Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Sensor de movimiento de la cabeza para niños con dificultades motoras

Julio 2023

Autor: Eduardo Castro Pozuelo

Tutor: Jonay Tomás Toledo Carrillo

Agradecimientos

Quiero expresar mi agradecimiento a mi tutor de TFG Jonay Toledo, por la facilitación de los conocimientos y medios técnicos para la realización de este proyecto.

A todos mis profesores que me han impartido clases y me han permitido obtener el Grado en Ingeniería Electrónica Industrial y Automática.

A mis padres, por su ayuda y apoyo incondicional durante estos cuatro años de mi vida.

A mi hermano, que siempre ha estado a mi lado.

A mis abuelos/as, mis tíos/as y primos/as, de los que siempre he recibido ánimo y fuerza para seguir adelante.

Gracias a todos/as por ayudarme a conseguir uno de los objetivos de mi vida: mi formación académica.

INDICE

Resumen

Los ingenieros electrónicos aplican los principios de la electrónica y la física para crear dispositivos y sistemas que utilizamos en nuestra vida cotidiana. Dichos principios pueden aplicarse para el diseño, análisis, desarrollo manejo, implementación y mantenimiento de aplicaciones y sistemas electrónicos. Algunas de las áreas en las que la electrónica está vinculada son: telecomunicaciones, instrumentación electrónica, microcontroladores y microprocesadores.

En este proyecto, la principal motivación ha sido desarrollar un prototipo que tenga una aplicación real y práctica en la vida diaria de las personas y que sirva para mejorar las funciones cotidianas de los niños con dificultad motora mediante el diseño de un sensor que sea capaz de detectar los movimientos de su cabeza en sus tres ejes x,y,z. Los valores así obtenidos podrían tener una aplicación posterior para medir la dificultad motora del niño y sobre todo para ir mirando cómo va mejorando si se le aplica algún tipo de terapia.

Este proyecto se ha llevado a cabo en sucesivas fases: estudio previo de los dispositivos electrónicos disponibles en el mercado, diseño del sistema, montaje, programación e implementación del conjunto y, finalmente, comprobación de su correcto funcionamiento, obteniendo como resultado un producto fiable y de bajo coste.

Con este proyecto también se está cumpliendo con uno de los objetivos de desarrollo sostenible aprobados a nivel mundial en la Agenda 2030: mejorar la vida de las personas y garantizar a todos los niños y niñas una educación inclusiva y de calidad.

Abstract

Electronic engineers apply the principles of electronics and physics to create devices and systems that we use in our daily lives..These principles can be applied to the design, analysis, development, management, implementation and maintenance of applications and electronic systems. Some of the areas in which electronics is linked are: telecommunications, electronic instrumentation, microcontrollers and microprocessors.

In this project, the main motivation has been to develop a prototype that has a real and practical application in the daily life of people and that serves to improve the daily functions of children with motor difficulties by designing a sensor that is capable of detect the movements of its head in its three axes x, y, z. The values obtained in this way could have a later application to measure the motor difficulty of the child and above all to see how he/she improves if some type of therapy is applied.

This project has been carried out in successive phases: preliminary study of the electronic devices available on the market, system design, assembly, programming and implementation of the set and, finally, verification of its correct operation, obtaining as a result a reliable and low cost product.

This project is also meeting one of the sustainable development goals approved worldwide in the 2030 Agenda: to improve people's lives and guarantee all children an inclusive and quality education.

1. Introducción

1.1 Justificación de la necesidad

Según la Organización Mundial de la Salud (OMS), la discapacidad motora es «la secuela o malformación que deriva de una afección en el sistema neuromuscular a nivel central o periférico, dando como resultado alteraciones en el control del movimiento y la postura». Por tanto, implica algún tipo de dificultad para llevar a cabo actividades de la vida cotidiana: dificultades para el desplazamiento, para manipular objetos, para acceder a determinados espacios o en el habla.[1]

Según publica en su página web la entidad *UNIR* (LA UNIVERSIDAD EN INTERNET), Los tipos de discapacidad motora en niños más frecuentes son [2]:

- Parálisis cerebral: alteraciones del tono muscular, la postura y la movilidad en niño, debido a una lesión encefálica en la etapa prenatal o durante la infancia.
- Espina bífida: anomalía congénita de la columna vertebral que provoca una parálisis por debajo de la lesión, falta de sensibilidad y/o una malformación en algún miembro inferior.
- Miopatía o distrofia muscular progresiva: pérdida de fuerza por una degeneración muscular progresiva a lo largo de los primeros años de vida del niño.
- Ataxia: dificultades del niño para mantener el equilibrio, la postura y el control del movimiento voluntario por una disfunción, principalmente, en el cerebelo.
- Derivados de traumatismos craneoencefálicos, que derivan en problemas motores, emocionales o dificultades en el lenguaje y la memoria.

En el blog del *Instituto Superior de Estudios Sociales y Sociosanitarios* (ISES) *de Valencia*, aparece publicado un artículo sobre la vida escolar del niño con dificultad motora. Según dicho artículo, un niño con discapacidad motora presenta unas características comunes, como son las dificultades para la motricidad fina y gruesa, las limitaciones en el conocimiento del medio que les rodea y en las posibilidades de actuación sobre el entorno y la imposibilidad o dificultad para la comunicación oral. [3]

Según publica en su página web la institución *Formainfancia Euopean School*, escuela de formación líder y pionera en programas formativos de Educación ,Psicopedagogía y Salud Infantil, los principales problemas que puede generar la discapacidad motriz son: movimientos exactos incontrolados, dificultad de coordinación, alcance limitado, fuerza reducida, habla no inteligible, dificultad con la motricidad fina y gruesa y mala accesibilidad al medio físico. Los niños con esta afección y sus familias necesitan apoyo. La dificultad para moverse, de interactuar con el entorno, va a afectar al desarrollo de estos niños en todos los niveles: sociales, cognitivos y emocionales. Muchos niños con discapacidad sufren baja autoestima. No se sienten competentes y esto afecta a sus aprendizajes y a su interacción con los demás. [4]

Para estimular a un niño o niña con discapacidad motora, debemos entregar estabilidad en algunas partes del cuerpo y estimular la movilidad en otras. Para que un niño/a se desarrolle en su área motriz es importante que pueda jugar y explorar sus movimientos en todas las posturas.

Es muy importante favorecer la inclusión de estos niños y propiciar la autonomía para el desplazamiento y el manejo de sus útiles tanto en casa como en el aula.

En el año 2020, el Instituto Nacional de Estadística (NE) publicaba que en España existe una tasa de población de niños y niñas con discapacidad con edades entre 6 y 15 años de 43,1 por cada 1.000 habitantes y una tasa de niños y niñas de 2 a 5 años con limitaciones de 78,5 por cada 1.000 habitantes. Según dicho estudio, en España, un total de 106.300 personas de entre 6 y 15 años presenta algún tipo de discapacidad (70.300 niños y 36.000 niñas), siendo la deficiencia más frecuente la motora: el 55,7% de las personas tiene discapacidad debido a un problema en huesos y articulaciones. [5] [6]

Un informe de UNICEF del año 2021 expone que 1 de cada 10 niños y niñas sufren discapacidad en todo el mundo. El informe pone de manifiesto los obstáculos a los que se enfrentan los niños con discapacidad para participar plenamente en sus sociedades y los efectos negativos para la salud y la sociedad que a menudo se derivan de esta situación. [7]

En este sentido es fundamental sensibilizar a la sociedad. Necesitamos que los gobiernos garanticen que los niños con discapacidad tengan un acceso igualitario e inclusivo a la educación. Los Objetivos de Desarrollo Sostenible (ODS) constituyen un llamamiento universal a la acción para poner fin a la pobreza, proteger el planeta y garantizar que todas las personas en todo el mundo disfruten de paz y prosperidad. De estos 17 ODS aprobados a nivel mundial por todos los Estados Miembros de las Naciones Unidas, el ODS número cuatro, persigue garantizar a todos los niños y niñas una educación gratuita, equitativa, inclusiva y de calidad. [8]

1.2 Objeto

Este Trabajo de Fin de Grado tiene por objeto el diseño e implementación de un prototipo que sirva como sensor de movimiento de la cabeza para niños con dificultades motoras, de tal forma que los datos recogidos por dicho sensor puedan resultar de utilidad para medir la dificultad motora del niño y sobre todo para ir mirando cómo va mejorando si se le aplica algún tipo de terapia.

1.3 Solución adoptada

Se ha utilizado una unidad de medición inercial o IMU (Inertial Measurment Units) de 9 grados de libertad (DoF) pues combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes, para obtener la orientación respecto al eje X e Y, además de un magnetómetro de 3 ejes para sacar la orientación. La unidad de medición inercial que se eligió fue el sensor MPU6050. Los datos obtenidos por este sensor son transferidos a un ordenador donde se almacenan, a la vez que se visualizan en una gráfica.

2. Fundamentos teóricos

2.1 ¿Qué es un IMU?

Es un dispositivo electrónico que mide e informa acerca de la velocidad, orientación y fuerzas gravitacionales de un aparato, usando una combinación de acelerómetros y giróscopos. La IMU es el componente principal de los sistemas de navegación inercial usados en aviones, naves espaciales, buques y misiles guiados entre otros. En este uso, los datos recolectados por los sensores de una IMU permiten a un computador seguir la posición del aparato, usando un método conocido como navegación por estima.

Una IMU funciona detectando la actual tasa de aceleración usando uno o más acelerómetros, y detecta los cambios en atributos rotacionales tales como cabeceo, alabeo y guiñada usando uno o más giróscopos. Contiene tres acelerómetros y tres giróscopos. Los acelerómetros están colocados de tal forma que sus ejes de medición son ortogonales entre sí. Ellos miden la aceleración inercial, también conocida como fuerzas G.

-Los acelerómetros no tienen deriva a medio o largo plazo. Sin embargo, se ven influenciados por los movimientos del sensor y el ruido por lo que no son fiables a corto plazo. También cabe destacar que también miden la aceleración estática de la gravedad, que es como se calcula la orientación con la vertical.

-Los giroscopios son aparatos eficientes para movimientos cortos o bruscos, pero al usar giroscopios que miden la velocidad angular, acumulan los errores y el ruido en la medición, por lo que a medio o largo plazo tienen deriva.

2.2 Orientación con IMU

Para determinar la distribución de un objeto con respecto a una base en un espacio tridimensional, debemos establecer su posición y orientación. Se necesitan tres parámetros para determinar la posición relativa y otros tres para determinar su orientación.

En la vida cotidiana estamos acostumbrados a abordar la posición de un objeto, y sabemos que existen diferentes formas de expresar su posición relativa matemáticamente. Por ejemplo, podemos utilizar coordenadas cartesianas (X, Y, Z), coordenadas cilíndricas o coordenadas esféricas.

Sin embargo, no estamos tan familiarizados con la parametrización de la orientación, ya que matemáticamente es más compleja. Al igual que con la posición, hay varias formas de expresar la orientación relativa de dos sistemas, como los ángulos de Euler, las matrices de rotación o los cuaterniones.

Uno de los enfoques más comunes e intuitivos son los ángulos de navegación, también conocidos como ángulos de Tait-Bryan, en los que la orientación se representa mediante tres rotaciones ortogonales alrededor de los ejes X (roll), Y (pitch) y Z (yaw), como se observa en la figura 1.

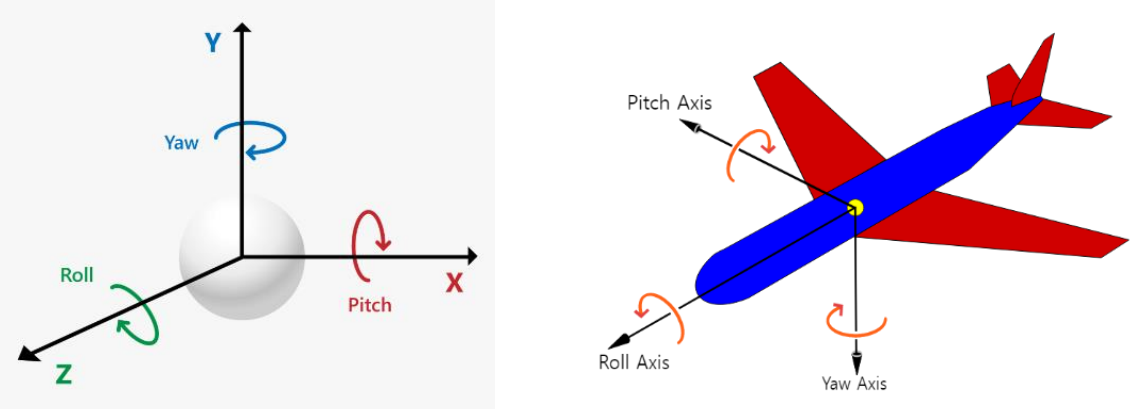

Figura 1: Ángulos de rotación

Los ángulos de Tait-Bryan son una variación de los ángulos de Euler, introducidos por el matemático Leonhard Euler en su estudio de la mecánica de sólidos rígidos.

Estos ángulos son considerados intuitivos y simples. Además, tienen la ventaja de ser conmutativos, lo que significa que el resultado de la orientación es independiente del orden en el que se apliquen las rotaciones (a diferencia de otras representaciones de rotación).

Sin embargo, presentan una importante desventaja conocida como bloqueo cardán, que afecta a todos los sistemas de ángulos de Euler. Cuando el eje Y gira aproximadamente ±90º, los ejes X y Z se superponen, lo que provoca que el sistema degenera a 2 grados de libertad. Esto da lugar a un punto singular no diferenciable que puede ocasionar problemas de estabilidad en los cálculos.

El bloqueo cardán es una desventaja tan significativa que la forma habitual de representar las rotaciones es mediante el uso de cuaterniones. Los cuaterniones son una extensión de los números complejos introducida por Hamilton en 1843, que emplean tres unidades imaginarias: i, j y k, como se ve en la ecuación 1.

$$i^2 = j^2 = k^2 = i \cdot j \cdot k = -1$$

Ecuación 1 de números complejos

Es bien sabido que los números complejos se pueden utilizar para representar vectores en el plano. De manera equivalente, un cuaternión permite expresar un vector en el espacio tridimensional.

Para expresar la rotación de un punto con un ángulo α alrededor de un vector arbitrario con coordenadas Ex, Ey, Ez, el cuaternión resultante se expresa mediante la ecuación 2:

$$q = cos\left(\frac{\alpha}{2}\right) + i \cdot \left(E_x \cdot sin\left(\frac{\alpha}{2}\right)\right) + j \cdot \left(E_y \cdot sin\left(\frac{\alpha}{2}\right)\right) + k \cdot \left(E_z \cdot sin\left(\frac{\alpha}{2}\right)\right)$$

Ecuación 2 de ángulo de rotación

En comparación con los ángulos de Euler, los cuaterniones son más fáciles de combinar y evitan el problema del bloqueo cardán. En comparación con las matrices de rotación, son más eficientes y más estables numéricamente.

Debido a estas ventajas, los cuaterniones son una de las formas más comunes de expresar la orientación en gráficos por computadora. También son la elección preferida al trabajar con orientaciones en IMU (Unidades de Medición Inercial) y en proyectos de Arduino debido a su eficiencia y estabilidad numérica.

2.3 Filtrado de señal en un IMU

Para aprovechar las ventajas a corto plazo del giroscopio y las ventajas a medio y largo plazo del acelerómetro, es necesario combinar y filtrar la señal registrada en bruto (RAW).

Existen varios filtros disponibles, siendo el filtro de Kalman el más conocido. Fue desarrollado en 1960 por Rudolf E. Kalman y se considera uno de los grandes descubrimientos del siglo XX debido a sus implicaciones en el proceso de filtrado de sensores (no solo IMUs) y su contribución a la carrera espacial.

En términos generales, el filtro de Kalman realiza una estimación del valor futuro de la medición y luego compara este valor real con un análisis estadístico para compensar los errores en futuras mediciones.

Sin embargo, la versión general del filtro de Kalman implica cálculos complejos que pueden resultar demasiado exigentes en términos de implementación y tiempo de cálculo para Arduino.

Por esta razón, es común utilizar un filtro más simple conocido como filtro complementario. En realidad, el filtro complementario puede considerarse una simplificación del filtro de Kalman que prescinde por completo del análisis estadístico.

Existen varias formulaciones para un filtro complementario. En su forma más básica, el filtro complementario puede expresarse mediante la ecuación 3:

$$\theta = A \cdot (\theta_{prev} + \theta_{gyro}) + B \cdot \theta_{accel}$$

Ecuación 3 de filtro complementario

Donde A y B son dos constantes que, inicialmente, pueden tomar los valores de 0.98 y 0.02 respectivamente. Es posible ajustar el filtro modificando los valores de A y B, siempre y cuando se cumpla la condición de que la suma de ambos sea igual a 1. [9][10][11]

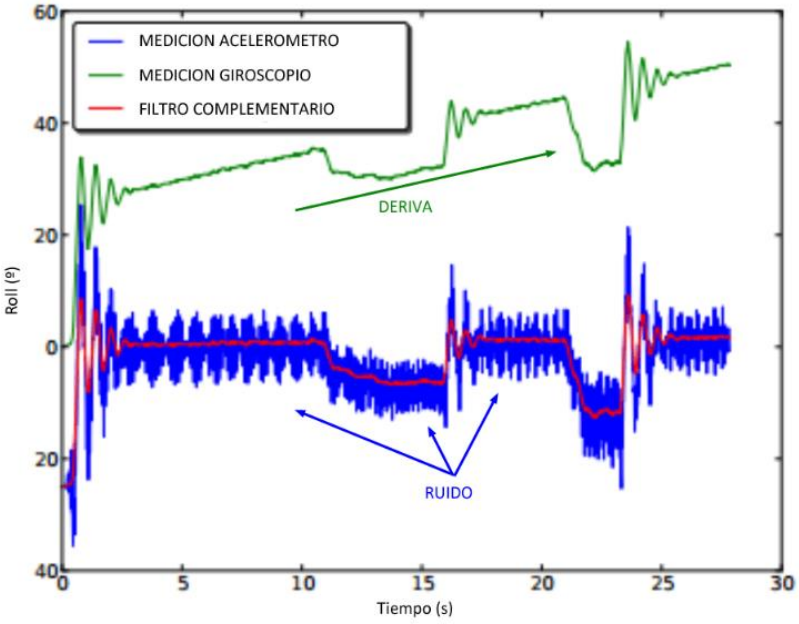

Figura 2: Gráfica Filtro Complemento

En la gráfica de la figura dos se aprecia el resultado de aplicar el filtro complementario a una medición de un acelerómetro y giróscopo.

El filtro complementario actúa como un filtro paso alto para las mediciones del giroscopio y como un filtro paso bajo para la señal del acelerómetro. Esto significa que la señal del giroscopio tiene más influencia a corto plazo, mientras que la señal del acelerómetro prevalece en el medio y largo plazo. Esto es exactamente lo que buscamos para compensar las ventajas y desventajas de ambos sensores.

En la mayoría de las aplicaciones domésticas, el filtro complementario es suficiente y proporciona valores muy similares al filtro de Kalman, siempre y cuando los valores de A y B estén calibrados correctamente.

Existen versiones más sofisticadas del filtro complementario, así como versiones simplificadas y unidimensionales del filtro de Kalman que pueden ser implementadas en Arduino. Estos conceptos se pueden explorar en futuras entradas avanzadas.

Mientras tanto, podemos utilizar IMUs como el MPU6050 y los filtros mencionados en nuestros proyectos de electrónica y Arduino. [9][10][11]

3 Componentes

3.1 Arduino UNO

Arduino Uno es una placa de microcontrolador basada en el ATmega328P. Tiene 14 pines de entrada/salida digital (de los cuales 6 se pueden usar como salidas PWM), 6 entradas analógicas, un resonador cerámico de 16 MHz, una conexión USB, un conector de alimentación, un cabezal ICSP y un botón de reinicio. Para hacerlo funcionar se debe conectarlo a un ordenador o batería. Este tipo de Arduino es el más simple de todos y, a su vez, el más utilizados, sobre todo para las personas que comienzan a trabajar con Arduino.

El Arduino Uno se ha convertido en una herramienta invaluable tanto para principiantes como para expertos en electrónica, ya que brinda una plataforma flexible y accesible para el desarrollo de proyectos y la experimentación. [9]

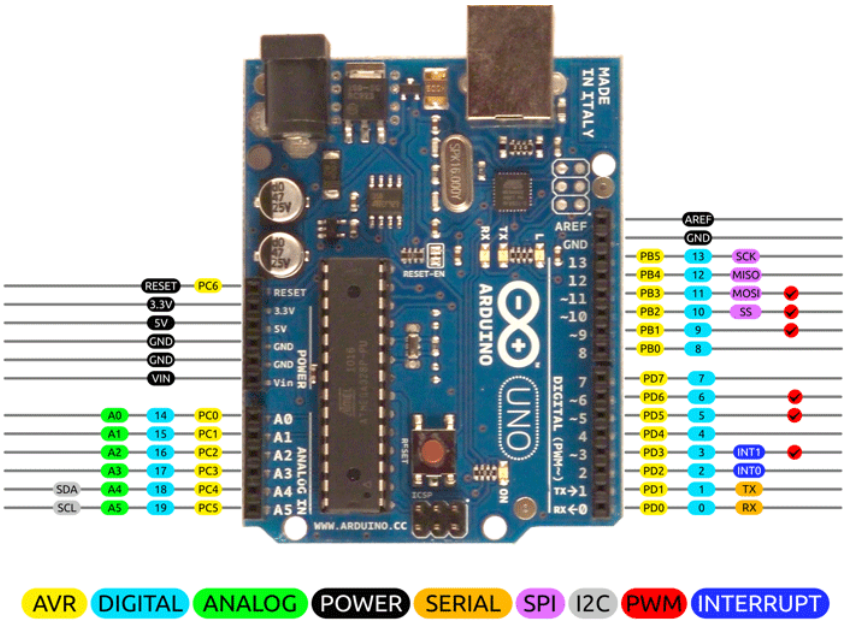

Figura 3: PINOUT Arduino Uno

A la hora de programarlo hay que tener en cuenta que no todos los pines tienen las mismas funciones, por ello hay que fijarse en el esquema de la figura 3 antes de conectar los diferentes componentes y programarlos. [12][13]

3.2 Arduino NANO

Arduino Nano es una placa pequeña, completa y compatible con placas de prueba basada en ATmega328. Tiene los mismos pines que el Arduino Uno, pero colocados de manera diferente. Carece de un conector de alimentación de CC y funciona con un cable USB Mini-B en lugar de uno estándar. [14]

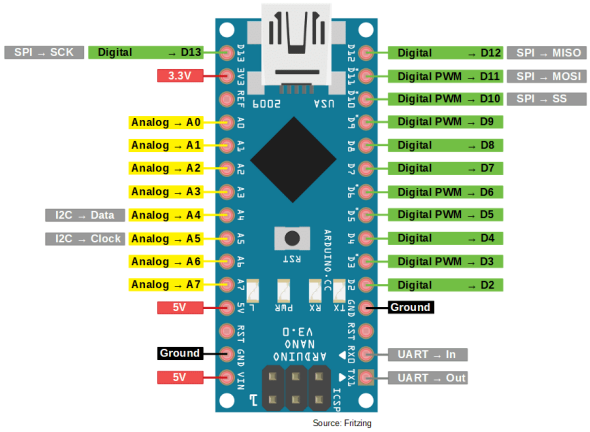

Figura 4: PINOUT Arduino Nano

Al igual que con el Arduino Uno hay que fijarse bien en el PINOUT de la figura 4, para asegurar su correcto funcionamiento.

3.3 Sensor de movimiento MPU6050

EL MPU6050 es una unidad de medición inercial o IMU (Inertial Measurment Units) de 6 grados de libertad pues combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Este sensor es muy utilizado en navegación, goniometría, estabilización, etc.

EL módulo Acelerómetro MPU tiene un giroscopio de tres ejes con el que podemos medir velocidad angular y un acelerómetro también de 3 ejes con el que medimos los componentes X, Y, Z de la aceleración.

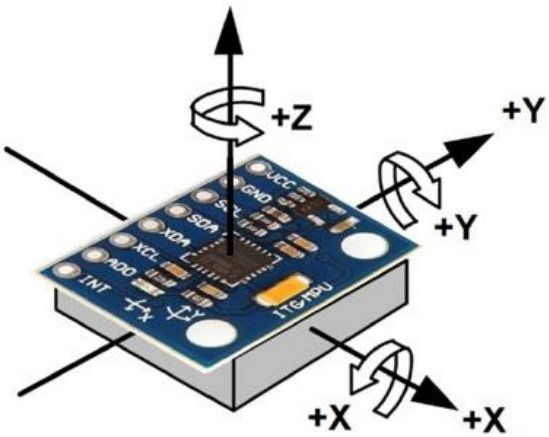

Figura 5: MPU6050

La dirección de los ejes está indicada en el módulo el cual hay que tener en cuenta para no equivocarnos en el signo de las aceleraciones, como se aprecia en la figura 5.

La comunicación del módulo es por I2C, esto le permite trabajar con la mayoría de microcontroladores. Los pines SCL y SDA tienen una resistencia pull-up en placa para una conexión directa al microcontrolador o Arduino. [15][16]

3.4 Módulo de comunicación NRF24L01

El NRF24L01 es un pequeño transceptor inalámbrico de muy bajo consumo y muy fácil de utilizar que funciona en el rango de los 2.4 GHz. Puede enviar y recibir datos, pero no puede hacerlo al mismo tiempo. Esto hace que se abarate su coste y su funcionamiento sea muy sencillo y robusto. Se puede utilizar en cualquier proyecto que necesite enviar y/o recibir datos de forma inalámbrica sin complicaciones. Lo ideal es tener una pareja y hacer un enlace completo. Es totalmente compatible con Arduino.

Se alimenta con 3.3V y su control de datos se realiza mediante el bus SPI, los pines toleran niveles de 5V así que se puede conectar directamente a cualquier microcontrolador de 5V sin necesidad de ningún conversor. También se utiliza con Raspberry Pi. [17][18]

Tiene una antena integrada, la cual no tiene mucho alcance. Lo normal son unos 10 o 20 metros aproximadamente en interiores y en campo abierto igual algo más. Este componente se aprecia en la figura 6.

Figura 6: Pareja de NRF24L01

3.5 Batería con regulador

Se utilizó una batería de litio de 3,7 V junto a una fuente boost para conseguir 5 voltios, además de tener un regulador de voltaje para convertir la salida de una batería en una fuente de alimentación estable de 5 V. El regulador de voltaje es un dispositivo que controla la tensión de salida de la batería y la mantiene constante a pesar de las variaciones en la carga y el nivel de carga de la batería.

Aquí hay una descripción básica de cómo funciona esta configuración:

- Batería con fuente boost: La configuración comienza con una batería de litio que suministra una tensión nominal de 3,7 voltios, gracias a la fuente boost.
- Regulador de voltaje: Se utiliza un regulador de voltaje para garantizar que la tensión de salida se mantenga en 5 V de manera estable, incluso cuando la carga de la batería varíe.
- Circuito de regulación: El regulador de voltaje se conecta al terminal de salida de la batería y regula la tensión a 5 V.
- Salida de tensión: La salida del regulador de voltaje se conecta a un puerto o conector que sea compatible con los dispositivos electrónicos que se deseen alimentar con 5 V. Esto puede ser un puerto USB estándar, un conector específico o incluso cables con terminales adecuados.

Esta configuración permite obtener una fuente de alimentación portátil de 5 V estable y más pequeña que una batería portátil comercial, como se observa en la figura 7.[19]

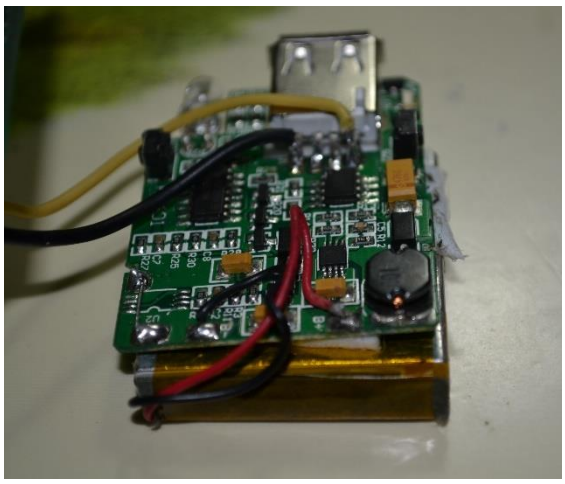

Figura 7: Batería con regulador

3.6 Cable USB con conector macho tipo B

Un cable USB con conector macho tipo B es un tipo de cable utilizado para la conexión de dispositivos electrónicos. El conector macho tipo B se caracteriza por tener una forma cuadrada con una pequeña protuberancia en la parte superior y se utiliza comúnmente en dispositivos periféricos como impresoras, escáneres, dispositivos de almacenamiento externos y otros equipos de comunicación.

El cable USB con conector macho tipo B generalmente se utiliza para establecer una conexión entre el dispositivo periférico, como un Arduino, y un host, como un ordenador. Este tipo de conexión permite la transferencia de datos, la impresión de documentos, el escaneo de imágenes, el acceso a unidades externas, entre otras funciones.

El conector macho tipo B se inserta en el dispositivo y permite la transferencia de datos y energía eléctrica entre el dispositivo y el host mediante un puerto USB correspondiente.

3.7 Cables de conexión

Se han empleado cables macho-hembra de Arduino de 1,5mm de diámetro para la interconexión de los diferentes componentes que han utilizado en el proyecto. Sobre todo, para realizar las pruebas necesarias antes de soldar el circuito definitivo. Los cables que hemos utilizados son los que se aprecian en la figura 8. [20]

Figura 8: Cables Arduino

3.8 Placa universal para soldadura de equipos electrónicos

Las placas universales, también conocidas como placas de circuito impreso (PCB) universales o placas de prototipado, son placas de base utilizadas para soldar componentes electrónicos y crear prototipos de circuitos electrónicos. Estas placas ofrecen una matriz de agujeros o pads de soldadura que permiten la conexión de componentes y la creación de circuitos personalizados.

Aquí hay algunos aspectos importantes sobre las placas universales para soldar equipos electrónicos:

- Diseño y estructura: Las placas universales están hechas de un material aislante, como fibra de vidrio o resina epoxi, en el cual se encuentran dispuestos los agujeros o pads de soldadura. Estos agujeros están organizados en filas y columnas, lo que facilita la colocación y conexión de componentes electrónicos.
- Matriz de conexión: La matriz de agujeros o pads de soldadura proporciona puntos de conexión para los componentes electrónicos. Cada agujero o pad está destinado a recibir una terminal de componente y se suelda para asegurar la conexión eléctrica.
- Versatilidad y flexibilidad: Son versátiles y permiten la soldadura de una amplia gama de componentes electrónicos, como resistencias, condensadores, transistores, diodos y circuitos integrados. Esto las hace adecuadas para la creación de prototipos y la realización de proyectos personalizados.
- Soldadura: La soldadura con estaño es el proceso mediante el cual los componentes electrónicos se unen a los pads de soldadura en la placa. Esto crea una conexión eléctrica sólida y duradera.
- Reusabilidad: Las placas universales son reutilizables, lo que significa que los componentes se pueden retirar y volver a soldar en diferentes ubicaciones o configuraciones según sea necesario.
- Limitaciones: Son adecuadas para proyectos de tamaño pequeño o mediano. Para circuitos más complejos o aplicaciones de producción en masa, es común utilizar placas de circuito impreso diseñadas específicamente para el proyecto. [21]

Las placas utilizadas para nuestro prototipo son las de la figura 9:

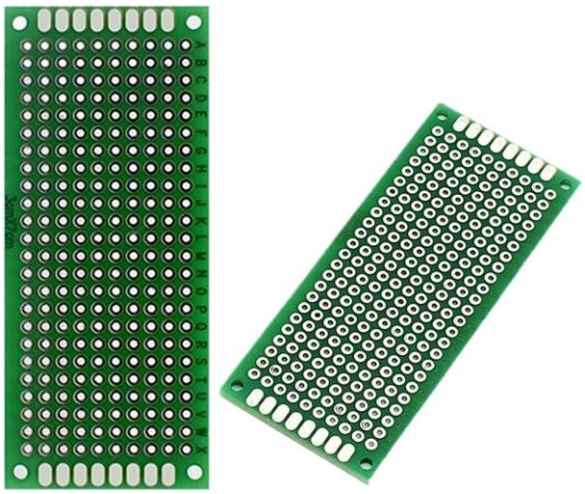

Figura 9: Placa de soldadura

3.9 Equipo informático (CPU, monitor, teclado y ratón e impresora 3D)

Este trabajo se ha desarrollado en un ordenador portátil Asus con sistema operativo Windows 8 con monitor y teclado integrado, haciendo uso de un ratón inalámbrico.

Para la fabricación de la caja se ha empleado la impresora 3D ubicada en el aula de PB.022 (Taller de investigación) de la Escuela Superior de Ingeniería y Tecnología (ESIT).

3.10 Caja de PVC para alojar el prototipo

El conjunto formado por el Arduino NANO, el sensor MPU6050, la batería y el módulo de comunicación van alojador en el interior de una caja de PVC de color negro, la cual ha sido previamente diseñada mediante el software Freecad y construida mediante la impresora 3D.

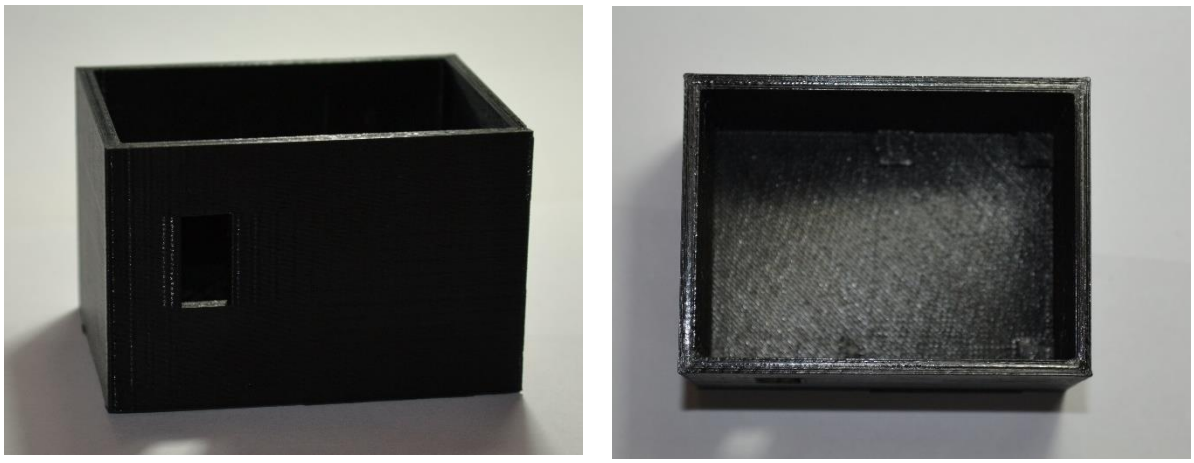

Figura 10: Caja de PVC

La caja dispone de una tapa rectangular transparente con un orificio circular, a través del cual, permite accionar el botón de encendido de la batería.

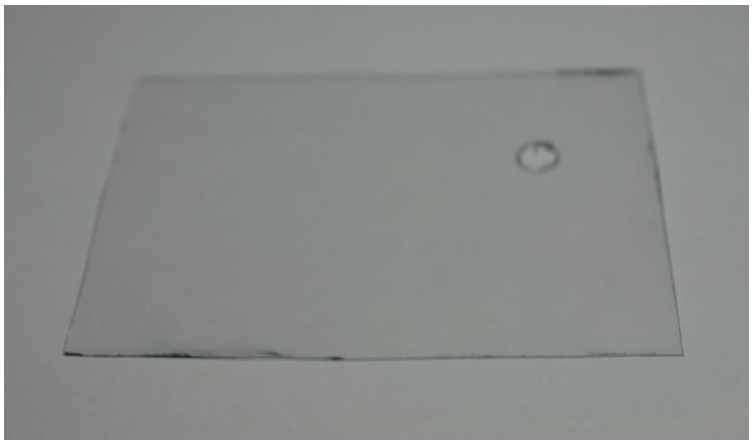

Figura 11: Tapa de la caja

3.11 Soporte de sujeción

Para realizar el soporte de sujeción para el prototipo se usó una correa con una hebilla para poder ajustarse a diferentes tamaños, tal y como se muestra en la figura 12.

Figura 12: Soporte de sujeción

4 Metodología empleada

Para llevar a cabo las tareas mencionadas anteriormente, se han utilizado los siguientes programas informáticos

1) Arduino IDE con las siguientes librerías

Arduino IDE es un entorno de desarrollo integrado utilizado para programar y desarrollar proyectos con placas Arduino. Es una herramienta de software que facilita la escritura, compilación y carga de código en las placas Arduino.

Algunos aspectos importantes de Arduino IDE son:

- Interfaz de usuario: Arduino IDE cuenta con una interfaz de usuario simple y fácil de usar. Tiene una barra de menú principal que proporciona acceso a varias opciones y funciones, y una ventana de código donde se escribe el programa Arduino.
- Lenguaje de programación: Utiliza un lenguaje de programación basado en C/C++, que es específico de Arduino. Sin embargo, para facilitar su uso, se proporciona una librería estándar de funciones y macros que simplifica la programación en Arduino.
- Compilación y carga: Facilita la compilación y carga del programa en la placa Arduino. Después de escribir el código, se utiliza la función "Verificar" para compilarlo y verificar si hay errores. Si la compilación es exitosa, se puede utilizar la función "Cargar" para transferir el programa compilado a la placa Arduino, lo que permite que la placa ejecute el código.
- Librerías y ejemplos: Ofrece una amplia biblioteca de librerías predefinidas que contienen funciones y rutinas comunes utilizadas en proyectos de Arduino. Estas librerías facilitan el desarrollo de proyectos complejos y ahorran tiempo al proporcionar una funcionalidad lista para usar. Además, el IDE incluye una amplia variedad de ejemplos de código que ilustran el uso de las funciones y características de las placas Arduino, lo que facilita el aprendizaje y la comprensión de la programación de Arduino.
- Plataformas y compatibilidad: Es compatible con una amplia gama de placas Arduino, incluyendo Arduino Uno, Arduino Mega, Arduino Nano, entre otras. También es compatible con diferentes sistemas operativos como Windows, macOS y Linux.
- Personalización y expansión: Permite la personalización y expansión a través de la instalación de complementos, que amplían las funcionalidades y permiten la programación de otras placas y microcontroladores no nativos de Arduino.

Es decir, Arduino IDE es un entorno de desarrollo integrado que facilita la programación de proyectos con placas Arduino. Proporciona una interfaz de usuario intuitiva, herramientas de compilación y carga de código, librerías predefinidas y ejemplos de código para acelerar el desarrollo de proyectos. Es una herramienta muy utilizada y accesible para principiantes y entusiastas de la electrónica que desean explorar el mundo de Arduino. [22]

Figura 13: Arduino IDE

En la figura 13 podemos apreciar la interfaz del programa Arduino Uno.

2) Processing

Processing es un entorno de desarrollo de software de código abierto y un lenguaje de programación diseñado para la creación de proyectos interactivos, visuales y de arte digital. Presenta las siguientes características:

- Diseño orientado a la creatividad: Está diseñado específicamente para facilitar la creación de proyectos creativos y visuales. Proporciona una sintaxis sencilla y una amplia variedad de funciones y librerías para trabajar con gráficos, animaciones, sonido y otros elementos multimedia.
- Lenguaje de programación: Utiliza un lenguaje de programación basado en Java, pero con una sintaxis simplificada y una serie de funciones y bibliotecas específicas para el desarrollo visual. Esto lo hace más accesible para los principiantes y aquellos con poca experiencia en programación.
- Entorno de desarrollo integrado: Ofrece un entorno de desarrollo integrado que incluye un editor de código, un intérprete y un visor de resultados en tiempo real. Esto permite a los usuarios escribir y ejecutar código de forma rápida y sencilla, facilitando el proceso de iteración y experimentación.
- Gráficos y visualización: Proporciona una serie de funciones y librerías para trabajar con gráficos y visualización. Permite crear y manipular formas geométricas, imágenes, animaciones, y aplicar efectos visuales y filtros.
- Comunidad y recursos: Cuenta con una gran comunidad de desarrolladores y usuarios que comparten proyectos, tutoriales y librerías adicionales. Esto facilita el aprendizaje, la resolución de problemas y la colaboración en proyectos.
- Multiplataforma: Es compatible con múltiples sistemas operativos, como Windows, macOS y Linux. Esto permite a los usuarios desarrollar y ejecutar sus proyectos en diferentes plataformas sin mayores problemas.[23]

Integración con otras herramientas: Processing se puede combinar con otras herramientas y entornos de programación, como Arduino, para crear proyectos que combinen interacción física y visual, como se ve en la figura 14.

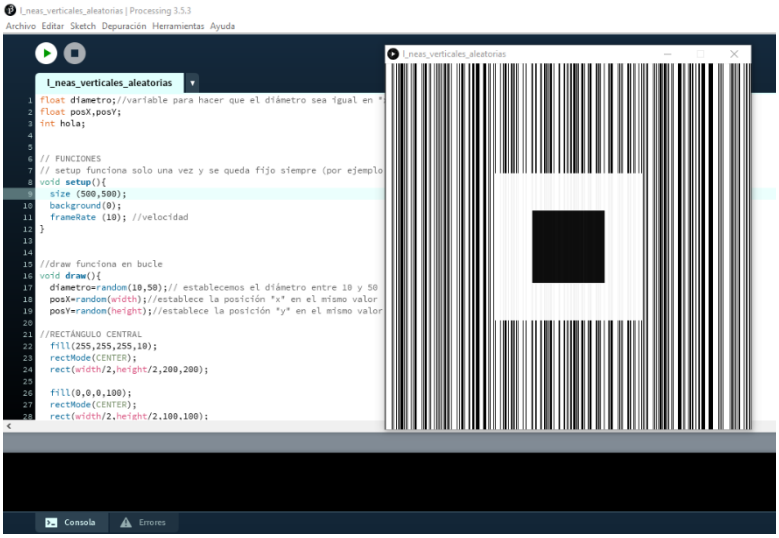

Figura 14: Processing

3) FreeCad

Es un software de modelado 3D de código abierto que se utiliza para el diseño de objetos y modelos en tres dimensiones. Es una herramienta muy versátil que permite a los usuarios crear diseños detallados en diversos campos, como la ingeniería mecánica, la arquitectura, la electrónica y más. Muestra diferentes aspectos importantes:

- Diseño de objetos 3D: Permite el diseño y modelado de objetos tridimensionales de manera precisa. Puedes crear geometrías complejas utilizando una amplia gama de herramientas. También puedes trabajar con diferentes tipos de objetos, como sólidos, superficies, alambres y mallas.
- Biblioteca de partes y módulos: Posee una biblioteca de partes y módulos que facilita el diseño al proporcionar una amplia gama de componentes y objetos predefinidos. También es posible importar y exportar archivos en diferentes formatos, como STEP, IGES, STL y más.
- Comunidad y documentación: Cuenta con una comunidad activa de usuarios y desarrolladores que proporcionan soporte, comparten conocimientos y colaboran en el desarrollo del software. También existe una amplia documentación disponible, incluyendo tutoriales, guías y foros de discusión, lo que facilita el aprendizaje y el acceso a recursos adicionales. [24]

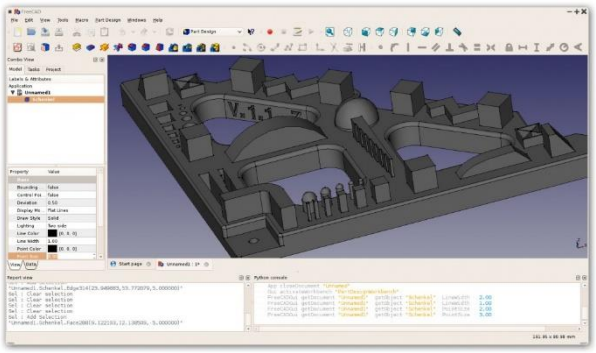

Figura 15: FreeCad

4) Ultimaker Cura

Es un software gratuito y de código abierto utilizado para preparar impresiones 3D. Permite ajustar los modelos 3D, configurar parámetros de impresión, visualizar y simular el proceso de impresión. Cuenta con perfiles predefinidos para diferentes impresoras 3D y es altamente personalizable a través de complementos. Cura tiene una comunidad activa y se actualiza regularmente para mejorar sus características. Es una herramienta popular en el ámbito de la impresión 3D debido a su facilidad de uso y funcionalidades avanzadas.

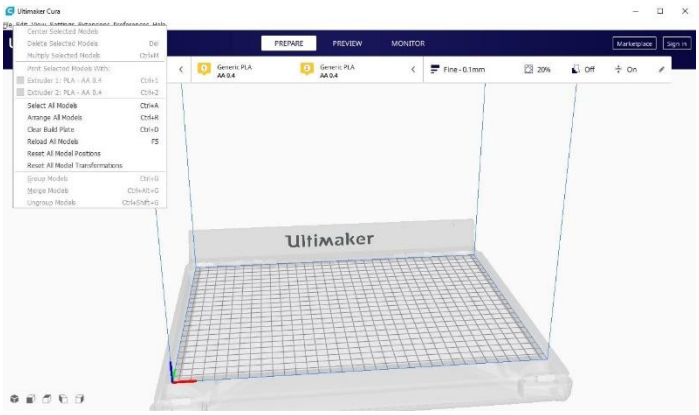

Figura 16: Ultimaker Cura

5 Trabajo desarrollado

El trabajo se ha desarrollado siguiendo el siguiente orden de tareas:

1) En primer lugar, se procedió a recabar información sobre el sistema de medición Inercial IMU consultando diferentes publicaciones de internet de carácter técnico. Se escogió este sistema por su idoneidad y sencillez para medir ángulos y aceleraciones en las tres dimensiones X, Y, Z.

2) En segundo lugar, se procedió a la selección de los componentes que constituirían el prototipo. Se necesitaba contar con un sensor que fuera capaz de medir el ángulo de rotación en los tres ejes, es decir un IMU, y a su vez contar con dos módulos de comunicación capaces de transmitir los datos obtenidos al ordenador. Además, se escogió el Arduino como microcontrolador, a través del cual se programaría tanto el sensor, como los dos módulos de comunicación empleados. El conjunto se completó con una batería de litio junto con un regulador de tensión que pudiera alimentar el prototipo.

3) Posteriormente, una vez escogidos los componentes electrónicos, se implementó la programación y calibración del sensor MPU6050 utilizando la herramienta informática Arduino IDE, para recabar la medida de los diferentes ángulos de inclinación, aplicando la fórmula matemática del filtro complementario. Para conocer el lenguaje de programación y utilizarlo correctamente, se procedió a documentarse en distintos foros informáticos de internet.

4) Por otro lado, se programaron los dos módulos de comunicación NRF24L01 mediante los dos microcontroladores Arduino UNO, para su interconexión, usando uno como emisor y el otro como receptor. Para comprobar el correcto funcionamiento de esta interconexión, se envió un “Hola mundo” desde el emisor hasta el receptor.

5) A continuación se unificaron los dos puntos anteriores en la programación para posibilitar la transmisión de datos de los ángulos medidos por el emisor al receptor. Una vez recibidos por el receptor, dichos datos son leídos por la CPU en la que reside el programa implementado, por medio de un conector USB. Así, los datos recogidos podrán visualizarse en pantalla.

6) Seguidamente se obtuvieron los datos medidos por el sensor mediante el programa Processing con el fin de mostrarlos en una gráfica y guardarlos en un archivo con formato CSV. La gráfica representa en el eje x el tiempo y en el eje y los datos de la rotación en las tres dimensiones x, y, z. Los datos con formato CSV pueden ser importados por el programa Excel, para generar una tabla ordenada con los resultados medidos.

7) Con el fin de reducir las dimensiones y minimizar el tamaño final del prototipo, se sustituyó el Arduino Uno que actuaba de emisor por un Arduino Nano, el cual tiene los mismos pines que el Arduino Uno pero diferentes conexiones.

8) Con el fin de obtener un resultado más compacto del emisor, se soldaron todos los componentes a una placa universal: el Arduino Nano, el sensor MPU6050, el

módulo de comunicación NRF24L01 y una fuente de alimentación portátil que pueda garantizar el suministro eléctrico al conjunto. El material utilizado para la soldadura fue el estaño, prestando especial precaución para no conectar al mismo tiempo dos pines de la placa y causar así un cortocircuito.

9) Para alojar en su interior los componentes mencionados en el apartado anterior, se diseñó una caja abierta con forma de prisma, la cual se fabricó con PLA mediante impresión 3D. Para ello se utilizó la impresora 3D ubicada en el taller de investigación situada en edificio de informática de la ULL. Para cerrar la caja, se fabricó una tapa rectangular transparente con un orificio que permitiera encender y apagar el prototipo.

10) Una vez montada la caja con su tapa, se colocó sobre un soporte de sujeción con el objeto de garantizar el agarre del prototipo a la cabeza del niño con dificultades motoras. Como soporte se utilizó la correa interior de un casco infantil de bicicleta.

6 Diseño del circuito

A continuación, se describe el esquema de conexión del circuito electrónico implementado.

En primer lugar, vemos la forma de conectar el Arduino Uno al sensor MPU6050, al que sólo hace falta conectarle cuatro pines: Vcc, GND, SCL y SDA. Como se explica en las figuras 17 y 18.

MPU60650	Arduino
Vcc	5V
Gnd	Gnd
SCL	A5
SDA	A4

Figura 17: Conexionado MPU6050

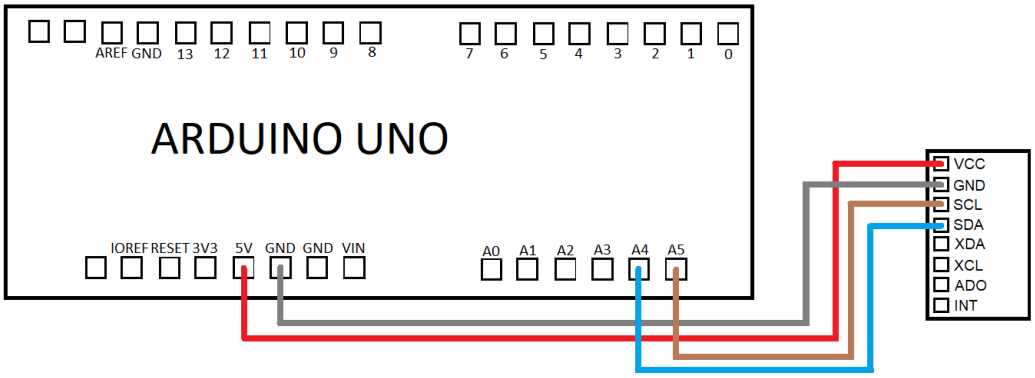

Figura 18: Esquema conexión MPU6050

A continuación vemos el del módulo de comunicación NRF24L01, también con el Arduino Uno, este circuito será el definitivo para el receptor. En este caso hay que conectar siete pines: Vcc, GND, CE, CSN, SCK, MOSI y MISO. Como se desarrolla en las figuras 19 y 20.

NRF24L01	Arduino
GND	GND
VCC	3V3
CE	9
CSN	10
SCK	13
MOSI	11
MISO	12

Figura 19: Conexionado NRF24L01

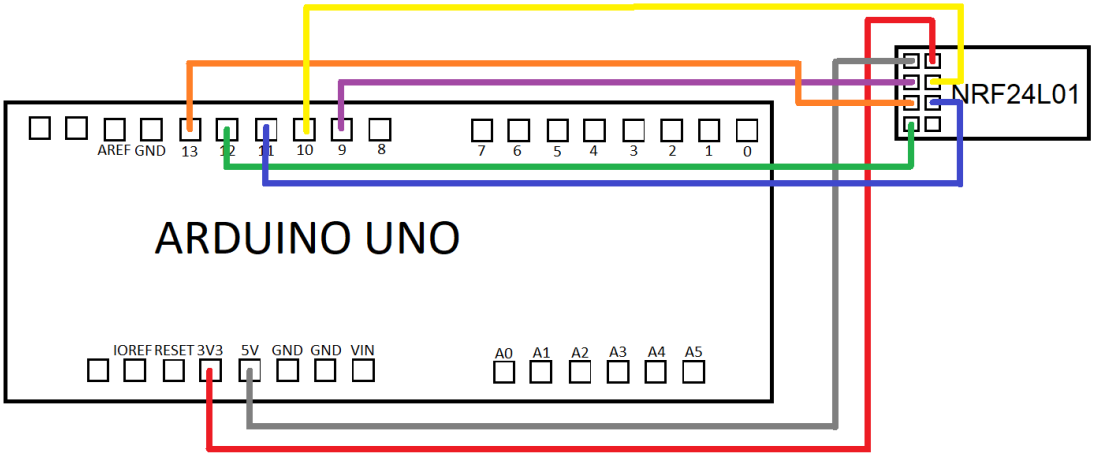

Figura 20: Esquema conexión NRF24L01

Para el circuito definitivo del emisor, en el cual se usa un Arduino Nano, en vez del Arduino Uno se utilizará el siguiente esquema. Teniendo en cuenta que los dos Arduino poseen los mismos pines, pero situados en diferente localización. Además hay que conectar la batería portátil, como se aprecia en el esquema de la figura 20.

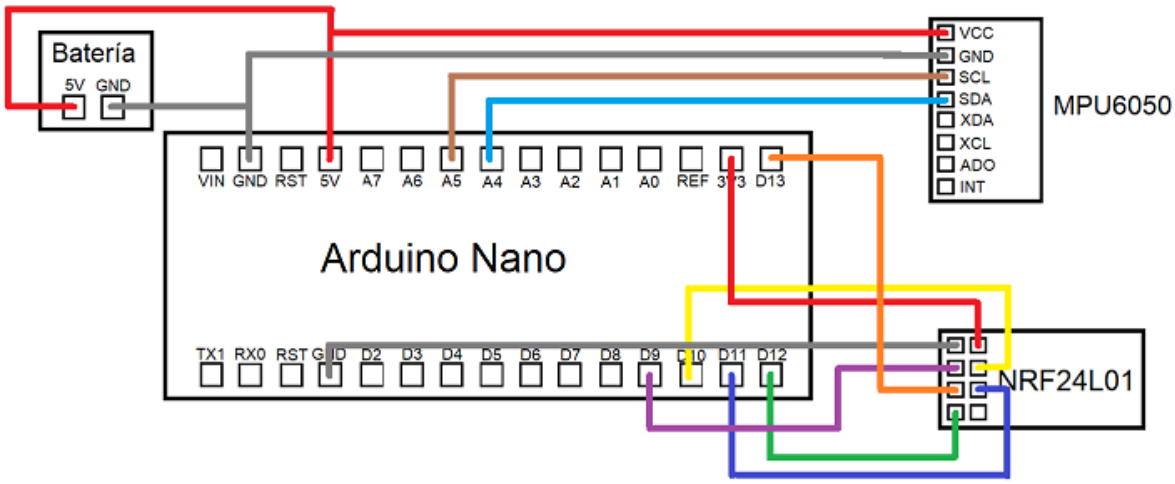

Figura 21: Esquema conexión Arduino Nano

7 Montaje del circuito

El montaje del circuito se realizó siguiendo los siguientes pasos.

En primer lugar se conectó un Arduino UNO al sensor MPU6050 siguiendo el esquema que se comentó anteriormente, y se probó que estuviera todo bien conectado a los pines correspondientes y se estuvieran leyendo los datos de forma correcta, como se ve en la figura 22.

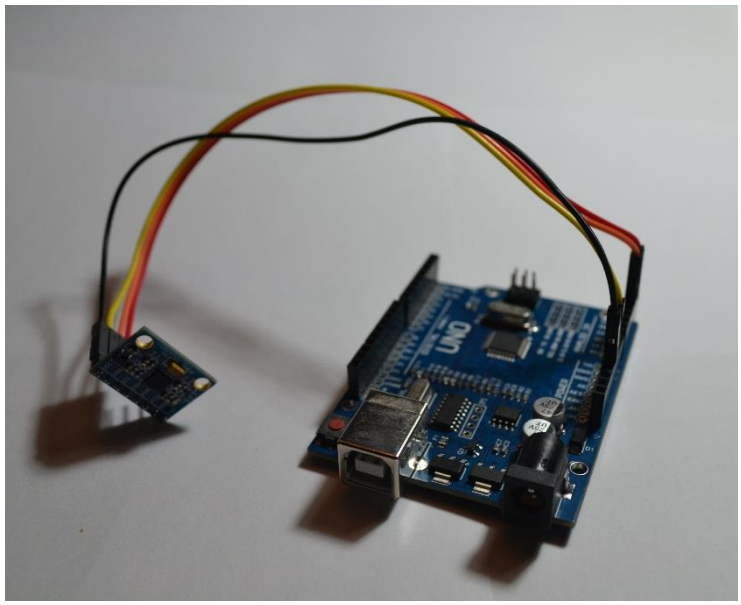

Figura 22: Arduino Uno con MPU6050

Una vez conectado el MPU6050, se conectó el componente de comunicaciones NRF24L01 siguiendo, de igual manera el esquema de conexión. Además de tener este Arduino con el sensor y el componente de comunicaciones, se conectó otro Arduino Uno a otro NRF24L01 para que actuara de receptor, como se observa en la figura 23.

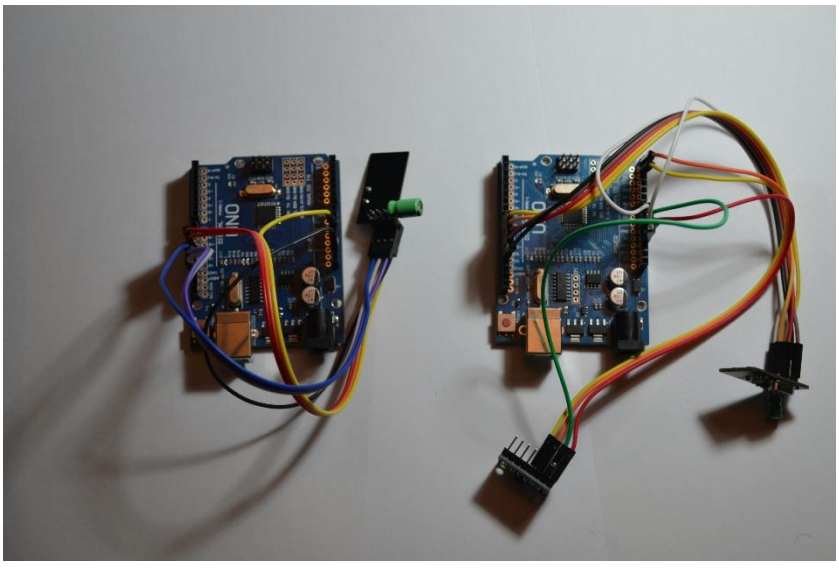

Figura 23: Arduino Uno con MPU6050 y dos NRF24L01

A continuación se conectó una batería portátil al Arduino que actúa de emisor y se comprobó que todo funcionaba de manera correcta, como se aprecia en la figura 24.

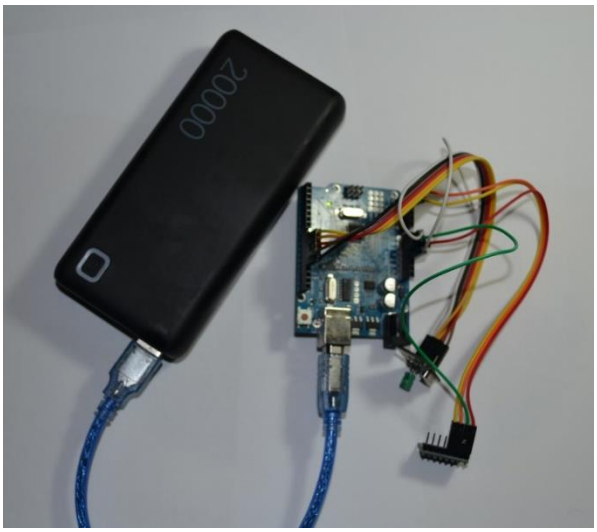

Figura 24:Arduino Uno con batería portátil

Teniendo todo el dispositivo montado y funcionando, se intercambió el Arduino Uno por un Arduino Nano, para hacerlo más compacto, siguiendo el mismo esquema, ya que posee los mismos pines que el Arduino Uno, como se distingue en la figura 25.

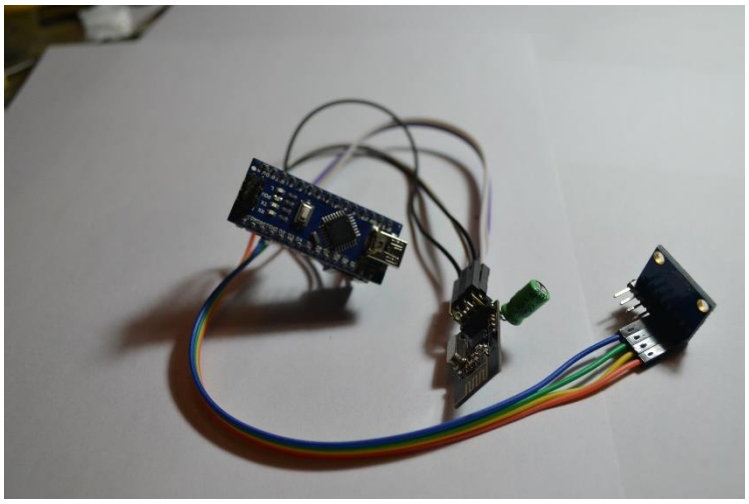

Figura 25: Arduino Nano con MPU6050 y NRF24L01

Seguidamente, se soldaron todos los componentes en la placa de soldadura, añadiendo la batería portátil. Teniendo especial cuidado con que el estaño no toque dos pines a la vez, ya que produciría un cortocircuito y no funcionarían bien dichos pines.

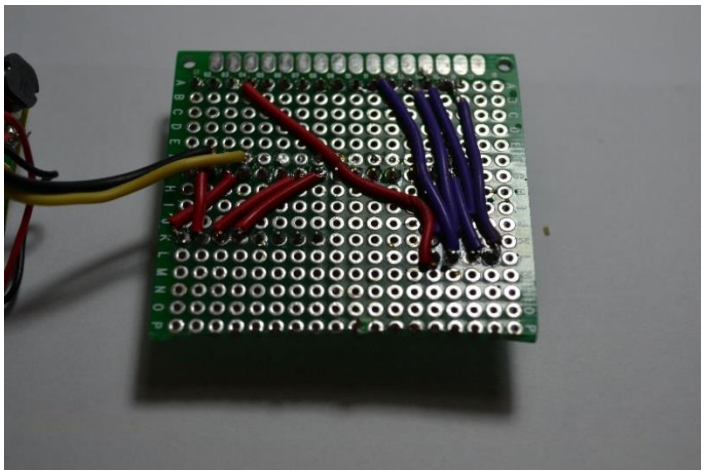

Figura 26: Soldadura de todos los componentes

Una vez soldado todos los pines a sus correspondientes pads y los cables siguiendo las conexiones de los esquemas, se pasa a conectar todos los componentes a la placa de soldadura, como se observa en la figura 2.

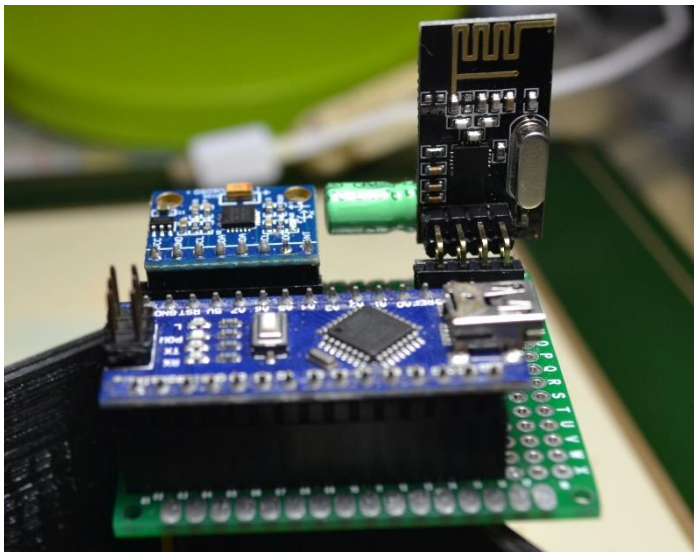

Figura 27: Dispositivo final

A continuación se imprimió la caja mediante la impresora 3D, lo que tuvo una duración de unas 4 horas. Asegurándose, durante y tras la impresión que se había fabricado de forma correcta, siguiendo el diseño hecho anteriormente, como se aprecia en la figura 27.

Seguidamente se comprobaron las medidas de la caja y que todo cabe de manera correcta.

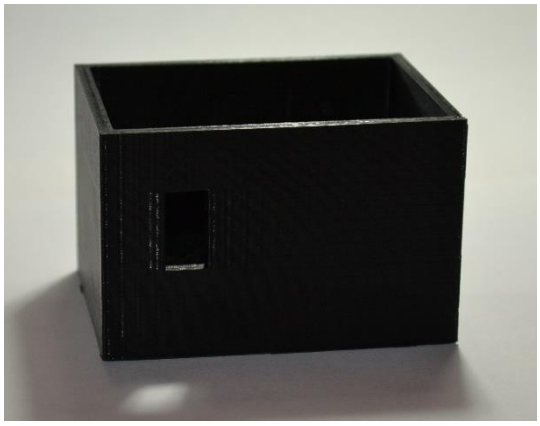

Figura 28: Dispositivo final

Por último se introdujo todo el prototipo en la caja impresa en 3D, pegando con cinta de doble cara la batería al lateral de la caja y metiendo la placa con todos los componentes ya conectados y se probó que todo estuviera bien sujeto y no hubiera peligro de que se moviera el sensor mientras se realizaban las pruebas.

Figura 29: Dispositivo final

Para finalizar se pegó la tapa transparente en su superficie para cerrarla y poder ver el circuito desde fuera y comprobar cuando está encendido o apagado el dispositivo. La tapa dispone de un agujero, para poder encender y apagar la batería.

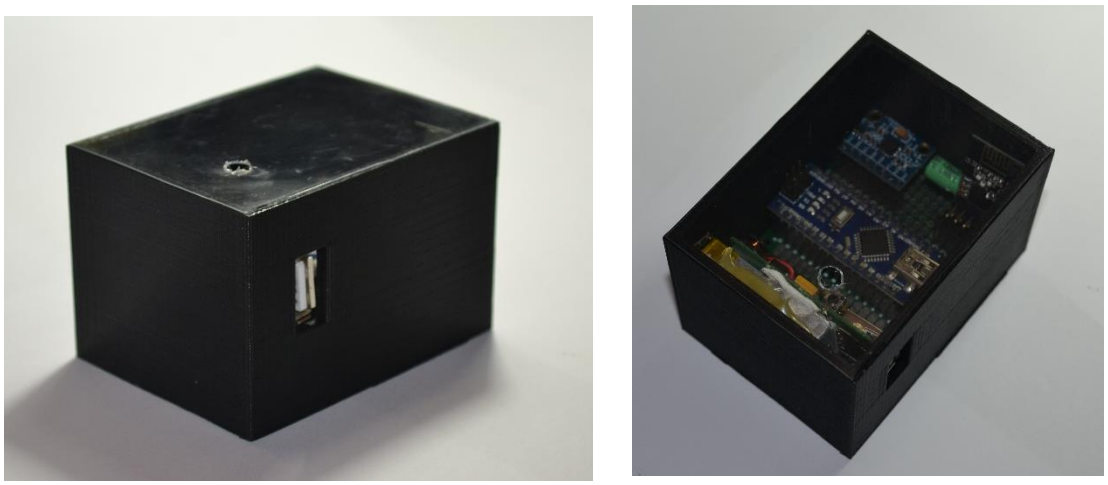

Figura 30: Dispositivo final

8 Programación

Para la realización de este prototipo hizo falta programar en dos programas diferentes, en el Arduino UNO, donde se desarrolló un programa para el emisor y uno para el receptor y en Processing, el cual se centra en la lectura y manipulación de datos realizando tablas y gráficas.

8.1 Programación en Arduino IDE

Como se comentó anteriormente se realizaron dos programas diferentes, uno para el Arduino NANO, que se centra en obtener los ángulos del sensor MPU6050 y de comunicárselo, mediante el NRF24L01 al Arduino receptor. Y otro para el receptor, el cual se encargará de recibir los datos y mostrarlos en pantalla.

1) Programa del Emisor

En primer lugar, se incluyen las librerías que se utilizan en el programa:

- Wire.h → Esta biblioteca le permite comunicarse con dispositivos I2C/TWI. En las placas Arduino con el diseño R3 (pinout 1.0), SDA (línea de datos) y SCL (línea de reloj) están en los encabezados de los pines cerca del pin AREF. Esta librería se puede descargar desde la página oficial de Arduino.[25]
- nRF24L01.h → Proporciona las funciones necesarias para la comunicación con la radio, que se utiliza para transmitir los datos. Se descarga desde github, donde se pueden encontrar muchas librerías.
- RF24.h → Es una extensión de la anterior librería, sirve para obtener una mejor interfaz y poder trabajar mejor con la radio. A l igual que la anterior se descarga desde github.[26]
- SPI.h →Esta biblioteca le permite comunicarse con dispositivos SPI, con Arduino como dispositivo controlador. Esta biblioteca se incluye con todas las plataformas Arduino, por lo que no necesita instalar la biblioteca por separado, es utilizada por el módulo NRF24L01.

Además de declarar todas las variables que se utilizarán más adelante en el código y se establecen los factores de escala para los sensores del MPU6050.

```
#include <Wire.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <SPI.h>

long loopTimer, loopTimer2;
int temperature;
double accelPitch;
double accelRoll;
long acc_x, acc_y, acc_z;
double accel_x, accel_y, accel_z;
double gyroRoll, gyroPitch, gyroYaw;
int gyro_x, gyro_y, gyro_z;
long gyro_x_cal, gyro_y_cal, gyro_z_cal;
double rotation_x, rotation_y, rotation_z;
double freq, dt;
double tau = 0.98;
double roll = 0;
double pitch = 0;

float ang[3];

// 250 deg/s --> 131.0, 500 deg/s --> 65.5, 1000 deg/s --> 32.8, 2000 deg/s --> 16.4
long scaleFactorGyro = 65.5;

// 2g --> 16384 , 4g --> 8192 , 8g --> 4096, 16g --> 2048
long scaleFactorAccel = 8192;
```

Figura 31: Declaración librerías y variables

A continuación, se definen los pines utilizados para la comunicación de la radio y se inicializa la comunicación con la radio. Además, con la constante pipe se introduce la dirección de la tubería de comunicación entre los dos dispositivos que utilizan la radio.

```
28
29  const int pinCE = 9;
30  const int pinCSN = 10;
31  RF24 radio(pinCE, pinCSN);
32
33  // Single radio pipe address for the 2 nodes to communicate.
34  const uint64_t pipe = 0xE8E8F0F0E1LL;
35
```

Figura 32: Declaración pines

Seguidamente, en la función setup(), se inicia la comunicación tanto del MPU6050 como del NRF24L01, teniendo una comunicación serial a 115200 baudios, lo que permite la conexión entre el Arduino y el puerto serial y se llama a la función setup_mpu_6050_registers(), la cual se explicará más adelante.

```
36  void setup() {
37    // Start
38    Wire.begin();
39    Serial.begin(115200);
40    radio.begin();              //Iniciando radio
41    radio.openWritingPipe(pipe);
42
43    // Setup the registers of the MPU-6050 and start up
44    setup_mpu_6050_registers();
45
```

Figura 33: Inicialización de programa

Para finalizar la función setup(), se realiza una función for() en la que se calibra el MPU6050 para su correcto funcionamiento y se reinicia los temporizadores utilizados para medir el tiempo de ejecución del bucle principal mediante loopTimer y loopTimer2.

```
49    // Take 3000 readings for each coordinate and then find average offset
50    for (int cal_int = 0; cal_int < 3000; cal_int ++){
51      if(cal_int % 200 == 0)Serial.print(".");
52      read_mpu_6050_data();
53      gyro_x_cal += gyro_x;
54      gyro_y_cal += gyro_y;
55      gyro_z_cal += gyro_z;
56
57      delay(3);
58    }
59
60    // Average the values
61    gyro_x_cal /= 3000;
62    gyro_y_cal /= 3000;
63    gyro_z_cal /= 3000;
64
65    // Reset the loop timer
66    loopTimer = micros();
67    loopTimer2 = micros();
68  }
```

Figura 34: Calibración del sensor

El bucle loop() comienza calculando la frecuencia del bucle en hercios (Hz), actualiza el valor de loopTimer2 con el tiempo actual en microsegundos y se llama a la función read_mpu_6050_data() para obtener los datos actuales del sensor MPU6050, incluyendo la aceleración y los valores de giro.

```
71  void loop() {
72    freq = 1/((micros() - loopTimer2) * 1e-6);
73    loopTimer2 = micros();
74    dt = 1/freq;
75
76    // Read the raw acc data from MPU-6050
77    read_mpu_6050_data();
78
79    // Subtract the offset calibration value
80    gyro_x -= gyro_x_cal;
81    gyro_y -= gyro_y_cal;
82    gyro_z -= gyro_z_cal;
83
```

Figura 35: Calculo de la frecuencia

A continuación, se realizan diferentes operaciones matemáticas para pasar los valores brutos a medidas adecuadas y utiliza el filtro complemento para combinar la información del acelerómetro y el giroscopio y obtener la orientación del dispositivo. Una vez realizado el cálculo se guardan los valores en el arreglo ang[] y se envían mediante radio.write() al receptor.

```
84    // Convert to instantaneous degrees per second
85    rotation_x = (double)gyro_x / (double)scaleFactorGyro;
86    rotation_y = (double)gyro_y / (double)scaleFactorGyro;
87    rotation_z = (double)gyro_z / (double)scaleFactorGyro;
88
89    // Convert to g force
90    accel_x = (double)acc_x / (double)scaleFactorAccel;
91    accel_y = (double)acc_y / (double)scaleFactorAccel;
92    accel_z = (double)acc_z / (double)scaleFactorAccel;
93
94    // Complementary filter
95    accelPitch = atan2(accel_y, accel_z) * RAD_TO_DEG;
96    accelRoll = atan2(accel_x, accel_z) * RAD_TO_DEG;
97
98    pitch = (tau)*(pitch + rotation_x*dt) + (1-tau)*(accelPitch);
99    roll = (tau)*(roll - rotation_y*dt) + (1-tau)*(accelRoll);
100
101   gyroPitch += rotation_x*dt;
102   gyroRoll -= rotation_y*dt;
103   gyroYaw += rotation_z*dt;
104
105   ang[0] = roll;
106   ang[1] = pitch;
107   ang[2] = gyroYaw;
108
109   radio.write(ang, sizeof ang);
110   delay(10);
111   // Data out serial monitor
112     //Serial.print("Rotacion en X= " );
113     Serial.print(ang[0]);
114     Serial.print(",");
115     //Serial.print(" Rotacion en Y= " );
116     Serial.print(ang[1]);
117     Serial.print(",");
118     ///Serial.print(" Rotacion en z= " );
119     Serial.println(ang[2]);
120     //Serial.println();
121
122   // Wait until the loopTimer reaches 4000us (250Hz) before next loop
123   while (micros() - loopTimer <= 4000);
124   loopTimer = micros();
125 }
```

Figura 36: Filtro Complemento

Cabe destacar dos funciones que se definen al final del código, read_mpu_6050_data(), que utiliza la comunicación I2C para leer los datos brutos del sensor MPU6050 y los almacena en variables correspondientes para su posterior procesamiento y cálculos en el código principal.

```
128 void read_mpu_6050_data() {
129   // Subroutine for reading the raw data
130   Wire.beginTransmission(0x68);
131   Wire.write(0x3B);
132   Wire.endTransmission();
133   Wire.requestFrom(0x68, 14);
134
135   // Read data --> Temperature falls between acc and gyro registers
136   acc_x = Wire.read() << 8 | Wire.read();
137   acc_y = Wire.read() << 8 | Wire.read();
138   acc_z = Wire.read() << 8 | Wire.read();
139   temperature = Wire.read() <<8 | Wire.read();
140   gyro_x = Wire.read()<<8 | Wire.read();
141   gyro_y = Wire.read()<<8 | Wire.read();
142   gyro_z = Wire.read()<<8 | Wire.read();
143 }
```

Figura 37: Función read_mpu_6050_data()

Y el setup_mpu_6050_registers() el cual utiliza la comunicación I2C para escribir los valores de configuración en los registros del sensor MPU-6050. Estas configuraciones permiten establecer el rango de escala del acelerómetro y el giroscopio para obtener mediciones precisas y adecuadas en el código principal.

```
146  void setup_mpu_6050_registers() {
147    //Activate the MPU-6050
148    Wire.beginTransmission(0x68);
149    Wire.write(0x6B);
150    Wire.write(0x00);
151    Wire.endTransmission();
152
153    // Configure the accelerometer
154    // Wire.write(0x__);
155    // Wire.write; 2g --> 0x00, 4g --> 0x08, 8g --> 0x10, 16g --> 0x18
156    Wire.beginTransmission(0x68);
157    Wire.write(0x1C);
158    Wire.write(0x08);
159    Wire.endTransmission();
160
161    // Configure the gyro
162    // Wire.write(0x__);
163    // 250 deg/s --> 0x00, 500 deg/s --> 0x08, 1000 deg/s --> 0x10, 2000 deg/s --> 0x18
164    Wire.beginTransmission(0x68);
165    Wire.write(0x1B);
166    Wire.write(0x08);
167    Wire.endTransmission();
168  }
```

Figura 38: Función setup_mpu_6050_registers()

2) Programa del Receptor

El comienzo del código del receptor es similar al del emisor, donde se incluyen las librerías, se establecen los pines del NRF24L01 y se declaran todas las variables que se utilizan más adelante.

```
1   #include <SPI.h>
2   #include <nRF24L01.h>
3   #include <RF24.h>
4
5   const int pinCE = 9;
6   const int pinCSN = 10;
7   RF24 radio(pinCE, pinCSN);
8
9   // Single radio pipe address for the 2 nodes to communicate.
10  const uint64_t pipe = 0xE8E8F0F0E1LL;
11
12  float data[3];
13  int count;
```

Figura 39: Declaración librerías y variables

En la función setup() se inicializa la comunicación serial a una velocidad de 115200 baudios utilizando Serial.begin() lo que permite la comunicación entre el Arduino y el monitor serial de la computadora para imprimir mensajes o datos. Se llama a radio.openReadingPipe(1, pipe) para establecer la dirección de la tubería de lectura del módulo NRF24L01 lo que indica desde qué dirección se recibirán los datos a través de la radio y se llama a radio.startListening() para habilitar la escucha de datos en la tubería configurada.

```
12  float data[3];
13  int count;
14  void setup()
15  {
16    radio.begin();
17    Serial.begin(115200);
18    radio.openReadingPipe(1, pipe);
19    radio.startListening();
20  }
```

Figura 40: Inicialización de comunicaciones

Por último, en la función loop() se leen los datos emitidos por el emisor y se imprimen en pantalla.

```
22  void loop()
23  {
24    if (radio.available())
25    {
26    radio.read(data, sizeof data);
27    Serial.print(data[0],1);   Serial.print(",");
28    Serial.print(data[1],1);  Serial.print(",");
29    Serial.println(data[2],1);
30    }
31    delay(10);
32  }
```

Figura 41: Impresión por pantalla de resultados

8.2 Programación en Processing

Processing se usó para poder guardar los datos que proporciona el dispositivo y generar gráficas. Al igual que en los códigos de Arduino, en un primer momento se incluyen las librerías necesarias.

- processing.serial.* → Permite establecer la comunicación con el puerto serial. Proporciona la clase Serial que se utiliza para interactuar con dispositivos conectados al puerto serial, como el Arduino. Permite leer y escribir datos a través del puerto serial.
- processing.data.Table → Proporciona la funcionalidad para trabajar con datos en forma de tablas. La clase Table se utiliza para crear una tabla donde se pueden almacenar y manipular datos estructurados. Permite agregar columnas, filas y acceder a los datos almacenados en la tabla.
- processing.data.TableRow → Es una extensión de la anterior librería, sirve para obtener una mejor interfaz y poder trabajar mejor con la radio.

Estas bibliotecas vienen todas ya instaladas en el programa por defecto.

Y se declaran las variables que se utilizan a continuación.

```
// Definmos las bibliotecas necesarias
import processing.serial.*;
import processing.data.Table;
import processing.data.TableRow;

//Declaramos las variables
Serial myPort;
int xPos = 1;

int H = 1350;
int V = 720;

Table table;

int contador = 1;
```

Figura 42: Declaración librerías y variables

Ya en el bucle setup() se define la pantalla y el número de frames, además de seleccionar el puerto serial que esté conectado al Arduino.

```
void setup() {
  //Definir el tamaño de la pantalla y el número de frames
  size(1350, 720);
  frameRate(200);

  //Seleccionar el puerto al que esté conectado el arduino
  myPort = new Serial(this, Serial.list()[0], 115200);
  myPort.bufferUntil('\n');

  background(0);
  int a = V / 36;
  fill(250);
```

Figura 43: Definir tamaño pantalla

A continuación, se dibujan las líneas de la cuadrícula y las etiquetas de la gráfica y se crean las tablas en donde más adelante se guardarán los datos.

```
  //Dibujar líneas de la cuadrícula y etiquetas
  for (int i = 0; i <= 36; i = i + 1) {
    stroke(255, 0, 0);
    line(0, a * i, H, a * i);
    text(180 - 10 * i, 2, a * i + 10);
  }

  // Crear la tabla y agregar columnas
  table = new Table();
  table.addColumn("Rotacion en x");
  table.addColumn("Rotacion en y");
  table.addColumn("Rotacion en z");
}
```

Figura 44: Dibujar fondo de la gráfica

En el bucle draw() se leen los datos transmitidos por el puerto serial y se escribe la leyenda de la gráfica.

```
void draw() {
  // Leer datos del puerto serial
  String inString = myPort.readStringUntil('\n');

  // Dibujar rectángulos y etiquetas para los ejes de rotación
  fill(0);
  stroke(255, 0, 0);
  rect(1170, 15, 150, 70);

  textSize(20);
  fill(127, 34, 255);
  text("Rotación en x", 1200, 35);
  stroke(127, 34, 255);
  rect(1180, 23.5, 12, 12);

  fill(0, 255, 0);
  text("Rotación en y", 1200, 55);
  stroke(0, 255, 0);
  rect(1180, 43.5, 12, 12);

  fill(0, 0, 255);
  text("Rotación en z", 1200, 75);
  stroke(0, 0, 255);
  rect(1180, 63.5, 12, 12);

  textSize(13);
  stroke(255, 0, 0);
  fill(250);
```

Figura 45: Dibujar leyenda de la gráfica

A continuación, se procesa la cadena recibida del Arduino y se dibujan los valores en la gráfica a medida que se van recibiendo.

```
 // Leer los datos y dibujar líneas de la gráfica
  if (inString != null) {
    // Procesar la cadena recibida del arduino
    String trimString = inString.substring(0, inString.length() - 2);
    String[] values = splitTokens(trimString, ",");

    if ((values[0] != null) && (values[1] != null) && (values[2] != null)) {

      // Mapear los valores de rotación a la escala de la pantalla
      float inByte1 = map(Float.parseFloat(values[0]), -180, 180, 0, height);
      float inByte2 = map(Float.parseFloat(values[1]), -180, 180, 0, height);
      float inByte3 = map(Float.parseFloat(values[2]), -180, 180, 0, height);

      // Dibujar las líneas de la gráfica
      stroke(127, 34, 255);
      line(xPos, -1 * (inByte1 - height), xPos, -1 * (inByte1 + 4 - height));

      stroke(0, 255, 0);
      line(xPos, -1 * (inByte2 - height), xPos, -1 * (inByte2 + 4 - height));

      stroke(0, 0, 255);
      line(xPos, -1 * (inByte3 - height), xPos, -1 * (inByte3 + 4 - height));

      println(values[0], values[1], values[2]);

```

Figura 46: Se dibujan valores en la gráfica

Una vez se va dibujando la gráfica, se van guardando los valores en un archivo .csv mediante el comando saveTable().

```
      // Agregar los valores a la tabla
      TableRow newRow = table.addRow();
      newRow.setString("Rotacion en x", values[0]);
      newRow.setString("Rotacion en y", values[1]);
      newRow.setString("Rotacion en z", values[2]);

      // Guardar valores en un archivo CSV
      saveTable(table, "resultados/08-07-2023/13-30/datos.csv");
```

Figura 47: Se guardan valores en un .csv

Para finalizar, cada vez que se termina la gráfica, ésta se guarda mediante el comando saveFrame() y se empieza a dibujar otra gráfica, y así hasta que se cierra el programa.

```
107       if (xPos >= width) {
108         // Guardar la gráfica
109         saveFrame("resultados/08-07-2023/13-30/grafica/grafica-"+ contador +".png");
110         contador++;
111         xPos = 0;
112         background(0);
113         int a = V / 36;
114 
115         for (int i = 0; i <= 36; i = i + 1) {
116           stroke(255, 0, 0);
117           line(0, a * i, H, a * i);
118           text(180 - 10 * i, 2, a * i + 10);
119         }
120 
121       } else {
122         xPos++;
123       }
124     }
125   }
126 }
```

Figura 48: Se guarda la gráfica

9 Fabricación de la caja

9.1 Diseño de la caja

El diseño de la caja se realizó en el programa FreeCad, el cual es muy sencillo de usar. Para diseñarla primero se realizó un croquis, teniendo en cuenta las medidas de los componentes y en qué lugar se iban a ubicar.

Una vez se dibujó el croquis se diseñó la caja en el programa mediante cubos y operaciones booleanas para juntar todo.

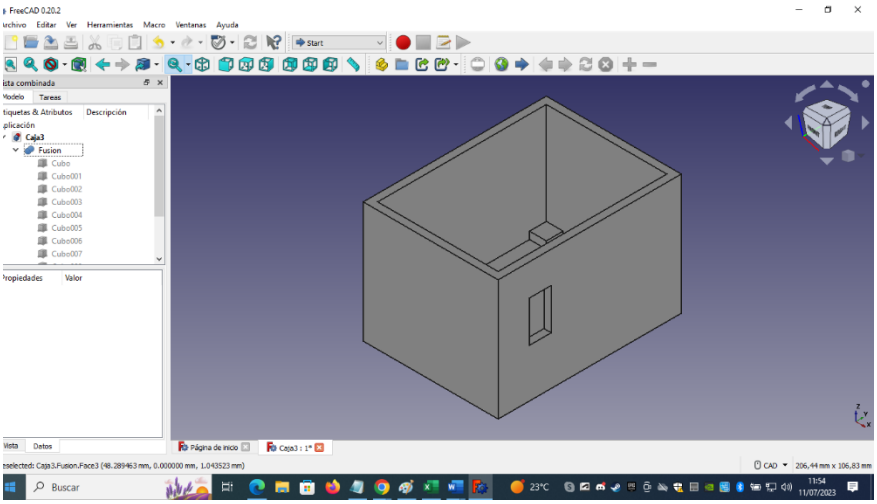

Figura 49: Diseño de caja

9.2 La impresora

La impresora utilizada para la impresión de la pieza fue la Witbox 2. La impresora 3D Witbox 2 es la segunda generación de impresoras de la marca española BQ. Puede imprimir con un volumen de impresión máximo de 297x210x200mm a partir de filamentos de plástico PLA.

Con su cámara, la Witbox 2 fue diseñada especialmente para colegios o entornos domésticos. El extrusor se calibra automáticamente y ha mejorado muchísimo con respecto a la primera generación de Witbox. Funciona con el software Cura y viene con un kit de mantenimiento y una bobina PLA 750g.

9.3 Material utilizado

El material elegido para la caja es el PLA (ácido poliláctico). Este material es un poliéster biodegradable y bioactivo, se deriva de materias primas naturales y renovables, como el maíz, y pertenece a los poliésteres como un polímero sintético. El almidón (glucosa) se extrae de las plantas y se convierte en dextrosa mediante la adición de enzimas. Esto es fermentado por microorganismos en ácido láctico, que a su vez se convierte en polilactida. La polimerización se produce con cadenas moleculares, similares en sus propiedades a los polímeros a base de petróleo.

Figura 50: Cables de PLA

Las principales características de este material son:

- Facilidad de impresión muy alta
- No desprende olor al imprimir
- Temperatura de extrusión 200 a 240 ºC
- Densidad 1.210-1.430 g/cm3
- Resistencia a tracción 3309 MPa
- Resistencia al impacto muy baja
- Temperatura de deformación 55ºC
- Resistencia UVA y humedad muy baja
- Buena reciclabilidad

9.4 Configuración de la impresora

Para imprimir el diseño que se ha realizado se utilizó el programa Ultimaker Cura, en el cual se introducía nuestro diseño en formato STL, se le introducían nuestras preferencias de impresión y se guardó en la tarjeta SD de la impresora.

Una vez en la tarjeta SD se introducía en la impresora y se mandaba a imprimir, el proceso de impresión duró unas cuatro horas.

10 Resultados obtenidos

Con todo el prototipo ya montado y finalizado, se realizaron varias pruebas para ver si funciona correctamente.

Los resultados, como se comentó anteriormente vienen definidos de dos formas diferentes. Todos los ángulos obtenidos se escriben en un archivo .csv, en grados.

Un archivo CSV (Comma-Separated Values) es un tipo de archivo de texto utilizado para almacenar datos estructurados en forma de tabla. En un archivo CSV, cada línea representa una fila de la tabla y los valores de cada columna están separados por comas. Es un formato simple y ampliamente utilizado que permite la interoperabilidad entre diferentes sistemas y aplicaciones. Estos archivos son especialmente útiles cuando se trata de grandes cantidades de datos, como listas de contactos, registros financieros o datos científicos, ya que pueden ser fácilmente procesados por programas de hojas de cálculo o bases de datos. Además, los archivos CSV son legibles tanto por humanos como por máquinas, lo que los convierte en una opción popular para el intercambio de información entre diferentes plataformas.

El otro tipo de resultado es una gráfica que se va creando de forma simultánea en la que se va moviendo el prototipo. En esta gráfica se puede apreciar de manera sencilla el ángulo en X, Y, Z que existe en ese momento.

A continuación veremos un ejemplo, en la que se estuvo un tiempo con el prototipo en la cabeza, moviéndolo en diferentes ángulos.

Una vez finalizada la prueba se guardan todos los datos en un archivo .csv, y con Excel se crea una tabla donde se pueden observar todos los ángulos obtenidos.

1	Rotacion en x	Rotacion en y	Rotacion en z
2	0,2	0	0
3	0,4	0,1	0
4	0,6	0,1	0
5	0,8	0,1	0
6	1	0,2	0
7	1,1	0,2	0
8	1,3	0,2	0
9	1,5	0,2	0
10	1,7	0,3	0
11	1,8	0,2	0
12	2	0,2	0
13	2,1	0,2	0
14	2,3	0,2	-0,1
15	2,4	0,2	-0,1
16	2,6	0,2	-0,1
17	2,7	0,2	-0,1
18	2,9	0,1	-0,2
19	3	0,1	-0,2
20	3,1	0,1	-0,2
21	3,2	0,1	-0,2

datos

1	Rotacion en x	Rotacion en y	Rotacion en z
965	4,8	34,4	-9,7
966	4,9	33,6	-9,7
967	5	32,8	-9,7
968	5,2	32,1	-9,6
969	5,4	31,3	-9,6
970	5,5	30,5	-9,6
971	5,7	29,7	-9,6
972	5,9	29	-9,6
973	6,1	28,3	-9,6
974	6,2	27,6	-9,6
975	6,3	26,9	-9,6
976	6,4	26,2	-9,6
977	6,5	25,5	-9,6
978	6,6	24,9	-9,6
979	6,7	24,2	-9,6
980	6,7	23,5	-9,6
981	6,7	22,9	-9,6
982	6,8	22,2	-9,6
983	6,8	21,6	-9,6
984	6,8	20,9	-9,6

datos

Figura 51: Datos en Excel

Teniendo los valores en el archivo .csv, también podemos observar las gráficas generadas. En la que se expone a continuación vemos claramente cómo la cabeza

realiza un movimiento de arriba abajo, como si estuviera diciendo si con la cabeza. Este movimiento lo vemos en la gráfica pintada de verde, la rotación en y:

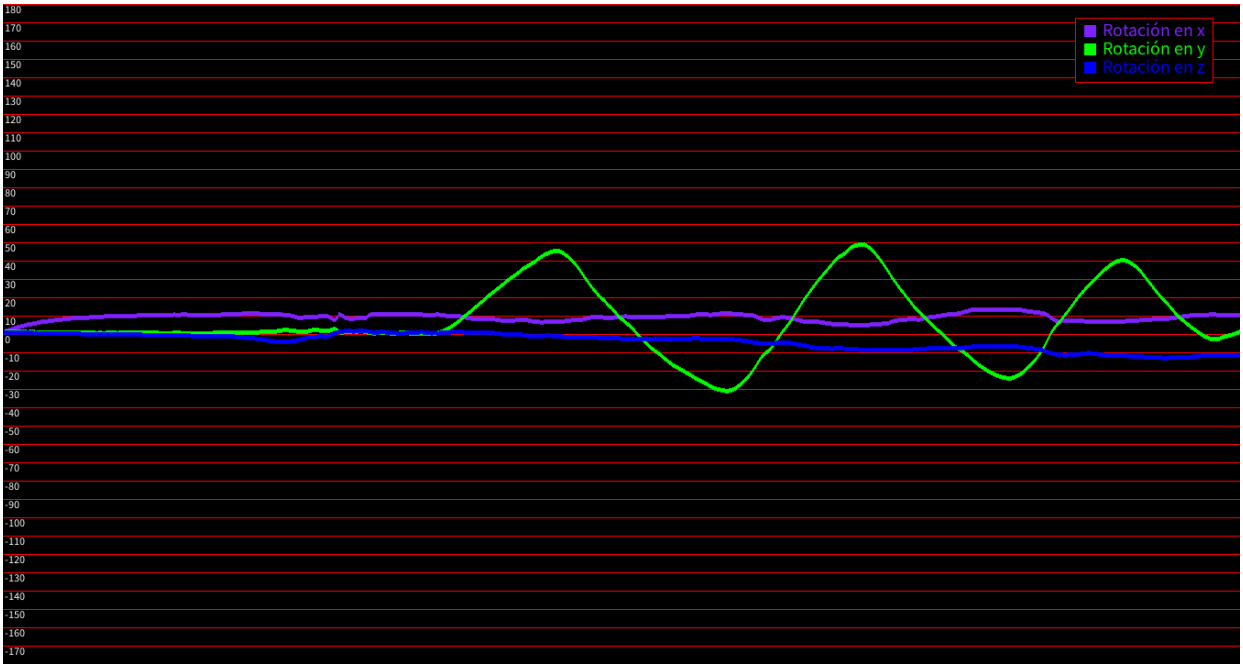

Figura 52: Gráfica 1

En la siguiente gráfica generada se observa el movimiento de la cabeza con la rotación en x esta vez:

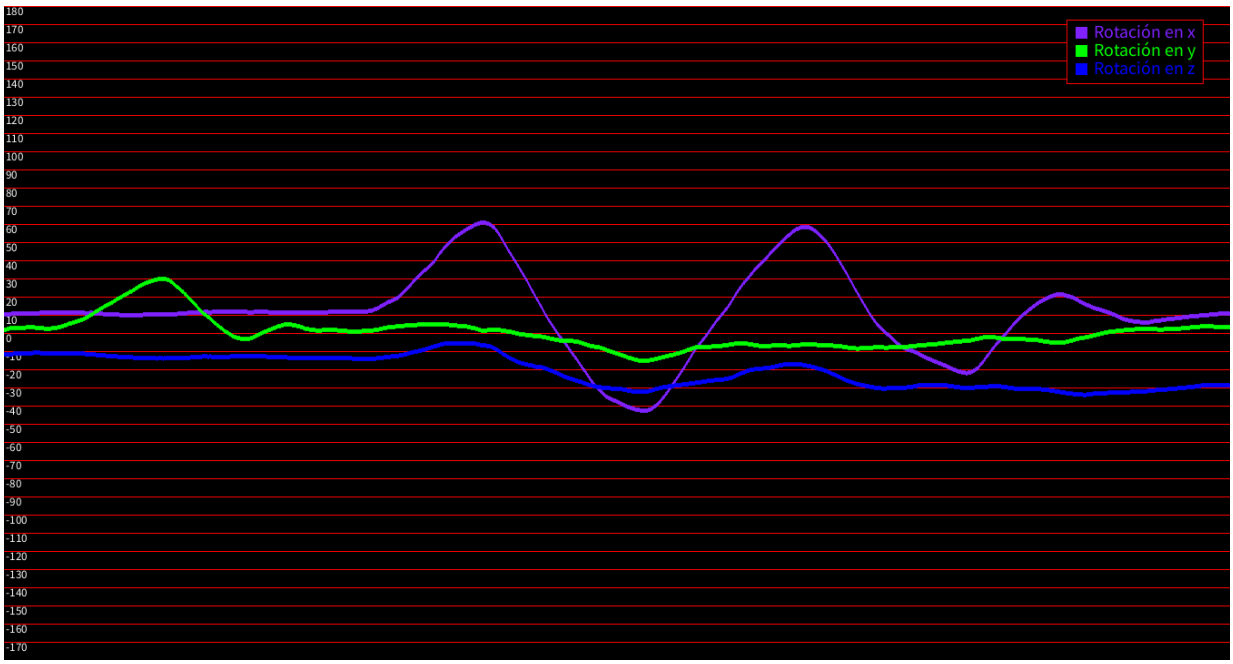

Figura 53: Gráfica 2

Y, por último, observamos la rotación en z, que se produce cuando se mira de un lado a otro:

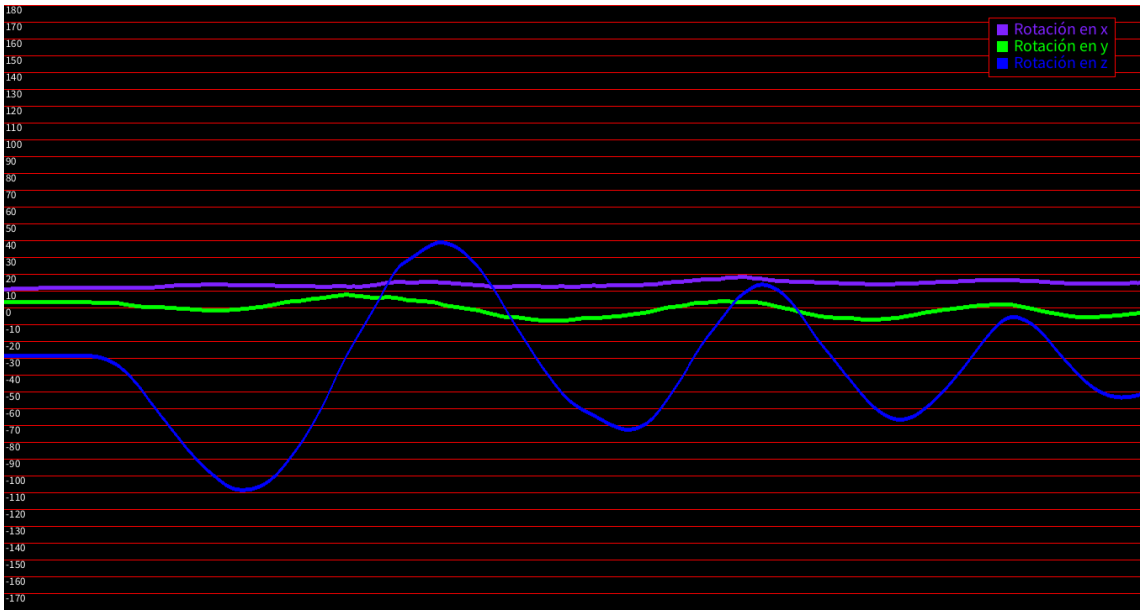

Figura 54: Gráfica 3

Tras realizar varias pruebas como esta, se determinó que el prototipo funcionaba de manera correcta, comprobando que los valores obtenidos de los ángulos medidos no tenían una precisión del 100%, existiendo un margen de error residual, lo cual no afecta a la función esencial que pretende dar nuestro prototipo y que es la detección del movimiento de la cabeza.

Otra prueba que se realizó para verificar el alcance de nuestro prototipo fue la de observar a cuantos metros funcionaba el dispositivo alejado del ordenador. En primer lugar, se realizó en un lugar abierto y se comprobó que funcionaba hasta unos 15 metros.

Posteriormente se comprobó su correcto funcionamiento en un recinto cerrado, dando como resultado, que la señal ha podido ser captada en el interior de una vivienda de aproximadamente 100 m^2.

Cabe destacar que cada vez que se enciende nuevamente el prototipo, transcurre un tiempo de unos 15 segundos en los que tiene que permanecer quieto y sobre una superficie horizontal para que se calibre de manera correcta y pueda comenzar a medir desde la posición 0,0,0.

11 Problemas y soluciones

A continuación se mencionan, a modo de resumen, los problemas planteados durante la realización del proyecto, así como las actuaciones llevadas a cabo para resolverlos:

1) Tras la programación inicial del sensor MPU6050, se obtuvieron datos relativos a ángulos de rotación, de tal forma que, al encontrarse el sensor en posición horizontal, no se obtenía el valor de 0º que le correspondería en esa posición. Dicho problema se solventó tras proceder a su calibración modificando la programación.
2) Después de conectar el emisor con el receptor, se detectó que no se transmitían correctamente los datos de uno a otro. El motivo era el orden de conexión. Primero se debía conectar el receptor y a continuación encender el emisor. Una vez comprobado, no volvió a suceder.
3) A la hora de leer los primeros datos de rotación en Processing, saltaba un error ocasionado por el tamaño de la cadena de caracteres, lo cual se subsanó limitando la longitud del valor leído.

12 Presupuesto

12.1 Presupuesto del prototipo

A continuación se detallan los importes para el diseño, fabricación y puesta en marcha del prototipo, ascendiendo a un coste total de 545,10 €.

Presupuesto del prototipo				
Cantidad	Ud	Descripción	Precio unitario/hora	Precio total
1	Ud	Arduino Uno	24,00 €	24,00 €
1	Ud	Arduino Nano	21,60 €	21,60 €
1	Ud	MPU6050	3,50 €	3,50 €
2	Ud	NRF24L01	6,00 €	12,00 €
1	Ud	Batería 5V con regulador	7,50 €	7,50 €
1	Ud	Caja de PVC	5,00 €	5,00 €
1	Ud	Placa de soldadura	0,50 €	0,50 €
5	horas	Montaje prototipo	15,00 €	75,00 €
15	horas	Programación	20,00 €	300,00 €
3	horas	Pruebas de funcionamiento	12,00 €	36,00 €
3	horas	Diseño 3D	20,00 €	60,00 €
			Total	545,10 €

11.2 Presupuesto del resto de unidades

A continuación se detallan los importes para la fabricación y puesta en marcha del resto de unidades, ascendiendo a un coste de 116,10 €/ud, ya que en esta fase se eliminan las tareas de programación y diseño 3D.

Presupuesto del resto de unidades				
Cantidad	Ud	Descripción	Precio unitario/hora	Precio total
1	Ud	Arduino Uno	24,00 €	24,00 €
1	Ud	Arduino Nano	21,60 €	21,60 €
1	Ud	MPU6050	3,50 €	3,50 €
2	Ud	NRF24L01	6,00 €	12,00 €
1	Ud	Batería 5V con regulador	7,50 €	7,50 €
1	Ud	Caja de PVC	5,00 €	5,00 €
1	Ud	Placa de soldadura	0,50 €	0,50 €
2	horas	Montaje	15,00 €	30,00 €
1	horas	Pruebas de funcionamiento	12,00 €	12,00 €
			Total	116,10 €

13 Conclusiones

El principal objetivo de este trabajo ha sido el diseño de un sensor de movimiento de la cabeza para niños con dificultad motora y las principales conclusiones son las siguientes:

1. El desarrollo de este trabajo me ha permitido aplicar los conocimientos teóricos y prácticos adquiridos en mi formación como Grado en Ingeniería Electrónica Industrial y Automática. Aunque la principal dificultad inicialmente encontrada ha sido el desconocimiento sobre el funcionamiento y programación del Arduino, la búsqueda de información sobre este tipo de componente electrónico y su aplicación práctica me ha permitido adquirir nuevos conocimientos en esta materia.
2. Se ha conseguido el diseño de un prototipo de bajo coste, utilizando herramientas informáticas gratuitas. Esto me ha permitido incrementar mi destreza en tareas de programación y de dibujo asistido por ordenador.
3. Una vez conectados todos los componentes, se ha comprobado el funcionamiento del prototipo de manera experimental. Aunque en ocasiones el emisor no transmitía correctamente los datos al receptor, dicha incidencia se subsanó alterando el orden de conexión de los elementos.
4. Los resultados obtenidos se registran y visualizan en la pantalla un ordenador, lo cual permite comprobar en tiempo real el correcto funcionamiento del prototipo.
5. Se ha obtenido como resultado un prototipo compacto, de pequeñas dimensiones y fácilmente manejable, que permita su posterior colocación sobre la cabeza del niño/a. Ello se ha conseguido mediante la utilización de un Arduino NANO, de menores dimensiones que el Arduino UNO, ubicándose el conjunto de componentes en el interior de una caja de pequeñas dimensiones, de forma ordenada. La fabricación de la caja mediante impresora 3D me ha permitido familiarizarme con este tipo de tecnología, de amplia aplicación en el campo de la ingeniería.
6. El resultado permite su divulgación y utilización en el ámbito académico, personal o profesional. El presente proyecto ha alcanzado satisfactoriamente los objetivos propuestos inicialmente, y se ha desarrollado exponiendo la información de forma clara, ordenada y fácilmente comprensible.
7. Una vez confirmado el correcto funcionamiento del prototipo, quedan las puertas abiertas a la investigación y el desarrollo, para conseguir mejoras técnicas y/o mejores prestaciones: reducción de su tamaño y peso, aumento del alcance entre el emisor y el receptor, capacidad de lectura de mayor número de variables, etc.

14 Conclusions

The main objective of this work has been the design of a head movement sensor for children with motor difficulties and the main conclusions are the following:

1. The development of this work has allowed me to apply the theoretical and practical knowledge acquired in my training as a Degree in Industrial Electronics and Automatic Engineering. Although the main difficulty initially encountered has been the lack of knowledge about the operation and programming of the Arduino, the search for information about this type of electronic component and its practical application has allowed me to acquire new knowledge in this matter.
2. The design of a low-cost prototype has been achieved, using free computer tools. This has allowed me to increase my skills in computer aided drawing and programming tasks.
3. Once all the components are connected, the operation of the prototype has been verified experimentally. Although sometimes the sender did not correctly transmit the data to the receiver, this incident was corrected by altering the connection order of the elements.
4. The results obtained are recorded and displayed on a computer screen, which makes it possible to check the correct functioning of the prototype in real time.
5. As a result, a compact prototype has been obtained, small in size and easily manageable, which allows its subsequent placement on the child's head. This has been achieved by using an Arduino NANO, smaller than the Arduino UNO, placing the set of components inside a small box, in an orderly manner. The manufacture of the box using a 3D printer has allowed me to become familiar with this type of technology, which is widely applied in the field of engineering.
6. The result allows its dissemination and use in the academic, personal or professional field. This project has satisfactorily achieved the initially proposed objectives, and has been developed exposing the information in a clear, orderly and easily understandable manner.

15 Bibliografía

[1] https://fundacionadecco.org/blog/que-es-la-discapacidad-motora/#:~:text=Seg%C3%BAn%20la%20Organizaci%C3%B3n%20Mundial%20de,del%20movimiento%20y%20la%20postura%C2%BB

[2] https://www.unir.net/educacion/revista/discapacidad-motora-en-el-aula/

[3] https://www.isesinstituto.com/noticia/la-vida-escolar-del-nino-con-discapacidad-motora#:~:text=El%20alumnado%20con%20deficiencia%20motora,dificultad%20para%20la%20comunicaci%C3%B3n%20oral.

[4] https://formainfancia.com/discapacidad-motriz-tipos-causas/

[5] https://www.ine.es/prensa/edad_2020_p.pdf

[6] https://www.heraldo.es/noticias/salud/2022/04/26/causas-aumento-discapacidad-ninos-adolescentes-espana-1568981.html

[7] https://www.unicef.org/es/comunicados-prensa/casi-240-millones-ninos-con-discapacidad-mundo-segun-analisis-estadistico

[8] https://www.un.org/sustainabledevelopment/es/development-agenda/

[9]https://www.luisllamas.es/medir-la-inclinacion-imu-arduino-filtro-complementario/

[10] https://www.blascarr.com/?lessons=imu-raw-data

[11] https://www.pngitem.com/middle/hRhbhm_kinect-joint-rotation-pitch-yaw-roll-unity-hd/

[12] https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

[13] https://store.arduino.cc/products/arduino-uno-rev3

[14] https://www.redgps.com/dispositivos-iot/arduino-nano#:~:text=El%20Arduino%20Nano%20es%20un,en%20lugar%20de%20uno%20est%C3%A1ndar.

[15] https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html

[16] https://www.prometec.net/usando-el-mpu6050/

[17] https://naylampmechatronics.com/blog/16_tutorial-basico-nrf24l01-con-arduino.html

[18] https://tienda.bricogeek.com/varios/906-transceptor-inalambrico-nrf24l01-24ghz.html

[19]https://repository.javeriana.edu.co/bitstream/handle/10554/21433/PerezValderramaJorgeIvan2016.pdf?sequence=1

[20] https://robotlandia.es/adaptadores-y-cables/481-cable-dupont-10-cm.html

[21] https://solectroshop.com/es/blog/soldadura-primeros-pasos-para-soldar-componentes-electronicos-y-uso-del-flux-n70

[22] https://aprendiendoarduino.wordpress.com/2016/12/11/ide-arduino/

[23] https://mosaic.uoc.edu/2012/04/30/introduccion-a-processing/

[24] https://www.3dnatives.com/es/freecad-modelador-3d-codigo-abierto-200520202/

[25] https://www.arduino.cc/reference/en/language/functions/communication/wire/

[26] https://github.com/nRF24/RF24

[27] http://www.datasheet.es/PDF/829184/NRF24L01-pdf.html

[28] http://www.datasheet.es/PDF/735134/MPU6050-pdf.html#google_vignette

ANEXO I Códigos utilizados

Código de Arduino del emisor

```
#include <Wire.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <SPI.h>

long loopTimer, loopTimer2;
int temperature;
double accelPitch;
double accelRoll;
long acc_x, acc_y, acc_z;
double accel_x, accel_y, accel_z;
double gyroRoll, gyroPitch, gyroYaw;
int gyro_x, gyro_y, gyro_z;
long gyro_x_cal, gyro_y_cal, gyro_z_cal;
double rotation_x, rotation_y, rotation_z;
double freq, dt;
double tau = 0.98;
double roll = 0;
double pitch = 0;
S
float ang[3];

// 250 deg/s --> 131.0, 500 deg/s --> 65.5, 1000 deg/s --> 32.8, 2000
deg/s --> 16.4
long scaleFactorGyro = 65.5;

// 2g --> 16384 , 4g --> 8192 , 8g --> 4096, 16g --> 2048
long scaleFactorAccel = 8192;

const int pinCE = 9;
const int pinCSN = 10;
RF24 radio(pinCE, pinCSN);

// Single radio pipe address for the 2 nodes to communicate.
const uint64_t pipe = 0xE8E8F0F0E1LL;

void setup() {
  // Start
  Wire.begin();
  Serial.begin(115200);
  radio.begin();              //Iniciando radio
  radio.openWritingPipe(pipe);

  // Setup the registers of the MPU-6050 and start up
  setup_mpu_6050_registers();
```

```
  // Calibration
  //Serial.println("Calibrando, colóquelo en una superficie nivelada y no
lo mueva.");

  // Take 3000 readings for each coordinate and then find average offset
  for (int cal_int = 0; cal_int < 3000; cal_int ++){
    if(cal_int % 200 == 0)Serial.print(".");
    read_mpu_6050_data();
    gyro_x_cal += gyro_x;
    gyro_y_cal += gyro_y;
    gyro_z_cal += gyro_z;

    delay(3);
  }

  // Average the values
  gyro_x_cal /= 3000;
  gyro_y_cal /= 3000;
  gyro_z_cal /= 3000;

  // Reset the loop timer
  loopTimer = micros();
  loopTimer2 = micros();
}


void loop() {
  freq = 1/((micros() - loopTimer2) * 1e-6);
  loopTimer2 = micros();
  dt = 1/freq;

  // Read the raw acc data from MPU-6050
  read_mpu_6050_data();

  // Subtract the offset calibration value
  gyro_x -= gyro_x_cal;
  gyro_y -= gyro_y_cal;
  gyro_z -= gyro_z_cal;

  // Convert to instantaneous degrees per second
  rotation_x = (double)gyro_x / (double)scaleFactorGyro;
  rotation_y = (double)gyro_y / (double)scaleFactorGyro;
  rotation_z = (double)gyro_z / (double)scaleFactorGyro;

  // Convert to g force
  accel_x = (double)acc_x / (double)scaleFactorAccel;
  accel_y = (double)acc_y / (double)scaleFactorAccel;
  accel_z = (double)acc_z / (double)scaleFactorAccel;
```

```
  // Complementary filter
  accelPitch = atan2(accel_y, accel_z) * RAD_TO_DEG;
  accelRoll = atan2(accel_x, accel_z) * RAD_TO_DEG;

  pitch = (tau)*(pitch + rotation_x*dt) + (1-tau)*(accelPitch);
  roll = (tau)*(roll - rotation_y*dt) + (1-tau)*(accelRoll);

  gyroPitch += rotation_x*dt;
  gyroRoll -= rotation_y*dt;
  gyroYaw += rotation_z*dt;

  ang[0] = roll;
  ang[1] = pitch;
  ang[2] = gyroYaw;

  radio.write(ang, sizeof ang);
  delay(10);
  // Data out serial monitor
    //Serial.print("Rotacion en X= " );
    Serial.print(ang[0]);
    Serial.print(",");
    //Serial.print(" Rotacion en Y= " );
    Serial.print(ang[1]);
    Serial.print(",");
    ///Serial.print(" Rotacion en z= " );
    Serial.println(ang[2]);

  // Wait until the loopTimer reaches 4000us (250Hz) before next loop
  while (micros() - loopTimer <= 4000);
  loopTimer = micros();
}


void read_mpu_6050_data() {
  // Subroutine for reading the raw data
  Wire.beginTransmission(0x68);
  Wire.write(0x3B);
  Wire.endTransmission();
  Wire.requestFrom(0x68, 14);

  // Read data --> Temperature falls between acc and gyro registers
  acc_x = Wire.read() << 8 | Wire.read();
  acc_y = Wire.read() << 8 | Wire.read();
  acc_z = Wire.read() << 8 | Wire.read();
  temperature = Wire.read() <<8 | Wire.read();
  gyro_x = Wire.read()<<8 | Wire.read();
  gyro_y = Wire.read()<<8 | Wire.read();
  gyro_z = Wire.read()<<8 | Wire.read();
```

```
}


void setup_mpu_6050_registers() {
  //Activate the MPU-6050
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();

  // Configure the accelerometer
  // Wire.write(0x__);
  // Wire.write; 2g --> 0x00, 4g --> 0x08, 8g --> 0x10, 16g --> 0x18
  Wire.beginTransmission(0x68);
  Wire.write(0x1C);
  Wire.write(0x08);
  Wire.endTransmission();

  // Configure the gyro
  // Wire.write(0x__);
  // 250 deg/s --> 0x00, 500 deg/s --> 0x08, 1000 deg/s --> 0x10, 2000
deg/s --> 0x18
  Wire.beginTransmission(0x68);
  Wire.write(0x1B);
  Wire.write(0x08);
  Wire.endTransmission();
}
```

Código de Arduino del receptor

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

const int pinCE = 9;
const int pinCSN = 10;
RF24 radio(pinCE, pinCSN);

// Single radio pipe address for the 2 nodes to communicate.
const uint64_t pipe = 0xE8E8F0F0E1LL;

float data[3];
int count;
void setup()
{
  radio.begin();
  Serial.begin(115200);
  radio.openReadingPipe(1, pipe);
  radio.startListening();
}

void loop()
{
  if (radio.available())
  {
  radio.read(data, sizeof data);
  Serial.print(data[0],1);   Serial.print(",");
  Serial.print(data[1],1);  Serial.print(",");
  Serial.println(data[2],1);
  }
  delay(10);
}
```

Código de Processing

```
// Definmos las bibliotecas necesarias
import processing.serial.*;
import processing.data.Table;
import processing.data.TableRow;

//Declaramos las variables
Serial myPort;
int xPos = 1;

int H = 1350;
int V = 720;

Table;
int contador = 1;

void setup() {
  //Definir el tamaño de la pantalla y el número de frames
  size(1350, 720);
  frameRate(200);

  //Seleccionar el puerto al que esté conectado el arduino
  myPort = new Serial(this, Serial.list()[0], 115200);
  myPort.bufferUntil('\n');

  background(0);
  int a = V / 36;
  fill(250);

  //Dibujar líneas de la cuadrícula y etiquetas
  for (int i = 0; i <= 36; i = i + 1) {
    stroke(255, 0, 0);
    line(0, a * i, H, a * i);
    text(180 - 10 * i, 2, a * i + 10);
  }

  // Crear la tabla y agregar columnas
  table = new Table();
  table.addColumn("Rotacion en x");
  table.addColumn("Rotacion en y");
  table.addColumn("Rotacion en z");
}

void draw() {
  // Leer datos del puerto serial
  String inString = myPort.readStringUntil('\n');

  // Dibujar rectángulos y etiquetas para los ejes de rotación
```

```
  fill(0);
  stroke(255, 0, 0);
  rect(1170, 15, 150, 70);

  textSize(20);
  fill(127, 34, 255);
  text("Rotación en x", 1200, 35);
  stroke(127, 34, 255);
  rect(1180, 23.5, 12, 12);

  fill(0, 255, 0);
  text("Rotación en y", 1200, 55);
  stroke(0, 255, 0);
  rect(1180, 43.5, 12, 12);

  fill(0, 0, 255);
  text("Rotación en z", 1200, 75);
  stroke(0, 0, 255);
  rect(1180, 63.5, 12, 12);

  textSize(13);
  stroke(255, 0, 0);
  fill(250);

 // Leer los datos y dibujar líneas de la gráfica
  if (inString != null) {
    // Procesar la cadena recibida del arduino
    String trimString = inString.substring(0, inString.length() - 2);
    String[] values = splitTokens(trimString, ",");

    if ((values[0] != null) && (values[1] != null) && (values[2] !=
null)) {

      // Mapear los valores de rotación a la escala de la pantalla
      float inByte1 = map(Float.parseFloat(values[0]), -180, 180, 0,
height);
      float inByte2 = map(Float.parseFloat(values[1]), -180, 180, 0,
height);
      float inByte3 = map(Float.parseFloat(values[2]), -180, 180, 0,
height);

      // Dibujar las líneas de la gráfica
      stroke(127, 34, 255);
      line(xPos, -1 * (inByte1 - height), xPos, -1 * (inByte1 + 4 -
height));

      stroke(0, 255, 0);
      line(xPos, -1 * (inByte2 - height), xPos, -1 * (inByte2 + 4 -
height));
```

```
      stroke(0, 0, 255);
      line(xPos, -1 * (inByte3 - height), xPos, -1 * (inByte3 + 4 -
height));

      println(values[0], values[1], values[2]);

      // Agregar los valores a la tabla
      TableRow newRow = table.addRow();
      newRow.setString("Rotacion en x", values[0]);
      newRow.setString("Rotacion en y", values[1]);
      newRow.setString("Rotacion en z", values[2]);

      // Guardar valores en un archivo CSV
      saveTable(table, "resultados/08-07-2023/13-30/datos.csv");

      if (xPos >= width) {
        // Guardar la gráfica
        saveFrame("resultados/08-07-2023/13-30/grafica/grafica-"+
contador +".png");
        contador++;
        xPos = 0;
        background(0);
        int a = V / 36;

        for (int i = 0; i <= 36; i = i + 1) {
          stroke(255, 0, 0);
          line(0, a * i, H, a * i);
          text(180 - 10 * i, 2, a * i + 10);
        }

      } else {
        xPos++;
      }
    }
  }
}
```

ANEXO II Datasheets de componentes

CONTENTS

nRF24L01
Single Chip 2.4GHz Transceiver

Product Specification

Key Features

- Worldwide 2.4GHz ISM band operation
- Up to 2Mbps on air data rate
- Ultra low power operation
- 11.3mA TX at 0dBm output power
- 12.3mA RX at 2Mbps air data rate
- 900nA in power down
- 22µA in standby-I
- On chip voltage regulator
- 1.9 to 3.6V supply range
- Enhanced ShockBurst™
- Automatic packet handling
- Auto packet transaction handling
- 6 data pipe MultiCeiver™
- Air compatible with nRF2401A, 02, E1 and E2
- Low cost BOM
- ±60ppm 16MHz crystal
- 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

Applications

- Wireless PC Peripherals
- Mouse, keyboards and remotes
- 3-in-one desktop bundles
- Advanced Media center remote controls
- VoIP headsets
- Game controllers
- Sports watches and sensors
- RF remote controls for consumer electronics
- Home and commercial automation
- Ultra low power sensor networks
- Active RFID
- Asset tracing systems
- Toys

July 2007

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-9V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) (0.5 KB used by bootloader)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz