Munro, H. (1988, 25 de agosto). *B.C. Inventor of walkie-talkie saluted*. *Vancouver Sun*.

[5] Thorp, E. O. (1998). The invention of the first wearable computer. *Digest of Papers. Second International Symposium on Wearable Computers (Cat. No. 98EX215)*, 4–8

[6] Bhatti, D. S., Saleem, S., Imran, A., Iqbal, Z., Alzahrani, A., Kim, H., & Kim, K. (2022). A Survey on Wireless Wearable Body Area Networks: A Perspective of Technology and Economy. *Sensors*, *22*(20), 7722. https://doi.org/10.3390/s22207722

[7] *IDC Forecasts Continued Growth for Wearables But Growth Will Be Uneven Across Product Categories*. (26 de Septiembre, 2024). IDC: The Premier Global Market Intelligence Company. https://www.idc.com/getdoc.jsp?containerId=prUS52615024.

[8] *Mordor Intelligence. (2024.). Análisis de participación y tamaño del mercado de dispositivos portátiles inteligentes: tendencias y pronósticos de crecimiento (2024-2029). Recuperado de https://www.mordorintelligence.com/es/industry-reports/smart-wearables-market* .

[9] Brophy, K., Davies, S., Olenik, S., Cotur, Y., Ming, D., Van Zalk, N., O'Hare, D., Guder, F., & Yetisen, A. (2021). The future of wearable technologies. *20*. https://doi.org/10.25561/88893.

[10] Marín, E. (2024, 6 de junio). *Apple es líder en el mercado de los wearables y relojes inteligentes, pero el Top 5 tiene una sorpresa*. Xataka Móvil. Recuperado de https://www.xatakamovil.com/mercado/apple-lider-mercado-wearables-relojes-inteligentes-top-5-tiene-sorpresa

[11] Dehghani, M., Abubakar, A. M., & Pashna, M. (2018). Market-driven management of start-ups: The case of wearable technology. *Applied Computing And Informatics*, *18*(1/2), 45-60. https://doi.org/10.1016/j.aci.2018.11.002

[12] Nakhla, J., Kobets, A., De la Garza Ramos, R., Haranhalli, N., Gelfand, Y., Ammar, A., Echt, M., Scoco, A., Kinon, M., & Yassari, R. (2016b). Use of Google Glass to Enhance Surgical Education of Neurosurgery Residents: "Proof-of-Concept" Study. *World Neurosurgery*, *98*, 711-714. https://doi.org/10.1016/j.wneu.2016.11.122

[13] Wikipedia. (s. f.). *Google Glass*. Wikipedia, la enciclopedia libre. Recuperado el 13 de marzo de 2025, de https://es.wikipedia.org/wiki/Google_Glass

[14] Park, S. R., Park, J. Y., Ghani, R., Ha, J., & Hester, T. (2022). Visualising the Future of Orthopaedic Surgery: A Novel Application of Wireless Smart Glasses to Visualise Intraoperative Imaging. *Cureus*. https://doi.org/10.7759/cureus.22004

[15] Van Doormaal, T. P. C., Van Doormaal, J. A. M., & Mensink, T. (2019b). Clinical Accuracy of Holographic Navigation Using Point-Based Registration on Augmented-Reality Glasses. *Operative Neurosurgery*, *17*(6), 588-593. https://doi.org/10.1093/ons/opz094

[16] Microsoft. (s. f.). *HoloLens 2: Ver especificaciones y características*. Recuperado de https://www.microsoft.com/es-es/d/hololens-2/91pnzzznzwcp?activetab=pivot:especificacionestécnicastab

[17] Guk, K., Han, G., Lim, J., Jeong, K., Kang, T., Lim, E., & Jung, J. (2019). Evolution of Wearable Devices with Real-Time Disease Monitoring for Personalized Healthcare. *Nanomaterials*, *9*(6), 813. https://doi.org/10.3390/nano9060813

[18] Kim, J., Kim, N., Kwon, M., & Lee, J. (2017). Attachable Pulse Sensors Integrated with Inorganic Optoelectronic Devices for Monitoring Heart Rates at Various Body Locations. *ACS Applied Materials & Interfaces*, *9*(31), 25700-25705. https://doi.org/10.1021/acsami.7b05264

[19] Anastasova, S., Crewther, B., Bembnowicz, P., Curto, V., Ip, H. M., Rosa, B., & Yang, G. (2016). A wearable multisensing patch for continuous sweat monitoring. *Biosensors And Bioelectronics*, *93*, 139-145. https://doi.org/10.1016/j.bios.2016.09.038

[20] Vashist, S. K. (2012). Non-invasive glucose monitoring technology in diabetes management: A review. *Analytica Chimica Acta*, *750*, 16-27. https://doi.org/10.1016/j.aca.2012.03.043

[21] López-Blanco, R., Velasco, M. A., Méndez-Guerrero, A., Romero, J. P., Del Castillo, M. D., Serrano, J. I., Rocon, E., & Benito-León, J. (2019). Smartwatch for the analysis of rest tremor in patients with Parkinson's disease. *Journal Of The Neurological Sciences*, *401*, 37-42. https://doi.org/10.1016/j.jns.2019.04.011

[22] Palastanga, N., & Soames, R. (2012). Anatomy and human movement: structure and function (6th ed.). Edinburgh: Churchill Livingstone.

[23] Harput, G. (2020). Kinesiology of the knee joint. En *Elsevier eBooks* (pp. 393-410). https://doi.org/10.1016/b978-0-12-812162-7.00022-9.

[24] Lippert, L. S. (2011). Clinical Kinesiology and Anatomy (5th ed.). Philadelphia, PA: F. A. Davis Company.

[25] Neumann, D. A. (2016). *Kinesiology of the Musculoskeletal System : Foundations for Rehabilitation*.

[26] Hall, S. J. (2015). Basic biomechanics (7th ed.). New York, NY: McGraw-Hill Education.

[27] Gilroy, A. M., MacPherson, B. R., Ross, L. M., Schünke, M., Schulte, E., & Schumacher, U. (2021). *Atlas of Anatomy, Latin Nomenclature*.

[28] Pamboris, G. M., Pavlou, K., Paraskevopoulos, E., & Mohagheghi, A. A. (2024). Effect of open vs. closed kinetic chain exercises in ACL rehabilitation on knee joint pain, laxity, extensor muscles strength, and function: a systematic review with meta-analysis. *Frontiers In Sports And Active Living*, *6*. https://doi.org/10.3389/fspor.2024.1416690

[29] Anwer, S., Li, H., Anwar, D., & Wong, A. y. L. (2023). Biomechanical principles of exercise prescription in knee rehabilitation. En *Elsevier eBooks* (pp. 617-631). https://doi.org/10.1016/b978-0-323-90597-8.00029-3

[30] Harput, G. (2020b). Kinesiology of the knee joint. En *Elsevier eBooks* (pp. 393-410). https://doi.org/10.1016/b978-0-12-812162-7.00022-9

[31] Flanigan, D. C., Everhart, J. S., & Glassman, A. H. (2015). Psychological Factors Affecting Rehabilitation and Outcomes Following Elective Orthopaedic Surgery. *Journal Of The American Academy Of Orthopaedic Surgeons*, *23*(9), 563-570. https://doi.org/10.5435/jaaos-d-14-00225

[32] Brander, V. A., Stulberg, S. D., Adams, A. D., Harden, R. N., Bruehl, S., Stanos, S. P., & Houle, T. (2003). Ranawat Award Paper: Predicting Total Knee Replacement Pain. *Clinical Orthopaedics And Related Research*, *416*, 27-36. https://doi.org/10.1097/01.blo.0000092983.12414.e9

[33] Riddle, D. L., Wade, J. B., Jiranek, W. A., & Kong, X. (2009). Preoperative Pain Catastrophizing Predicts Pain Outcome after Knee Arthroplasty. *Clinical Orthopaedics And Related Research*, *468*(3), 798-806. https://doi.org/10.1007/s11999-009-0963-y

[34] Hanusch, B. C., O'Connor, D. B., Ions, P., Scott, A., & Gregg, P. J. (2014). Effects of psychological distress and perceptions of illness on recovery from total knee replacement. *The Bone & Joint Journal*, *96-B*(2), 210-216. https://doi.org/10.1302/0301-620x.96b2.31136

[35] Bricogeek**.** (s.f.). *Arduino Nano*. Bricogeek Lab. Recuperado el 13 de marzo de 2025. https://lab.bricogeek.com/tutorial/guia-de-modelos-arduino-y-sus-caracteristicas/arduino-nano

[36] InvenSense**.** (2013). *MPU-6000 and MPU-6050 product specification* (Rev. 3.4). TDK. Recuperado el 13 de marzo de 2025. https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[37] Oxdea**.** (s.f.). *Módulo de motor de vibración PWM, 5VDC*. Recuperado el 13 de marzo de 2025, de https://oxdea.gt/product/modulo-de-motor-de-vibracion-pwm-5vdc/

[38] JNHuaMao Technology Company. (2013). Recuperado el 13 de marzo de 2025 *HM Bluetooth module datasheet* (Versión V507). Recuperado de https://www.hadex.cz/spec/m432e.pdf

[39] DSD TECH- Connecting the future. (s. f.) *HM-10 Bluetooth Module*. Recuperado el 13 de marzo de 2025. https://www.deshide.com/productdetails.html?pid=1663285&_t=1665209850

[40] Adafruit**.** (s.f.). *Adafruit MMA8451 accelerometer breakout*. Adafruit Learning System. Recuperado el 13 de marzo de 2025, de https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/downloads

[41] Wikipedia. (s.f.). *Plataforma universal de Windows*. Wikipedia, la enciclopedia libre. Recuperado el 15 de marzo de 2025, de https://es.wikipedia.org/wiki/Plataforma_universal_de_Windows

[42] Microsoft. (s.f.). *Universal Windows Platform (UWP) Guide*. Microsoft Learn. Recuperado el 15 de marzo de 2025, de https://learn.microsoft.com/es-es/windows/uwp/get-started/universal-application-platform-guide

[43] Bluetooth SIG. (s.f.). *Bluetooth technology overview*. Bluetooth. Recuperado el 15 de marzo de 2025, de https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[44] Pasternak, J. (2020, 17 de marzo). *¿Cómo funciona Bluetooth Low Energy?* WeLiveSecurity. Recuperado el 15 de marzo de 2025, de https://www.welivesecurity.com/la-es/2020/03/17/como-funciona-bluetooth-low-energy/

[45]Townsend, K., Cufí, C., Akiba, & Davidson, R. (2014). *Getting Started with Bluetooth Low Energy*. http://ci.nii.ac.jp/ncid/BB17000516

[46] Microsoft. (s.f.). *Gatt server*. Microsoft Learn. Recuperado el 15 de marzo de 2025, de https://learn.microsoft.com/en-us/windows/uwp/devices-sensors/gatt-server

[47] Microsoft. (s.f.). *Gatt client*. Microsoft Learn. Recuperado el 15 de marzo de 2025, de https://learn.microsoft.com/en-us/windows/uwp/devices-sensors/gatt-client

[48] Microsoft. (s.f.). *DeviceWatcher class (Windows.Devices.Enumeration)*. Microsoft Learn. Recuperado el 15 de marzo de 2025, de https://learn.microsoft.com/en-us/uwp/api/windows.devices.enumeration.devicewatcher?view=winrt-26100

[49] Cornel Human (2022). *Quick C# application to connect with a Bluetooth LE device [video]. Youtube.* Recuperado el 15 de marzo de 2025. https://www.youtube.com/watch?v=CozmqN_iwNs&t=1953s&ab_channel=CornelHuman

9. Anexos

Código realizado en Arduino IDE.

```
#define vibrador 7
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
#include "SoftwareSerial.h"
#include "Adafruit_MMA8451.h"
#include "Adafruit_Sensor.h"

SoftwareSerial BT(10, 11);
Adafruit_MMA8451 mma = Adafruit_MMA8451();
MPU6050 mpu(0x68);
unsigned long tiempo_prev;
enum State { label0 = 0, label1 = 1, label2 = 2, label3 = 3 };
State var = label0;
int ax, ay, az;
int gx, gy, gz;

int bytesRead = 0;
const int bufferSize9 = 9;
char buffer[bufferSize9];

unsigned long tiempo_muestras;
unsigned long tiempo_vibrador;

int8_t vibrador_x = 99;
int8_t vibrador_y = 100;
int8_t vibrador_z = 100;

float accel_ang_x_6050, accel_ang_y_6050, accel_ang_z_6050;
float accel_ang_x_MMA, accel_ang_y_MMA, accel_ang_z_MMA;


int8_t angle_x_6050 = 0, angle_y_6050 = 0, angle_z_6050 = 0;
float dt;

void setup() {
  tiempo_muestras = millis();
  tiempo_vibrador = millis();
  Serial.begin(9600);
  BT.begin(9600);
  pinMode(vibrador, OUTPUT);
  Wire.begin();

  mpu.initialize();
  mpu.setXAccelOffset(-3599);
```

```
  mpu.setYAccelOffset(1057);
  mpu.setZAccelOffset(1404);
}
void loop() {
  switch(var)
  {
    case label0:
      Vibracion_bluetooth();
      digitalWrite(vibrador, LOW);
      break;
    case label1:
      Vibracion_bluetooth();
      MPU_Tilt();
      BT.print(accel_ang_x_6050);
      BT.print("M");
      BT.print(accel_ang_y_6050);
      BT.print("M");
      BT.print(accel_ang_z_6050);
      delay(100);
      break;
    case label2:
      if (millis() - tiempo_muestras >= 150)
      {
        Vibracion_bluetooth();
        MPU_Tilt();
        BT.print(accel_ang_x_6050);
        BT.print("M");
        BT.print(accel_ang_y_6050);
        BT.print("M");
        BT.print(accel_ang_z_6050);

        if (millis() - tiempo_vibrador >= 2000)
        {
          tiempo_vibrador = millis();
          if (vibrador_y < accel_ang_y_6050)
            {
              float valor_x = 90;
              int accel_ang_y_6050_ant = -50;
              int accel_ang_z_6050_ant = -50;
              delay(10);
              BT.print(accel_ang_y_6050_ant);
              BT.print("M");
              BT.print(valor_x);
              BT.print("M");
              BT.print(accel_ang_z_6050_ant);
              digitalWrite(vibrador, HIGH);
            }
         }
        else
```

```
        {
          digitalWrite(vibrador, LOW);
        }
        tiempo_muestras = millis();
      }
      break;
    case label3:
      Vibracion_bluetooth();
      ADA_Tilt();
      MPU_Tilt();
      float diff_x = accel_ang_x_6050 - accel_ang_x_MMA;
      float diff_y = accel_ang_y_6050 - accel_ang_y_MMA;
      float diff_z = accel_ang_z_6050 - accel_ang_z_MMA;
      BT.print(diff_x);
      BT.print("M");
      BT.print(diff_y);
      BT.print("M");
      BT.print(diff_z);
      delay(100);
    break;
  }
}
void Vibracion_bluetooth()
{
  while (BT.available())
    {
      buffer[bytesRead++] = BT.read();
    }
    if (bytesRead == 8 )
    {
      processData(buffer, bytesRead);
      bytesRead = 0;
      memset(buffer, 0, sizeof(buffer));
    }
    else if (bytesRead == 1)
    {
      int commandValue = (int)strtol(buffer, NULL, 16);
      var = (State)commandValue;
      bytesRead = 0;
      memset(buffer, 0, sizeof(buffer));
    }
}

void processData(char* data, int length)
{
  if (length == 8)
  {
    char tempX[3] = {data[0], data[1], '\0'};
    char tempY[3] = {data[3], data[4], '\0'};
```

```
    char tempZ[3] = {data[6], data[7], '\0'};

    vibrador_x = atof(tempX);
    vibrador_y = atof(tempY);
    vibrador_z = atof(tempZ);
  }
}

void MPU_Tilt()
{
  mpu.getAcceleration(&ax, &ay, &az);
  mpu.getRotation(&gx, &gy, &gz);
  dt = (millis()-tiempo_prev)/1000.0;
  tiempo_prev=millis();
  float gyro_rate_x = gx / 131.0;
  float gyro_rate_y = gy / 131.0;
  float gyro_rate_z = gz / 131.0;
  accel_ang_x_6050 = atan(ax / sqrt(pow(ay, 2) + pow(az, 2))) * RAD_TO_DEG;
  accel_ang_y_6050 = atan(ay / sqrt(pow(ax, 2) + pow(az, 2))) * RAD_TO_DEG;
   angle_x_6050  =  0.98  *  (accel_ang_x_6050  +  gyro_rate_x  *  dt)  +  0.02  *
accel_ang_x_6050;
   angle_y_6050  =  0.98  *  (accel_ang_y_6050  +  gyro_rate_y  *  dt)  +  0.02  *
accel_ang_y_6050;
  angle_z_6050 += gyro_rate_z * dt;
  accel_ang_x_6050 = angle_x_6050;
  accel_ang_y_6050 = angle_y_6050;
  accel_ang_z_6050 = angle_z_6050;
}

void ADA_Tilt()
{
  mma.read();
  accel_ang_x_MMA = atan(mma.x / sqrt(pow(mma.y, 2) + pow(mma.z, 2))) * RAD_TO_DEG;
  accel_ang_y_MMA = atan(mma.y / sqrt(pow(mma.x, 2) + pow(mma.z, 2))) * RAD_TO_DEG;
  accel_ang_z_MMA = asin(mma.z / sqrt(pow(mma.x, 2) + pow(mma.y, 2) + pow(mma.z, 2)))
* RAD_TO_DEG;
}
```

La programación de Visual Studio UWP

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Devices.Bluetooth;
using Windows.Devices.Enumeration;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
```

```
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using Windows.UI.Popups;
using System.Diagnostics;
using Windows.Devices.Bluetooth.GenericAttributeProfile;
using System.Reflection.PortableExecutable;
using Windows.Storage.Streams;
using System.Data;
using System.Text;
using Windows.ApplicationModel.Background;
using System.Globalization;
using Windows.Graphics.Display;
using Microsoft.Graphics.Canvas.UI.Xaml;
using Microsoft.Graphics.Canvas.Geometry;
using System.Numerics;
using Windows.UI;
using Microsoft.Graphics.Canvas;
using System.Threading;
using Microsoft.Graphics.Canvas.Text;
using Windows.UI.Xaml.Media.Animation;
using System.Collections;
using System.Threading.Tasks;
using Windows.Security.Cryptography.Core;
using Microsoft.Graphics.Canvas.Brushes;
using System.Drawing;
using Windows.UI.Core;
using System.Collections.ObjectModel;
using static System.Net.WebRequestMethods;

namespace App6
{
    public sealed partial class MainPage : Page
    {
        public                  ObservableCollection<DeviceInformationWrapper>
DispositivosAnadidos2     {     get;     private     set;     }     =     new
ObservableCollection<DeviceInformationWrapper>();


        public static string DSD_TECH_UUID = "ffe0";
        public GattCharacteristic _caracteristica;
        public BluetoothLEDevice bluetoothLeDevice;
        public DeviceWatcher deviceWatcher;
        private List<float> x_medidas = Enumerable.Repeat(0.0f, 21).ToList();
        private List<float> y_medidas = Enumerable.Repeat(0.0f, 21).ToList();
        private List<float> z_medidas = Enumerable.Repeat(0.0f, 21).ToList();

        private List<float> vision_globalX = new List<float>();
        private List<float> vision_globalY = new List<float>();
        private List<float> vision_globalZ = new List<float>();

        bool auxiliar = false;


        public MainPage()
        {
            this.InitializeComponent();
            InitializeDeviceWatcher();
            this.DataContext = this;

            TxTX.Visibility = Visibility.Collapsed;
            TxTY.Visibility = Visibility.Collapsed;
            TxTZ.Visibility = Visibility.Collapsed;
```

```
            canvas1.Visibility = Visibility.Collapsed;
            canvas2.Visibility = Visibility.Collapsed;
            canvas3.Visibility = Visibility.Collapsed;
            canvas4.Visibility = Visibility.Collapsed;

            Angulos_.Visibility = Visibility.Collapsed;
            Modulo_Vibración.Visibility = Visibility.Collapsed;

            AnguloX.Visibility = Visibility.Collapsed;
            AnguloY.Visibility = Visibility.Collapsed;
            AnguloZ.Visibility = Visibility.Collapsed;

            Envio_Dato.Visibility = Visibility.Collapsed;

            RomperEnvio.Visibility = Visibility.Collapsed;

            Global.Visibility = Visibility.Collapsed;

            Fusion_acelerometros.Visibility = Visibility.Collapsed;



        }

        void InitializeDeviceWatcher()
        {
            string[]             requestedProperties             =             {
"System.Devices.Aep.DeviceAddress", "System.Devices.Aep.IsConnected" };
            DeviceWatcher deviceWatcher =
                DeviceInformation.CreateWatcher(

BluetoothLEDevice.GetDeviceSelectorFromPairingState(false),
                    requestedProperties,
                    DeviceInformationKind.AssociationEndpoint);
            deviceWatcher.Added += DeviceWatcher_Added;
            deviceWatcher.Removed += DeviceWatcher_Remove;
            deviceWatcher.Start();
        }


        static bool IsWatcherStarted(DeviceWatcher watcher)
        {
            return (watcher.Status == DeviceWatcherStatus.Started) ||
                (watcher.Status == DeviceWatcherStatus.EnumerationCompleted);
        }

        public bool IsWatcherRunning()
        {
            if (deviceWatcher == null)
            {
                return false;
            }

            DeviceWatcherStatus status = deviceWatcher.Status;
            return (status == DeviceWatcherStatus.Started) ||
                (status == DeviceWatcherStatus.EnumerationCompleted) ||
                (status == DeviceWatcherStatus.Stopping);
        }

        private void StopDeviceWatcher()
        {
            if (deviceWatcher != null)
            {
                deviceWatcher.Stop();
                deviceWatcher.Added -= DeviceWatcher_Added;
```

```
                deviceWatcher.Removed -= DeviceWatcher_Remove;

                deviceWatcher = null;
            }
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            base.OnNavigatedFrom(e);
            StopDeviceWatcher();
        }

        private    async    void    DeviceWatcher_Added(DeviceWatcher    sender,
DeviceInformation args)
        {
            await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
            {
                if (IsWatcherStarted(sender))
                {
                    DispositivosAnadidos2.Add(new
DeviceInformationWrapper(args));

                }
            });

        }
private     async     void     DeviceWatcher_Remove(DeviceWatcher     sender,
DeviceInformationUpdate args)
 {
     {
         await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
         {
             if (IsWatcherStarted(sender))
             {
                 var deviceToRemove = DispositivosAnadidos2.FirstOrDefault(d =>
d.Id == args.Id);
                 if (deviceToRemove != null)
                 {
                     DispositivosAnadidos2.Remove(deviceToRemove);

                 }
             }
         });
     }
 }


 public class DeviceInformationWrapper
 {
     public DeviceInformation DeviceInformation { get; set; }

     public string DisplayName
     {
         get
         {
             return       string.IsNullOrEmpty(DeviceInformation.Name)       ?
DeviceInformation.Id : DeviceInformation.Name;
         }
     }

     public string Id
     {
         get
         {
             return DeviceInformation.Id;
         }
```

```
    }

    public DeviceInformationWrapper(DeviceInformation deviceInformation)
    {
        DeviceInformation = deviceInformation;
    }
}
```

```
async void Conexion_Click(object sender, RoutedEventArgs e)
{
    var seleccionado = DispositivoComboBox.SelectedItem as
DeviceInformationWrapper;
    if (seleccionado == null || seleccionado.DisplayName != "DSD TECH" )
    {
        var mostrarmensaje = new MessageDialog("No Encontrado");
        await mostrarmensaje.ShowAsync();
    }

    else
    {
        var mostrarmensaje = new MessageDialog("¡Dispositivo Encontrado!");
        StopDeviceWatcher();

        await mostrarmensaje.ShowAsync();
        var DSD_TECH = DispositivosAnadidos2.FirstOrDefault(dispositivo =>
dispositivo.DisplayName == "DSD TECH");
        BluetoothLEDevice bluetoothLeDevice = await
BluetoothLEDevice.FromIdAsync(DSD_TECH.Id);
        //deviceWatcher.Stop();
        GattDeviceServicesResult result = await
bluetoothLeDevice.GetGattServicesAsync();

        if (result.Status == GattCommunicationStatus.Success)
        {
            var services = result.Services;
            Debug.WriteLine(services);
            foreach (var servicios in services)
            {
                Debug.WriteLine(servicios.Uuid);
            }


            foreach (var service in services)
                if (service.Uuid.ToString("N").Substring(4, 4) ==
DSD_TECH_UUID)
                {
                    Debug.WriteLine("Se ha encontrado el servicio");

                    GattCharacteristicsResult
caracteristica_servicio_resultado = await service.GetCharacteristicsAsync();
                    foreach (var caracteristicas in
caracteristica_servicio_resultado.Characteristics)
                    {
                        Debug.WriteLine(caracteristicas.Uuid);
                    }



                    Debug.WriteLine(caracteristica_servicio_resultado);
```

```
                    if      (caracteristica_servicio_resultado.Status      ==
GattCommunicationStatus.Success)
                    {
                        var             caracteristica_servicio             =
caracteristica_servicio_resultado.Characteristics;
                        foreach         (var         caracteristica         in
caracteristica_servicio)
                        {
                            _caracteristica = caracteristica;
                            GattCharacteristicProperties     properties     =
_caracteristica.CharacteristicProperties;

                            Debug.WriteLine(properties);

                            if
(properties.HasFlag(GattCharacteristicProperties.Read))
                            {
                                GattCommunicationStatus    status    =    await
_caracteristica.WriteClientCharacteristicConfigurationDescriptorAsync(

GattClientCharacteristicConfigurationDescriptorValue.Notify);

                                if              (result.Status              ==
GattCommunicationStatus.Success)
                                {
                                    DispositivoComboBox.Visibility          =
Visibility.Collapsed;
                                    BotonConexion.Visibility                =
Visibility.Collapsed;


                                    Angulos_.Visibility = Visibility.Visible;
                                    Modulo_Vibración.Visibility             =
Visibility.Visible;
                                    Global.Visibility = Visibility.Visible;

                                    Fusion_acelerometros.Visibility         =
Visibility.Visible;

                                }



                            }
                        }



                    }

                }




            }

        }



    }


    async public void Angulos(object sender, RoutedEventArgs e)
    {
```

```
     byte[] bytes = Encoding.ASCII.GetBytes("1");


     var writer = new DataWriter();
     writer.WriteBytes(bytes);
     await Task.Delay(30);

     GattCommunicationStatus             result             =             await
_caracteristica.WriteValueAsync(writer.DetachBuffer());
     TxTX.Visibility = Visibility.Visible;
     TxTY.Visibility = Visibility.Visible;
     TxTZ.Visibility = Visibility.Visible;

     canvas1.Visibility = Visibility.Visible;
     canvas2.Visibility = Visibility.Visible;
     canvas3.Visibility = Visibility.Visible;

     RomperEnvio.Visibility = Visibility.Visible;

     Angulos_.Visibility = Visibility.Collapsed;
     Modulo_Vibración.Visibility = Visibility.Collapsed;
     Global.Visibility = Visibility.Collapsed;

     Fusion_acelerometros.Visibility = Visibility.Collapsed;

     _caracteristica.ValueChanged += Lectura_Datos;

 }

 async public void ModuloVibracion(object sender, RoutedEventArgs e)
 {
     byte[] bytes = Encoding.ASCII.GetBytes("2");


     var writer = new DataWriter();
     writer.WriteBytes(bytes);
     await Task.Delay(30);


     GattCommunicationStatus             result             =             await
_caracteristica.WriteValueAsync(writer.DetachBuffer());

     Angulos_.Visibility = Visibility.Collapsed;
     Modulo_Vibración.Visibility = Visibility.Collapsed;
     Global.Visibility = Visibility.Collapsed;

     Fusion_acelerometros.Visibility = Visibility.Collapsed;

     AnguloX.Visibility = Visibility.Visible;
     AnguloY.Visibility = Visibility.Visible;
     AnguloZ.Visibility = Visibility.Visible;

     Envio_Dato.Visibility = Visibility.Visible;
 }

 async public void EnviarClick(object sender, RoutedEventArgs e)
 {

     Codigo();
     await Task.Delay(30);

     AnguloX.Visibility = Visibility.Collapsed;
     AnguloY.Visibility = Visibility.Collapsed;
     AnguloZ.Visibility = Visibility.Collapsed;

     TxTX.Visibility = Visibility.Visible;
```

```
     TxTY.Visibility = Visibility.Visible;
     TxTZ.Visibility = Visibility.Visible;

     canvas1.Visibility = Visibility.Visible;
     canvas2.Visibility = Visibility.Visible;
     canvas3.Visibility = Visibility.Visible;

     RomperEnvio.Visibility = Visibility.Visible;

     _caracteristica.ValueChanged += Lectura_Datos;


 }

 async public void GlobalCLick(object sender, RoutedEventArgs e)
{
     byte[] bytes = Encoding.ASCII.GetBytes("1");


     var writer = new DataWriter();
     writer.WriteBytes(bytes);
     await Task.Delay(30);

     GattCommunicationStatus             result             =             await
_caracteristica.WriteValueAsync(writer.DetachBuffer());

     auxiliar = true;
     Angulos_.Visibility = Visibility.Collapsed;
     Modulo_Vibración.Visibility = Visibility.Collapsed;
     RomperEnvio.Visibility = Visibility.Visible;

     Global.Visibility = Visibility.Collapsed;
     _caracteristica.ValueChanged += Lectura_Datos;


     await Task.Delay(10000);

     canvas4.Visibility = Visibility.Visible;


 }

 async public void Fusion(object sender, RoutedEventArgs e)
 {

     byte[] bytes = Encoding.ASCII.GetBytes("3");


     var writer = new DataWriter();
     writer.WriteBytes(bytes);
     await Task.Delay(30);


     GattCommunicationStatus             result             =             await
_caracteristica.WriteValueAsync(writer.DetachBuffer());

     Angulos_.Visibility = Visibility.Collapsed;
     Modulo_Vibración.Visibility = Visibility.Collapsed;
     Global.Visibility = Visibility.Collapsed;
     Fusion_acelerometros.Visibility = Visibility.Collapsed;

     TxTX.Visibility = Visibility.Visible;
     TxTY.Visibility = Visibility.Visible;
     TxTZ.Visibility = Visibility.Visible;

     canvas1.Visibility = Visibility.Visible;
```

```
     canvas2.Visibility = Visibility.Visible;
     canvas3.Visibility = Visibility.Visible;

     RomperEnvio.Visibility = Visibility.Visible;

     _caracteristica.ValueChanged += Lectura_Datos;



 }

        async public void RomperEnvio_(object sender, RoutedEventArgs e)
        {
            _caracteristica.ValueChanged -= Lectura_Datos;

            byte[] bytes0 = Encoding.ASCII.GetBytes("0");


            var writer0 = new DataWriter();
            writer0.WriteBytes(bytes0);
            GattCommunicationStatus          result0          =          await
_caracteristica.WriteValueAsync(writer0.DetachBuffer());

            Angulos_.Visibility = Visibility.Visible;
            Modulo_Vibración.Visibility = Visibility.Visible;
            Global.Visibility = Visibility.Visible;
            Fusion_acelerometros.Visibility = Visibility.Visible;

            RomperEnvio.Visibility = Visibility.Collapsed;
            Envio_Dato.Visibility = Visibility.Collapsed;

            TxTX.Visibility = Visibility.Collapsed;
            TxTY.Visibility = Visibility.Collapsed;
            TxTZ.Visibility = Visibility.Collapsed;

            canvas1.Visibility = Visibility.Collapsed;
            canvas2.Visibility = Visibility.Collapsed;
            canvas3.Visibility = Visibility.Collapsed;
            canvas4.Visibility = Visibility.Collapsed;
        }

        async public void Codigo()
        {
                string  listaConcatenada  =  string.Join("  ",  new  string[]  {
AnguloX.Text, AnguloY.Text, AnguloZ.Text });

                byte[] bytes = Encoding.ASCII.GetBytes(listaConcatenada);


                var writer = new DataWriter();
                writer.WriteBytes(bytes);

                GattCommunicationStatus         result         =         await
_caracteristica.WriteValueAsync(writer.DetachBuffer());

        }


        private       void       Lectura_Datos(GattCharacteristic       sender,
GattValueChangedEventArgs args)
        {

            var reader = DataReader.FromBuffer(args.CharacteristicValue);

            byte[] input = new byte[reader.UnconsumedBufferLength];
```

```
            reader.ReadBytes(input);
            List<int> indices = input
                                .Select((value, index) => new { value, index })
                                .Where(x => x.value == 77)
                                .Select(x => x.index)
                                .ToList();

            string x_ASCII = Encoding.ASCII.GetString(input, 0, indices[0]);
            float.TryParse(x_ASCII,                         NumberStyles.Float,
CultureInfo.InvariantCulture, out float x_medida);

            string y_ASCII = Encoding.ASCII.GetString(input, indices[0] + 1,
indices[1] - indices[0] - 1);
            float.TryParse(y_ASCII,                         NumberStyles.Float,
CultureInfo.InvariantCulture, out float y_medida);

            string z_ASCII = Encoding.ASCII.GetString(input, indices[1] + 1,
input.Length - (indices[1] + 1));
                float.TryParse(z_ASCII,                     NumberStyles.Float,
CultureInfo.InvariantCulture, out float z_medida);

            x_medidas.RemoveAt(0);
            x_medidas.Insert(x_medidas.Count, x_medida);

            y_medidas.RemoveAt(0);
            y_medidas.Insert(y_medidas.Count, y_medida);

            z_medidas.RemoveAt(0);
            z_medidas.Insert(z_medidas.Count, z_medida);

            canvas1.Invalidate();
            canvas2.Invalidate();
            canvas3.Invalidate();

            vision_globalX.Add(x_medida);
            vision_globalY.Add(y_medida);
            vision_globalZ.Add(z_medida);


        }



        private      void      canvasControl_Draw1(CanvasControl      sender,
CanvasDrawEventArgs args)
        {


            Grafica_Dibujo(canvas1, args, x_medidas);
            Creacion_Ejes(canvas1, args);


        }
        private      void      canvasControl_Draw2(CanvasControl      sender,
CanvasDrawEventArgs args)
        {


                Grafica_Dibujo(canvas2, args, y_medidas);
                Creacion_Ejes(canvas2, args);

        }
        private      void      canvasControl_Draw3(CanvasControl      sender,
CanvasDrawEventArgs args)
        {
```

```
                Grafica_Dibujo(canvas3, args, z_medidas);
                Creacion_Ejes(canvas3, args);


        }
        private      void      canvasControl_Draw4(CanvasControl      sender,
CanvasDrawEventArgs args)
        {
            if (auxiliar == true)
            {


                Grafica_Dibujo_Global(canvas4,      args,      vision_globalX,
vision_globalY, vision_globalZ);
                Creacion_Ejes(canvas4, args);
            }


        }

        private      void      Grafica_Dibujo_Global(CanvasControl      canvas,
CanvasDrawEventArgs  args,  List<float>  data1,  List<float>  data2,  List<float>
data3)
        {
            var midWidth = (float)(canvas.ActualWidth * 0.5);
            var midHeight = (float)(canvas.ActualHeight * 0.5);

            float intervaloX = (float)(canvas.ActualWidth - 73) / 17;
            float intervaloY = midHeight / 92;

            using (var cpb1 = new CanvasPathBuilder(args.DrawingSession))
            using (var cpb2 = new CanvasPathBuilder(args.DrawingSession))
            using (var cpb3 = new CanvasPathBuilder(args.DrawingSession))
            {
                cpb1.BeginFigure(new Vector2(50, midHeight));
                cpb2.BeginFigure(new Vector2(50, midHeight));
                cpb3.BeginFigure(new Vector2(50, midHeight));

                for (int i = 0; i < data1.Count; i++)
                {
                    float x = 50 + i * intervaloX;
                    float y1 = midHeight - (data1[i] * intervaloY);
                    float y2 = midHeight - (data2[i] * intervaloY);
                    float y3 = midHeight - (data3[i] * intervaloY);

                    cpb1.AddLine(new Vector2(x, y1));
                    cpb2.AddLine(new Vector2(x, y2));
                    cpb3.AddLine(new Vector2(x, y3));
                }
                cpb1.EndFigure(CanvasFigureLoop.Open);
                cpb2.EndFigure(CanvasFigureLoop.Open);
                cpb3.EndFigure(CanvasFigureLoop.Open);


args.DrawingSession.DrawGeometry(CanvasGeometry.CreatePath(cpb1),
Colors.Black);

args.DrawingSession.DrawGeometry(CanvasGeometry.CreatePath(cpb2),
Colors.DarkRed);

args.DrawingSession.DrawGeometry(CanvasGeometry.CreatePath(cpb3),
Colors.Blue);
            }
```

```
        }



        private void Grafica_Dibujo(CanvasControl canvas, CanvasDrawEventArgs
args, List<float> data)
        {
            using (var cpb = new CanvasPathBuilder(args.DrawingSession))
            {
                var midWidth = (float)(canvas.ActualWidth * .5);
                var midHeight = (float)(canvas.ActualHeight * .5);

                cpb.BeginFigure(new Vector2(50, midHeight));

                float intervaloX = (float)(canvas.ActualWidth - 73) / 17;
                float intervaloY = midHeight / 92;

                for (int i = 0; i < data.Count; i++)
                {
                    float x = 50 + i * intervaloX ;
                    float y = midHeight - (data[i] * intervaloY);
                    cpb.AddLine(new Vector2(x, y));
                }

                cpb.EndFigure(CanvasFigureLoop.Open);

args.DrawingSession.DrawGeometry(CanvasGeometry.CreatePath(cpb),
Colors.Black);

            }

        }


        public  void  Creacion_Ejes(CanvasControl  canvas,  CanvasDrawEventArgs
args)
        {
            var width = (float)canvas.ActualWidth;
            var height = (float)(canvas.ActualHeight);
            var midWidth = (float)(width * .5);
            var midHeight = (float)(height * .5);

            using (var cpb = new CanvasPathBuilder(args.DrawingSession))
            {

                cpb.BeginFigure(new Vector2(0, midHeight));
                cpb.AddLine(new Vector2(width, midHeight));
                cpb.EndFigure(CanvasFigureLoop.Open);


args.DrawingSession.DrawGeometry(CanvasGeometry.CreatePath(cpb),
Colors.Black);
            }


            using (var cpb = new CanvasPathBuilder(args.DrawingSession))
            {
                // Vertical line
                cpb.BeginFigure(new Vector2(50, 0));
                cpb.AddLine(new Vector2(50, height));
                cpb.EndFigure(CanvasFigureLoop.Open);


args.DrawingSession.DrawGeometry(CanvasGeometry.CreatePath(cpb),  Colors.Black,
1);
```

```
            }

            args.DrawingSession.DrawText("-90°", 20, height - 20, Colors.Gray,
new CanvasTextFormat()
            {
                FontSize = 15
            });

            args.DrawingSession.DrawText("90°", 25, height - 330, Colors.Gray,
new CanvasTextFormat()
            {
                FontSize = 15
            });

        }
    }

}
```

El siguiendo código corresponde a XAML

```
<Page
    x:Class="App6.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:App6"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:canvas="using:Microsoft.Graphics.Canvas.UI.Xaml"
    mc:Ignorable="d"
    >

    <Grid>
        <!---Creacion del Primer Bloque de la Interfaz-->
        <ComboBox       x:Name="DispositivoComboBox"       ItemsSource="{x:Bind
DispositivosAnadidos2}" DisplayMemberPath="DisplayName"
                  SelectedValuePath="Id"
                  HorizontalAlignment="Center"
                  VerticalAlignment="Top"
                  Width="300"
                  Margin="20"/>


        <Button          x:Name     =     "BotonConexion"     Content="Conexion"
Click="Conexion_Click"                              HorizontalAlignment="Center"
VerticalAlignment="Center"/>

        <!--- Segundo Bloque de la Interfaz Cuando hay conexion-->
        <Button  x:Name  =  "Angulos_"  Content  =  "Angulos"  Click  ="Angulos"
HorizontalAlignment="Center" VerticalAlignment="Center" Width="200" Height="50"
Margin="0,0,0,250" />
        <TextBlock      x:Name="AnguloX_string"      HorizontalAlignment="Left"
VerticalAlignment="Center"    Width="200"    Height="300"    TextWrapping="Wrap"
Foreground="Blue" />
        <TextBlock      x:Name="AnguloY_string"      HorizontalAlignment="Left"
VerticalAlignment="Center"    Width="200"    Height="300"    TextWrapping="Wrap"
Foreground="Blue"/>
        <TextBlock      x:Name="AnguloZ_string"      HorizontalAlignment="Left"
VerticalAlignment="Center"    Width="200"    Height="300"    TextWrapping="Wrap"
Foreground="Blue" />


        <!---Llamamiento a Modulo Vibracion -->
```

```
        <Button  x:Name  =  "Modulo_Vibración"    Content  =  "Ejercicio"  Click
="ModuloVibracion"    HorizontalAlignment="Center"    VerticalAlignment="Center"
Width="200" Height="50" Margin="0,0,0,0"/>
        <TextBox     x:Name     =     "AnguloX"     HorizontalAlignment="Center"
VerticalAlignment="Center"   Width="300"   Height="50"   Margin   =   "0,0,0,300"
PlaceholderText ="Limite X"/>
        <TextBox     x:Name     =     "AnguloY"     HorizontalAlignment="Center"
VerticalAlignment="Center"   Width="300"   Height="50"   Margin   =   "0,0,0,0"
PlaceholderText ="Limite Y"/>
        <TextBox     x:Name     =     "AnguloZ"     HorizontalAlignment="Center"
VerticalAlignment="Center"   Width="300"   Height="50"   Margin   =   "0,300,0,0"
PlaceholderText ="Limite Z"/>
        <Button    x:Name="Envio_Dato"    Content="Envio"    Click="EnviarClick"
HorizontalAlignment="Center" VerticalAlignment="Bottom"/>


        <!---Boton de Vision Global-->
        <Button  x:Name="Global"  Content="Vision  Global"  Click="GlobalCLick"
HorizontalAlignment="Center" VerticalAlignment="center" Width="200" Height="50"
Margin="0,250,0,0"/>
        <canvas:CanvasControl    x:Name="canvas4"    Draw="canvasControl_Draw4"
Height= "800" Width= "1800"
                HorizontalAlignment="Center"  VerticalAlignment="center" />

        <!---Fusión-->
        <Button     x:Name="Fusion_acelerometros"     Content="Cadera-Rodilla"
Click="Fusion"     HorizontalAlignment="Center"     VerticalAlignment="center"
Width="200" Height="50" Margin="0,500,0,0"/>

        <!---Boton de Apagado-->
        <Button x:Name="RomperEnvio" Content="RomperEnvio" Click="RomperEnvio_"
HorizontalAlignment="Right" VerticalAlignment="Center" Margin = "0,250,40,0"/>


        <!--- Creacion de Gráficas -->
        <Grid Margin="0,0,0,700">
            <TextBlock     x:Name     ="TxTX"     Text="Angulo     Eje     X"
HorizontalAlignment="Left" VerticalAlignment="Top"
                    Margin="500,0,40,0"/>
            <canvas:CanvasControl  x:Name="canvas1"  Draw="canvasControl_Draw1"
Height= "330" Width= "700"
                    HorizontalAlignment="Center"   />
        </Grid>

        <Grid Margin="0,0,0,30">
            <TextBlock     x:Name     ="TxTY"     Text="Angulo     Eje     Y"
HorizontalAlignment="Left" VerticalAlignment="Bottom"
                   Margin="500,0,0,630"/>
            <canvas:CanvasControl  x:Name="canvas2"  Draw="canvasControl_Draw2"
Height= "330" Width= "700"
                    HorizontalAlignment="Center"
VerticalAlignment="Center" />
        </Grid>


        <Grid Margin="0,0,0,10">
            <TextBlock     x:Name     ="TxTZ"     Text="Angulo     Eje     Z"
HorizontalAlignment="Left" VerticalAlignment="Bottom"
                    Margin="500,0,0,300"/>
            <canvas:CanvasControl  x:Name="canvas3"  Draw="canvasControl_Draw3"
Height= "330" Width= "700"
                    HorizontalAlignment="Center"
VerticalAlignment="Bottom" />

        </Grid>
```

```
    </Grid>
</Page>
```

.