PRÁCTICA 5 – SENSOR DE ORIENTACIÓN (IMU)

En esta práctica se experimentará con datos proporcionados por un sensor de orientación. En concreto, con el acelerómetro y el giroscopio que se incluyen dentro de la popular IMU (*Interntial measurement Unit*) integrada MPU 6050 de InvenSense-TDK. El desarrollo de la práctica consistirá en desarrollar, primero un programa para Arduino que filtre y procese los datos del sensor para obtener las medidas de orientación como ángulos de Tait-Bryan, y segundo, un sencillo programa de Matlab que represente en una gráfica la evolución de los ángulos durante un tiempo. Opcionalmente, se podrá completar el programa de Matlab para que muestre una representación gráfica 3D de la IMU con su orientación.

Para abordar los experimentos, es muy recomendable repasar el contenido de las diapositivas de teoría del tema 8 sobre sensores de orientación, que sirve como manual básico para entender cómo funcionan y se utilizan los sensores de orientación.

1. Conceptos básicos

1.1. Sensores de orientación: giroscopios y acelerómetros integrados

Un sensor de orientación permite obtener la orientación de un objeto en un sistema 2D o 3D. Existen diversos tipos de sensores de orientación, entre los que destacan el giróscopo y el acelerómetro. El desarrollo y abaratamiento de los dispositivos MEMS (*Micro-Electro-Mechanical Systems*) ha provocado que actualmente se comercialicen multitud de circuitos integrados que incluyen dentro sensores de orientación MEMS, y estos sensores de orientación se han extendido mucho en los últimos años en aplicaciones de ingeniería y especialmente en la robótica. Así, hay muchos dispositivos integrados y circuitos impresos compactos que incluyen múltiples sensores de orientación MEMS, de forma que pueden medir diferentes magnitudes físicas a partir de las que se pude obtener información de orientación y posicionamiento relativos. Estos dispositivos se conocen como IMUs (*Inertial Measurement Unit*) y proporcionan medidas como la velocidad angular, las aceleraciones o fuerzas gravitacionales, y los campos magnéticos a los que están sometidos.

Cada tipo de medida se suele proporcionar para dos o tres ejes de movimiento en un sistema de coordenadas 2D o 3D respectivamente, y así una IMU se caracteriza por el sus DOF (*Degrees Of Freedom*) que representa el número de medidas totales que proporciona para todos sus ejes en una misma lectura. En esta práctica se utilizará un circuito integrado de bajo coste que conforma una IMU con un giroscopio y un acelerómetro de 6DOF; 3 medidas de velocidad para el giroscopio en lo ejes de un sistema 3D, y 3 medidas de aceleración para el acelerómetro.

Si se quiere representar la posición de un objeto en el espacio 3D, se requieren 3 coordenadas para determinar la posición respecto a un origen, y otras 3 coordenadas para determinar la orientación o actitud (attitude) con respecto a un sistema de coordenadas de referencia. Para la posición se suele usar coordenadas cartesianas, mientras que para la orientación hay diversas opciones comunes: matrices de rotación, ángulos de Euler, ángulos de Tait-Bryan y cuaterniones. En esta práctica de laboratorio se utilizará la representación mediante ángulos de Tait-Bryan (Figura 1), que es una variación de los ángulos de Euler más intuitiva, y además es muy empleada en navegación con vehículos o naves por su sencillez.

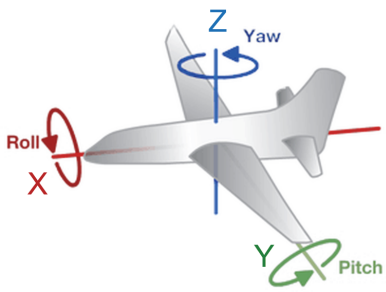


Figura 1. Orientación de un objeto expresada con los ángulos de Tait-Bryan.

Los ángulos de Tait-Bryan definen 3 rotaciones ortogonales conmutativas:

- *Roll* (alabeo, escora), en torno al eje X.
- *Pitch* o *tilt* (elevación, cabeceo, inclinación), en torno al eje Y
- *Yaw* (dirección, guiñada, deriva), en torno al eje Z.

Aunque el concepto y el uso de los ángulos Tait-Bryan son sencillos, éstos presentan un problema de singularidad en algunos casos, en los que, tras el giro de uno de los ángulos, un eje del sistema de coordenadas coincide con otro, de modo que ambos ejes no sean diferenciables y solo sea posible obtener una representación en dos dimensiones. Por ejemplo, los ejes X y Z coindicen cuando el eje Y gira ±90º, y entonces X y Z representan el mismo ángulo y se pierde un grado de libertad.

Un giroscopio es básicamente un dispositivo mecánico que permite medir su cambio de orientación en diversos ejes. En el caso de los giroscopios MEMS, se mide la velocidad angular (rad/s o grados/s) a la que está sometida cada eje del dispositivo, en lugar de una orientación absoluta como ángulos. Por eso se requiere realizar una integración numérica para obtener los ángulos de giro a partir de las velocidades angulares, como se muestra en los ejemplos de la Figura 2.

Integración simple:

$$\theta_{Xn-1} = \theta_{Xn-1} + \omega_{Xn} dt$$

```
dt = (millis() - tiempoPrev) / 1000.0;
tiempoPrev = millis();
dt *= 0.5; // Divide by 2
aX_roll += gyro_Wx * dt;
aY_pitch += gyro_Wy * dt;
aZ_yaw += gyro_Wz * dt;
```

Integración trapezoidal:

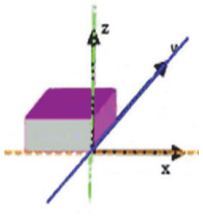
$$\theta_{Xn} = \theta_{Xn} + \frac{(\omega_{Xn-1} + \omega_{Xn})}{2} dt$$

```
dt = (millis() - tiempoPrev) / 1000.0;
tiempoPrev = millis();
dt *= 0.5; // Divide by 2
aX_roll += (gyroPrev_Wx + gyro_Wx) * dt;
aY_pitch += (gyroPrev_Wy + gyro_Wy) * dt;
aZ_yaw += (gyroPrev_Wz + gyro_Wz) * dt;
gyroPrev_Wx= gyro_Wx; gyroPrev_Wy= gyro_Wy;
gyroPrev_Wz= gyro_Wz;
```

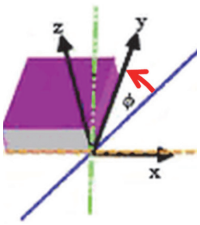
Figura 2. Métodos básicos de integración numérica.

La integración genera un problema: provoca una deriva o desviación de la medida base de orientación por culpa de que los errores también se van acumulando con la integración. No obstante, los giroscopios permiten medir con mucha precisión cambios rápidos de rotación en los ejes.

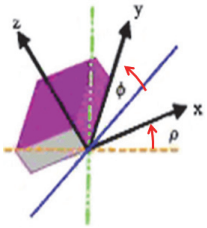
En contraste con el giroscopio, el acelerómetro es un tipo de sensor más estable a lo largo del tiempo. Un acelerómetro mide la aceleración (m/s^2) a la que está sometido cada uno de sus ejes. Para usarlo como sensor de orientación, se considera que el acelerómetro solo está sometido a la aceleración que produce la fuerza de la gravedad (g=9,81m/s^2), y se determina la orientación mediante trigonometría a partir de las aceleraciones medidas, que corresponden con componentes del vector de gravedad. La Figura 3 muestra algunos casos de ejemplo de los ángulos de rotación, así como las expresiones trigonométricas simplificadas para el cálculo de los ángulos de rotación de Tait-Bryan. Hay que prestar atención a los signos de los numeradores en las expresiones, que dependen de si se desea obtener un ángulo de inclinación o de rotación.



Posición de referencia, sin rotación, con Z alineado con g (A_Z=-g, A_X=A_Y=0)



Rotación en X (A_X=0, A_Y=-g_Y, A_Z=-g_Z)



Rotación en X e Y (A_X=-g_X, A_Y=-g_Y, A_Z=-g_Z)

$$Pitch\ (Y):\ \rho = arctan\left(\frac{-A_X}{\sqrt{A_Y^2 + A_Z^2}}\right)$$

$$Roll\ (X):\ \phi = arctan\left(\frac{A_Y}{\sqrt{A_X^2 + A_Z^2}}\right)$$

$$Yaw\ (Z):\ \theta = arctan\left(\frac{\sqrt{A_X^2 + A_Y^2}}{A_Z}\right)$$

Figura 3. Obtención de los ángulos de Tait-Bryan a partir de las medidas de un acelerómetro, con ángulos acordes a la MPU-6050 como muestra la Figura 6.

Por su funcionamiento, si un eje del acelerómetro está orientado en la dirección vertical de la gravedad, no se puede obtener su giro en ese ángulo. Por ejemplo, si se observa la representación de la Figura 1, que representa los ángulos de *roll*, *pitch* y *yaw*, cuando el eje Z de la nave está alineado con la gravedad, solo resulta posible medir los ángulos de *roll* y *pitch*, los cuales serán 0. Para obtener el ángulo de *yaw* habrá que hacer uso de la información adicional de un giroscopio.

A continuación se muestra un ejemplo de código C++ que calcula los ángulos de *roll* y *pith* en grados:

```
float accel_ang_x_roll = atan(ay/sqrt(pow(ax,2) + pow(az,2)))*(180.0/3.14);
float accel_ang_y_pith = atan(-ax/sqrt(pow(ay,2) + pow(az,2)))*(180.0/3.14);
```

Los acelerómetros MEMS suelen ser dispositivos muy sensibles a las vibraciones y movimientos rápidos, por lo que las medidas que proporcionan presentan un ruido de alta frecuencia y no son fiables con cambios rápidos.

1.2. Filtrado y fusión de las medidas

Para aprovechar las ventajas que ofrecen un giroscopio y un acelerómetro, a la vez que trata de superar sus inconvenientes, se suelen filtrar y combinar las medidas que proporcionan ambos sensores.

Así, primero se suele filtrar las medidas de ambos, de forma que solo se tomen las mejores medidas de cada uno, y después se combinan las medidas para los mismos ejes para lograr una mayor exactitud. Mientras que para el acelerómetro se emplea un filtro digital paso-bajo, que elimina el ruido de la medida debido a los movimientos rápidos, para el giroscopio se usa un filtro paso-alto, que elimina el error de deriva acumulado y solo deja pasar las variaciones rápidas de movimiento. Para implementar los filtros digitales y la combinación de las medidas de un acelerómetro y un giroscopio, el algoritmo más utilizado es el del filtro complementario. La Figura 4 muestra un ejemplo del comportamiento típico que tiene el valor de un ángulo de la orientación obtenido de las medidas de un acelerómetro, de un giroscopio y como resultado de un usar filtro complementario con ambas.

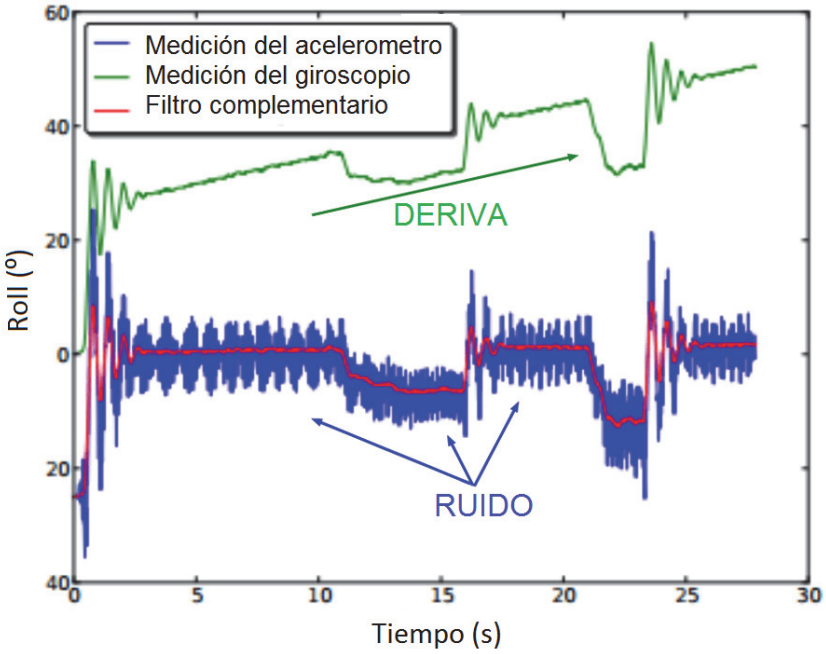
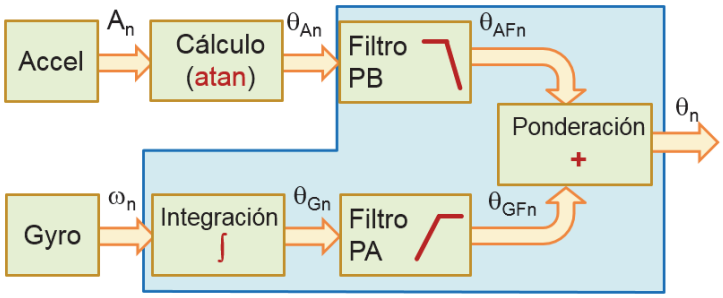


Figura 4. Ejemplo de comparación del valor de un ángulo de la orientación obtenido de las medidas de un acelerómetro, de un giroscopio y como resultado de un usar filtro complementario con ambas.

El filtro complementario incluye los filtros digitales paso-bajo para el acelerómetro y paso-alto para el giroscopio, y además lleva a cabo la combinación ponderada de las medidas de los dos dispositivos. Todo ello con operaciones simples, que consumen pocos recursos y se pueden programar en microcontroladores pequeños, como son muchos de los usados con Arduino. En algoritmo consiste en realizar los cálculos representados en el esquema de la Figura 5 para cada medida de cada eje.

Las variables implicadas en el filtro complementario de la Figura 5 son las siguientes:

- θ_n es el valor angular que se está calculando para un eje.
- θ_{n-1} es el valor angular obtenido en la ejecución previa del algoritmo.
- ω_n es la velocidad angular obtenida del giroscopio en la ejecución actual para el eje.
- θ_{An} es la aceleración obtenida del acelerómetro en la ejecución actual.
- *dt* es el periodo de muestreo, es decir, el tiempo entre medidas y ejecuciones consecutivas del filtro. Conviene programar el algoritmo para utilizar un valor de *dt* constante, de forma que α sea constante.
- α representa la constante del filtro que determina en qué grado se consideran las medidas del acelerómetro o las del giroscopio. El valor de α debe ser de 0 a 1, y depende del periodo de muestreo *dt* y de la constante de tiempo τ del filtro.
- τ es la constante de tiempo del filtro, y suele tener un valor entre 0,5 segundos y 1 segundo, que conviene ajustar experimentalmente en función de la calidad de las medidas del ángulo obtenidas.



$$\theta_n = \alpha(\theta_{n-1} + \omega_n dt) + (1-\alpha)\theta_{An}$$

```
ang_x = alfa*(ang_x + gyro_vel_x*dt) +
    +(1-alfa)*accel_ang_x
```

$$\alpha = \frac{\tau}{\tau + dt}$$

Figura 5. Esquema y algoritmo del filtro complementario.

1.3. Calibración inicial

Antes de iniciar el proceso de adquisición de medidas con un acelerómetro o un giroscopio, conviene realizar una calibración inicial con la IMU en una posición de origen. El resultado de la calibración se usa después para disminuir los errores de medida que se producen en la IMU durante su operación normal, debido principalmente a imperfecciones tanto en el montaje del circuito integrado de la IMU sobre la placa de circuito impreso como en la ubicación del circuito impreso sobre la plataforma o robot donde se usa.

En muchas aplicaciones basta con una calibración sencilla, que simplemente determine la desviación (*bias* u *offset*) que hay en los datos brutos que proporcionan los sensores cuando estos están en la posición de origen, cunado en teoría los datos deberían corresponder con unos ángulos de orientación nulos en la posición de origen. Para ello, cuando el programa de Arduino arranque, y antes de empezar a tomar y procesar las medidas de orientación, puede capturar los datos que proporcionan el acelerómetro y el giroscopio durante un tiempo para sus diferentes ejes, y calcular los valores promedio en cada eje, que se utilizarán como un vector de *offsets*.

Para que la calibración sea útil, ésta se debe realizar mientras la IMU permanece estable en la posición que se considera como origen o referencia de los movimientos. Por eso, conviene que el programa avise al usuario (con un LED, un mensaje enviado por el puerto serie, o un texto pintado en una pantalla) y espere una acción de éste (como pulsar un botón) antes de empezar el proceso de calibración. También es conveniente que el programa avise de cuando acaba la calibración.

En la práctica, se puede considerar como posición de referencia la que tiene la IMU cuando está horizontal respecto al suelo, con la gravedad afectando solo al eje Z.

Cuando el programa de Arduino pase al proceso de medición, cada vez que lee los valores de los ejes de los sensores de la IMU, deberá restar a esos valores la desviación correspondiente calculada antes. Normalmente, las bibliotecas que facilitan la programación de una IMU (ver apartado 1.5) ofrecen una función que permite establecer los valores de *offsets* para cada sensor al acabar la calibración, y unas funciones para obtener los valores de los ejes de cada sensor que ya realizan la diferencia con los *offsets* internamente.

1.4. IMU MPU-6050

En esta práctica se trabajará con datos obtenidos del giroscopio y acelerómetro incluidos en el circuito integrado MPU-6050 fabricado por la marca InvenSense-TDK. Este circuito integrado es una IMU de 6 seis grados de libertad o 6DOF (*Degrees Of Freedo*m), que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes, y es una IMUs muy empleada en robots y naves de bajo coste de bajo coste por su buena relación calidad-precio.

La comunicación con esta IMU puede realizarse tanto con SPI como mediante el bus I2C, de forma que los datos de las medidas de aceleración y velocidad angular se obtienen en el microcontrolador de forma digital directamente. Internamente la IMU utiliza un ADC (*Analoge to Digital Converter*) de 16 bits. Además, también es posible programar parámetros de la IMU y configurar su modo de funcionamiento.

El MPU-6050 incorpora un procesador interno llamado DMP (*Digital Motion Processor*) que ejecuta algoritmos de filtrado y fusión de medidas de los sensores internos, evitando tener que programar filtros digitales en el microcontrolador que lee las medidas. Pero esta función no se utilizará en la práctica, ya que un objetivo de la misma es aprender a programar los algoritmos.

Además del acelerómetro, del giroscopio y del DMP, el MPU-6050 dispone de un sensor de la temperatura del integrado, que se puede usar para compensar el error que generan las variaciones de temperatura sobre las medidas, un reloj de alta precisión, salidas para interrupciones programables al microcontrolador, y un puerto de comunicación I2C adicional para conectarse como maestro a otros dispositivos (como un receptor GNSS). En esta práctica de laboratorio no hará falta usar estas características.

La medida de cada eje de cada sensor se obtiene como un valor entero de 16 bits con signo, esto es, un dato en el rango -32.768 a 32.767. El rango de medidas correspondientes al rango de datos puede ser ajustado entre las siguientes opciones, en función de las necesidades de precisión y tiempo de respuesta:

- Acelerómetro: ±2g, ±4g, ±8g o ±16g (por defecto está configurado el rango ±2g), con g=9,81m/s^2.
- Giroscopio: ±250°/s, ±500°/s, ±1.000°/s, o ±.2000°/s (por defecto está configurado el rango ±250°/s).

La tensión de alimentación del MPU-6050 debe estar en el rango 2.4 a 3.6V, usándose típicamente 3,3V, por lo que se requiere un regulador de tensión si se va a conectar a un microcontrolador de 5V. El circuito integrado consume 3.5mA, con todos los sensores y el DMP activados.

En montajes de prototipos con Arduino, no se suele emplear directamente un circuito integrado MPU-6050, ya que su pequeño encapsulado SMD (*Surface-Mount Technology*) requiere de herramientas especiales para la soldadura de sus terminales. En su lugar se suele utilizar una placa de prototipado o *breadboard*, la cual ya tiene soldados, además del circuito integrado de la IMU, los otros componentes necesarios, como es el caso de la popular placa GY-521 mostrada en la Figura 6. Esta placa también incluye un regulador de tensión de 3,3V, con lo que se puede alimentar directamente desde una fuente de 5V, las resistencias de *pull-up* para el bus I2C, varios condensadores de desacoplo, y un LED que indica cuando la placa está alimentada.

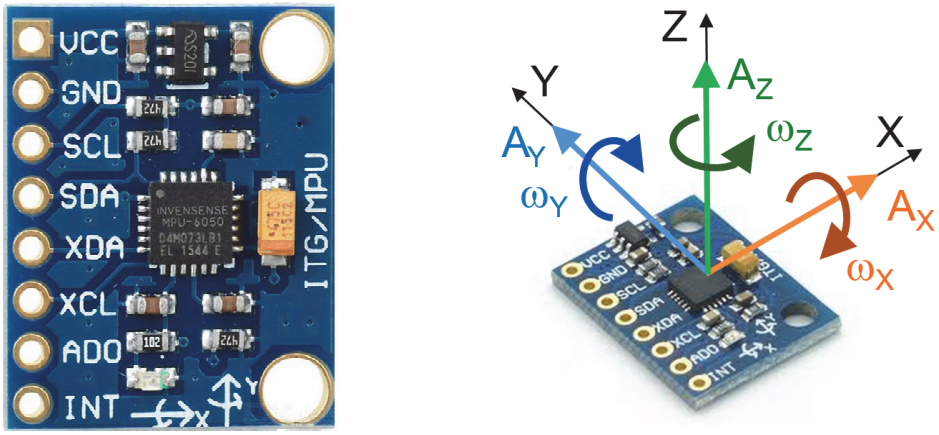


Figura 6. Placa GY-521 con la IMU MPU-6050, y sus ejes de referencia.

Las conexiones de la placa GY-521 son las siguientes:

- **VCC**: Entrada de alimentación de 3V a 5V.
- **GND**: Referencia común o 0V de la alimentación.
- **SCL**: Línea de reloj del bus I2C que comunica el MPU-6050 con el microcontrolador.
- **SDA**: Línea de datos del bus I2C que comunica el MPU-6050 con el microcontrolador.
- **XDA**: Línea de datos del bus I2C que comunica el MPU-6050 con otro dispositivo I2C esclavo.
- **XCL**: Línea de reloj del bus I2C que comunica el MPU-6050 con otro dispositivo I2C esclavo.
- **AD0**: Permite seleccionar la dirección de la IMU como dispositivo esclavo de I2C. Cuando AD0 está nivel alto o 3,3V, la IMU se accede con la dirección hexadecimal 0x69. Si el pin se pone a nivel bajo conectándolo a GND, o se deja sin conectar, la IMU utiliza la dirección 0x68. Esto permite conectar dos MPU-6050 al mismo bus I2C.
- **INT**. Salida que puede ser conectada a una interrupción del microcontrolador para que éste último sepa cuando hay nuevas medidas disponibles.

La conexión de la IMU MPU-6050 a un Arduino Uno es bastante sencilla cuando se utiliza una placa como la GY-510, y basta con las conexiones básicas para la alimentación y los datos serie de I2C, como muestra la Figura 7. Cabe recordar que, en una placa de Arduino Uno, las conexiones para el bus I2C están disponibles en dos lugares: en los pines SDA y SCL de la parte superior derecha cerca del conector USB, y en los pines A4 para SDA y A5 para SCL. No hace falta añadir las resistencias de *pull-up* para las señales SCL y SDA porque estás ya están integradas en la placa GY-510.

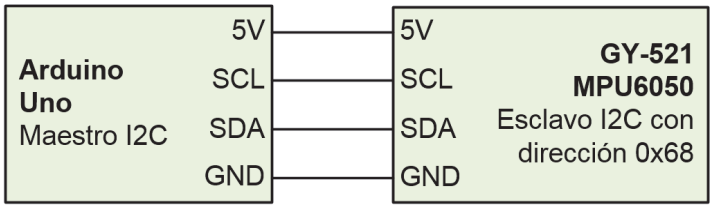


Figura 7. Conexión básica entre un Arduino Uno y una placa GY-521 con la IMU MPU-6050 mediante I2C.

Aunque aún resulta fácil adquirir la IMU MPU-6050 y placas de prototipo GY-521, la marca InvenSense-TDK comercializa actualmente unas versiones mejoradas, denominadas MPU-9150 y MPU-9250, las cuales también incluyen un magnetómetro además del giroscopio y del acelerómetro, y tienen errores de desviación y deriva menores. También es fácil encontrar placas de prototipo con las nuevas IMUs por unos 4€.

1.5. Programación de la IMU MPU-6050 en Arduino

Para poder acceder a la IMU MPU-6080 desde un programa de Arduino, existen diversas alternativas, pero la más aconsejable es sin duda utilizar una biblioteca que encapsule todas las tareas de la comunicación I2C, y de acceso a los registros del circuito integrado MPU-6080, en un objeto de C++. Esto facilita mucho el desarrollo y ayuda a evitar errores de programación. Es fácil encontrar bibliotecas para sensores habituales gracias a proyectos de código libre compartidos en Internet. Para esta práctica, se recomienda utilizar la biblioteca "I2Cdevlib" de Jeff Rowberg. Ésta es una biblioteca muy completa y popular, compatible con Arduino y muchos otros microcontroladores, que proporciona acceso a muchos sensores y dispositivos que usan I2C, incluida la IMU MPU-6050. Se puede encontrar mucha información en Internet sobre cómo usar la IMU MPU-6050 con la biblioteca "I2Cdevlib".

A continuación se muestran algunos enlaces con ejemplos que hacen uso de la biblioteca "I2Cdevlib" y que pueden ser muy útiles para llevar a cabo la práctica:

- Determinar la orientación con Arduino y el IMU MPU-6050, de Luis Llamas:

  https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050

- Tutorial MPU6050, Acelerómetro y Giroscopio, de Naylamp Mechatronics:

  https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html

- Gyroscopes and Accelerometers on a Chip, de Geek Mom Projects:

  http://www.geekmomprojects.com/gyroscopes-and-accelerometers-on-a-chip

- Página principal de la biblioteca "i2cdevib" de Jeff Rowberg, y repositorio de "i2cdevib" en Github:

  https://www.i2cdevlib.com
  https://github.com/jrowberg/i2cdevlib

Un programa básico de Arduino que haga uso de la IMU para obtener los ángulos de orientación se pude estructurar en tres etapas:

- En la función *setup*, iniciar las comunicaciones serie e I2C, así como la IMU. Para la práctica, la IMU se debe inicializar con los rangos de medida por defecto: ±2g para el acelerómetro, y ±250°/s para el giroscopio.
- Realizar una calibración de la IMU, antes de empezar a tomar medidas. Conviene incluir el código que realiza la calibración en una función, la cual se puede llamar desde la función *setup*, o en la primera iteración de la función *loop*. La calibración de la IMU se debe hacer colocando está en la posición de origen o referencia, y manteniéndola estable. Es habitual usar una posición de origen horizontal, de forma que la aceleración de la gravedad captada para los ejes X, Y y Z del acelerómetro sea AX=0g, AX=0, y AZ=-g respectivamente (ver la Figura 6). Conviene enviar por el puerto serie cadenas de texto que avisen sobre el inicio y el final del proceso de calibración.
- Después del proceso de calibración, y dentro de la función *loop*, iniciar el proceso de medida de la orientación. Este proceso debe repetir las siguientes operaciones cada cierto intervalo de tiempo (periodo de muestreo):
  - Leer los datos "brutos" del acelerómetro y del giroscopio usando las funciones adecuadas de biblioteca. Estos datos son valores enteros de 16 bits sin procesar en rango -32.768 a 32.767,
  - Escalar los datos las medidas adecuadas: m/s^2 para el acelerómetro, y °/s para el giroscopio. Para ello basta multiplicar los datos por las constantes adecuadas que transforman un valor en rango de 16 bits a los rangos con los que se inicializa la IMU.
  - A continuación se puede calcular los ángulos de orientación de Tait-Bryan por separado según las medidas del giroscopio y del acelerómetro (ver apartado 1.1), o calcular los ángulos mediante fusión de las medidas de ambos sensores (ver apartado 1.2). En cualquier caso, para la integración hay que tener en cuenta que *dt* es el periodo de muestreo usado, Además, para facilitar los cálculos, en la práctica se pueden limitar los movimientos a ángulos de Tait-Bryan en el rango de -90° a 90°.
  - Enviar los ángulos obtenidos por el puerto serie, formateados como una cadena de texto. Conviene usar una velocidad de comunicación serie alta como 115.200bps.

  El periodo de muestreo debe ser constante, y para esta práctica se considerará un periodo de 50ms. Para ejecutar las operaciones anteriores cada periodo de muestreo dentro de la función *loop* se puede utilizar la función *millis*.
- El proceso de medida debe finalizar después de haber trascurrido 10s, o tras haber realizado 200 medidas de la orientación.

Para esta práctica, las medidas de orientación que el programa envía por el puerto serie cada periodo de muestreo deben seguir el siguiente formato como cadenas de texto, de modo que resulte fácil interpretarlas con el monitor serie de un ordenador, así como guardarlas en archivo de texto con formato CSV (*Comma Separated Values*):

- Una primera línea con los nombres de los datos de las siguientes líneas:
  - "T, RollA, PithA, RollG, PithG". Para la actividad 1.
  - "T, RollA, PithA, RollG, PithG, RollFC, PithFC". Para la actividad 2.
- Cada una del resto líneas contiene una muestra formada por 7 o 10 valores separados por comas:

- T: instante de tiempo en qué se tomó la medida, como un número entero de milisegundos. El tiempo se cuenta desde la primera medida, a la que corresponde el instante 0.
- RollA, PithA: Ángulos *roll* y *pith* de Tait-Bryan, obtenidos a partir del acelerómetro, y expresados como un número real de grados con un decimal.
- RollG, PithG: Ángulos *roll* y *pith* de Tait-Bryan, obtenidos a partir del giroscopio, y expresados como un número real de grados con un decimal.
- RollFC, PithFC: Ángulos *roll* y *pith* de Tait-Bryan, obtenidos al aplicar el filtro complementario para combinar las medidas del acelerómetro y el giroscopio, y expresados como un número real de grados con un decimal. Estos valores no se consideran en la actividad 1.

2. Experimentos a realizar

2.1. Actividad 1: Lectura y procesamiento de los valores de orientación (6 puntos)

a) Desarrollar un programa para un Arduino uno que utilice la IMU MPU-6050. Este programa, debe llevar a cabo una calibración de la IMU, calcular los ángulos *roll* (eje X) y *pith* (eje Y) de Tait-Bryan a partir del acelerómetro y del giroscopio de forma separa, y enviar los valores obtenidos por el puerto serie como una cadena de texto en formato CSV, según se describe en el apartado 1.5.

b) Conectar la IMU a un Arduino Uno, y depurar y evaluar el programa examinado las cadenas de texto que envía con el monitor serie. Copiar y guardar los resultados de varias pruebas de movimientos como archivos de texto CSV. Conviene planificar un poco los movimientos para las pruebas en vez de que mover arbitrariamente la IMU, por ejemplo, tratando de hacer primero movimientos por separado para los distintos ejes de la IMU, y luego movimiento combinados. Tomar una fotografía del circuito montado para incluirla en el informe

c) Desarrollar y ejecutar un programa de Matlab que coja los datos de un archivo CSV, y muestre gráficamente los valores de los ángulos *roll* y *pith* para el acelerómetro y el giroscopio respecto al tiempo. Fara facilitar el análisis de los resultados, hay que hacer dos gráficas similares a la de la Figura 4, una para cada ángulo de orientación, y en cada gráfica representar los valores del acelerómetro y del giroscopio. Realizar una captura de las gráficas para incluirla en el informe.

d) Responder las siguientes cuestiones en el informe:

- ¿Los ángulos obtenidos del acelerómetro se corresponden con la orientación de la IMU?
- ¿Las medidas son estables o varían mucho cuando se mantiene la IMU quieta?
- ¿Qué ángulos presentan más ruido, los del acelerómetro o los del giroscopio?
- ¿Qué ángulos son más estables a lo largo del tiempo, los del acelerómetro o los del giroscopio?

2.2. Actividad 2: Filtro complementario (3 puntos)

e) Modificar el programa de la actividad anterior para que también filtre y fusione las medidas del acelerómetro y del giroscopio mediante el algoritmo de filtro complementario, con el objetivo de obtener unos valores mejores de los ángulos de orientación *roll* (eje X) y *pith* (eje Y). Hay que ajustar la constante τ del filtro para obtener medidas estables a la vez que se captan variaciones rápidas de orientación, teniendo en cuenta el periodo de muestreo utilizado.

f) Conectar la IMU a un Arduino Uno, y depurar y evaluar el programa examinado las cadenas de texto que envía con el monitor serie. Copiar y guardar los resultados de varias pruebas de movimientos como archivos de texto CSV. Conviene planificar un poco los movimientos para las pruebas en vez de que mover arbitrariamente la IMU, por ejemplo, tratando de hacer primero movimientos por separado para los distintos ejes de la IMU, y luego movimiento combinados.

g) Modificar el programa de Matlab de la actividad anterior para que añada en las gráficas de los ángulos de *roll* y *pith* los valores correspondientes obtenidos del filtro complementario. Realizar una captura de las gráficas para incluirla en el informe.

h) Responder las siguientes cuestiones en el informe:

- ¿Qué valor de constante τ se ha utilizado? ¿Cómo se ha calculado?
- ¿Qué similitud hay entre los ángulos que proporciona el filtro complementario, y los obtenidos directamente del acelerómetro y del giroscopio?
- ¿El filtro consigue eliminar los errores típicos de las medidas del acelerómetro y del giroscopio?
- ¿Los ángulos proporcionados por el filtro siguen fielmente a los obtenidos directamente del acelerómetro y del giroscopio, o se produce algún retardo?

2.3. Actividad 3: tareas complementarias (1 punto)

Se plantean a continuación dos tareas que se pueden desarrollar para subir la nota de la práctica hasta llegar a un máximo de 10. No es necesario desarrollar las dos actividades, y se puede escoger entre una, otra o las dos.

i) Incluir el cálculo del ángulo Tait-Bryan de *yaw* para el eje Z en los programas de Arduino y Matlab de la actividad 2. Estudiar si es buena idea usar las medidas del eje Z del acelerómetro, y analizar problema presenta el uso de las medidas del giroscopio, explicando estas cuestiones en el informe.

j) Mejorar la aplicación en Matlab de forma que utilice la información de orientación proporcionada por el filtro complementario para representar, en una vista 3D, un objeto simple (por ejemplo un prisma rectangular) cuya orientación se vaya actualizando según los ángulos de orientación de la IMU tomados del archivo CSV y considerando los tiempos de las muestras.

   Para esta tarea, se pueden usar funciones de Matlab para gráficas 3D, como plot3 o path (para *parches* o figura de uno o más polígonos), y no hace falta recurrir a bibliotecas o aplicaciones externas. Con una de esas funciones se puede pintar un objeto sencillo, como un cubo, en una figura 3D, y después rotar el objeto en cada eje con la función *rotate* dentro de un bucle que vaya aplicando los ángulos orientaciones.

2.4. Entregas

Cada equipo entregará a través de UACloud un archivo comprimido en formato ZIP que contendrá los siguientes archivos y documentos:

- Para cada una de las actividades completadas, se incluirá una carpeta con los programas fuente de Arduino y de Matlab. Los programas deben estar bien explicados mediante comentarios en los mismos, y deben compilar sin errores y haber sido probados previamente.
- Un breve informe, en formato PDF, donde se incluya las respuestas de las siguientes tareas:
  - b) Fotografía del montaje con el Arduino Uno y la IMU.
  - c) Captura de las gráficas obtenidas con Matlab.
  - d) Respuestas a las preguntas planteadas.
  - g) Captura de las gráficas obtenidas con Matlab.
  - h) Respuestas a las preguntas planteadas.
  - i) Si se ha desarrollado esta tarea, hay que explicar brevemente los problemas que pueden surgir al usar las medidas de *yaw* en eje Z.
  - j) Si se ha desarrollado esta tarea, hay que explicar brevemente como se ha programado la representación 3D, e incluir una captura de la misma.