Informática Industrial

2018

Profesores: N. Acosta, J. Toloza, F. Lagorio

Informe Final

Objetivo del trabajo: obtener un modelo tridimensional en la pantalla de la PC que replique el movimiento traslacional y rotacional físico de dos sensores: un giróscopo y un acelerómetro. Se resumen en este informe todos los resultados obtenidos a lo largo de la cursada.

Tecnologías utilizadas:

- Arduino Uno
- MPU 6050
- Processing 3.3.7
- Java 8
- Diversas librerías

Código fuente del programa:
https://github.com/SynysterLove/InfoIndustrialMPU6050

(Disponible a partir del 19/06/18)

1. Observaciones generales del problema

El problema presentado consiste en construir una simulación gráfica que emule el movimiento de un sensor MPU-6050 real. Para obtener este producto final, se propuso un enfoque en dos etapas:

- Etapa 1: construir una simulación gráfica que replique el movimiento *rotacional* del sensor.

- Etapa 2: agregar a la simulación gráfica de la Etapa 1 la réplica del movimiento *traslacional* del sensor.

La primera etapa pudo resolverse al 100% con base a una aplicación ya existente, conformada por un programa Arduino y un programa Processing. Los resultados fueron muy satisfactorios; el movimiento se replica en la pantalla de manera muy fiel y fluida, y los valores numéricos representativos de la rotación se muestran en pantalla en varios formatos (cuaternión, ángulos de Euler, Yaw-Pitch-Roll).
La segunda etapa presentó varios problemas, de los cuales hubo uno en particular que no fue posible resolver. Se logró el reconocimiento del movimiento traslacional y la consecuente réplica en la pantalla, pero la representación no es fiel debido a una característica inherente al sensor. Esto es, el acelerómetro mide la aceleración indirectamente midiendo *fuerza*. Cuando el sensor es acelerado linealmente, el sensor detecta una cierta fuerza (vectorial, con componente en las tres direcciones) y arroja los valores de aceleración correspondientes. Pero cuando el sensor es *rotado*, el mismo acelerómetro también detecta una *fuerza* que actúa sobre él, y esta fuerza es mucho más intensa que la asociada al movimiento lineal. Por lo tanto, el acelerómetro es mucho más sensible a los movimientos rotacionales que a los traslacionales (siendo estos últimos los de interés cuando se trabaja con dicho sensor). Más formalmente, si calculásemos la relación señal-ruido del acelerómetro obtendríamos siempre valores negativos, ya que es mucho más intenso el ruido introducido que la señal de interés. La solución propuesta, entonces, permite replicar el movimiento lineal del dispositivo *siempre y cuando*

este se encuentre en la misma posición relativa respecto de cualquier eje de rotación.

2. Montaje del dispositivo

En las siguientes figuras (Fig. 2.1 y 2.2) se muestran un diagrama de las conexiones de la placa Arduino y el sensor, y una fotografía real del mismo sistema.

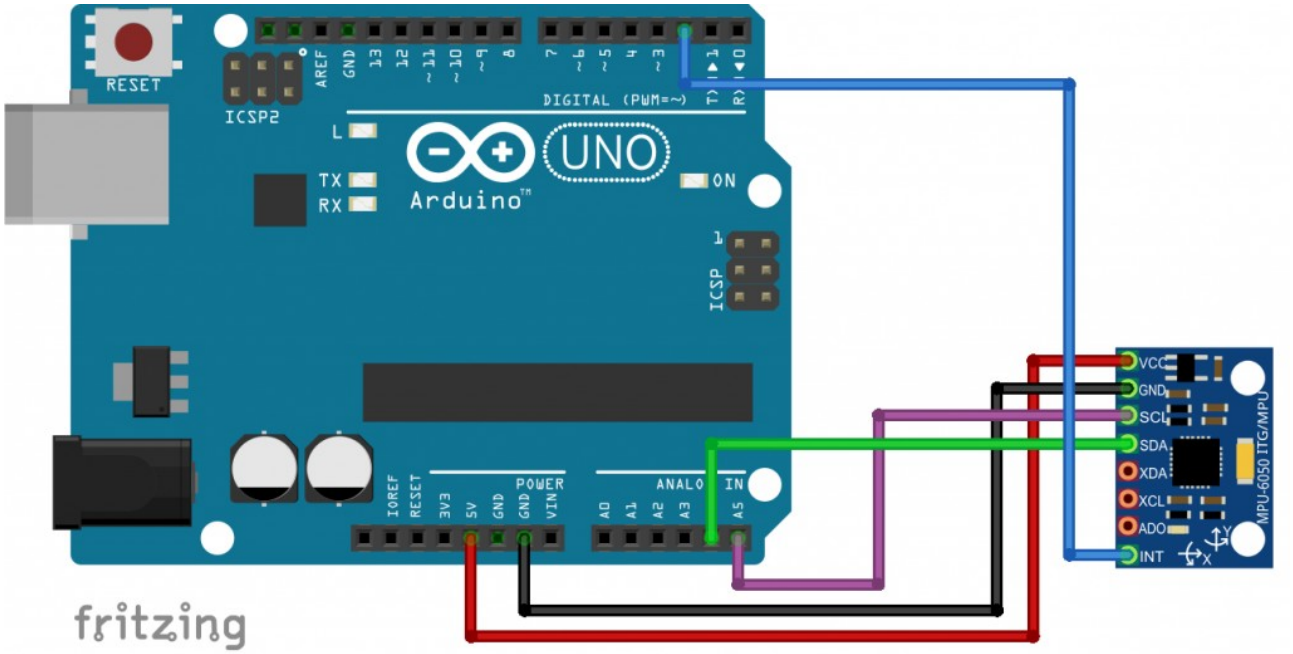

Fig. 2.1 Diagrama de conexión entre placa Arduino y sensor MPU-6050.

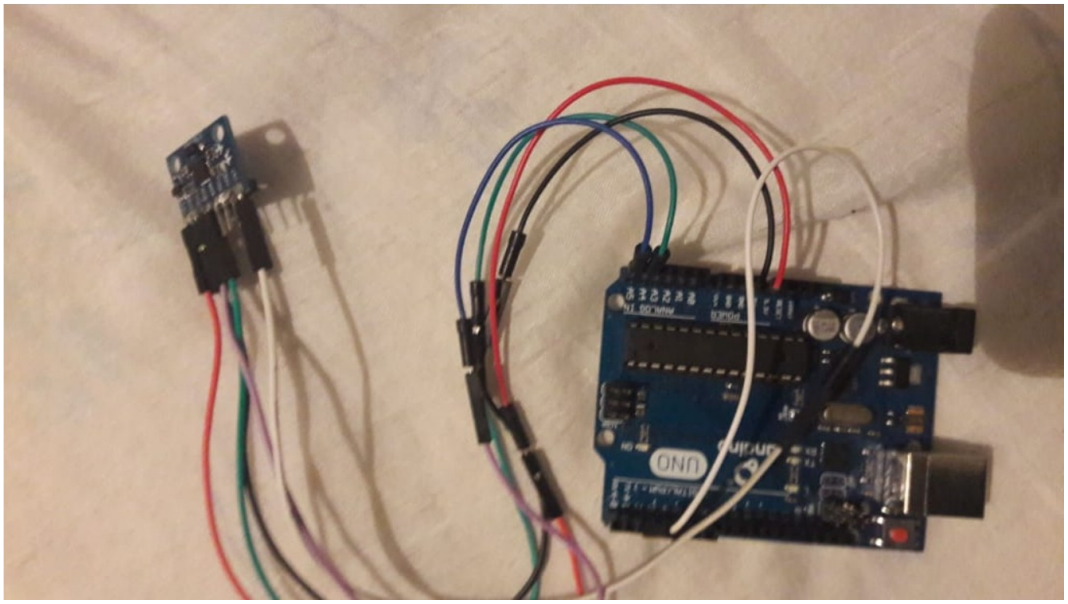

Fig 2.2 Fotografía real que muestra las conexiones de la placa con el sensores

La alimentación del sensor se conecta a la terminal de 5V de la placa; el cable violeta y el verde, que toman los pines SCL y SDA del dispositivo sensor, se conectan a los puertos analógicos A5 y A4 de Arduino; el pin de inicialización se conecta por el cable blanco al pin digital 2. Por último, el cable negro es la conexión a masa.

3. Soluciones exploradas

Para ambos casos, se utilizó una interface gráfica ya construida en Processing, y se trabajó sobre el parser, la parte del programa que se encarga de la interacción Arduino-Processing, así como sobre el programa Arduino. Se especifican a continuación los detalles de las soluciones exploradas para resolver cada problema.

Para resolver el problema de comunicación, se propusieron varias soluciones.

La primera, bastante rudimentaria, consistió en un sencillo programa de Arduino que utilizaba la librería Wire para leer directamente valores del sensor; así, se obtuvieron mediciones de rapidez angular alrededor de los tres ejes y de aceleración lineal en sus direcciones, y con ellas se utilizaron las fórmulas de cinemática del movimiento circular y lineal respectivamente para calcular los desplazamientos.

La segunda solución consistió en utilizar las librerías MPU6050 e I2C, que resuelven el problema de comunicación Arduino-sensor llevando al programador a un nivel de abstracción más alto. El uso de estas librerías permite no solo prescindir de utilizar directamente la librería Wire (cuyos comandos son complicados, de muy bajo nivel y requieren configuración previa), sino que también proveen al usuario una larga serie de funciones muy útiles para tomar y/o modificar los valores del sensor, dando también la posibilidad, entre otras cosas, de ajustar offsets y de obtener valores ya procesados (por ejemplo, los valores del cuaternión asociado a un giro o las componentes del vector aceleración sin el aporte de la gravedad). Esta

solución era mucho mejor que la primera, pero fue parcialmente descartada debido a que el programa original se decantaba por una tercera solución.

Esta última solución consistía en utilizar la misma base del programa original para Arduino, que no utilizaba las mencionadas librerías sino que en su lugar definía directamente una gran variedad de funciones y direcciones de memoria a muy bajo nivel. Para la primer implementación dicho programa funcionó, pero debido a la imposibilidad de modificar y extender este programa, este método fue abandonado y se diseñó una versión diferente basada en la solución anterior.

3.1 Problema rotacional

Para la resolución del problema rotacional se utilizó un programa ya existente basado en el uso de cuaterniones; dicho enfoque resultó muy eficiente, y lo único que se ajustó posteriormente fue la frecuencia relativa de funcionamiento del programa Arduino contra el programa Processing.

3.2 Problema traslacional

Para esta cuestión se ensayaron varias soluciones posibles, de distinto nivel de complejidad pero todas partiendo del mismo enfoque: estimar una curva de aceleración y a partir de ella integrar dos veces para determinar cambios en la posición.

La primera solución propuesta era ligeramente distinta a la que acaba de describirse: en lugar de estimar una curva de aceleración, se hacía una aproximación lineal de la curva de velocidad, como lo muestra la Figura 3.2.1. La pendiente de cada segmento lineal es una constante que equivale a la aceleración en dicho intervalo; tomando un promedio de las aceleraciones medidas, se estimó dicha aceleración y luego se procedió con las ecuaciones usuales del MRUV, la ecuación horaria de posición (Ecuación 1) y la de velocidad (Ecuación 2):

$$x(t)=v_0 t+\frac{1}{2}at^2$$ (Ecuación 1)

$v(t)=v_0+at$ (Ecuación 2)

No se considera el término de la posición inicial porque interesa calcular únicamente la magnitud de *desplazamiento*.

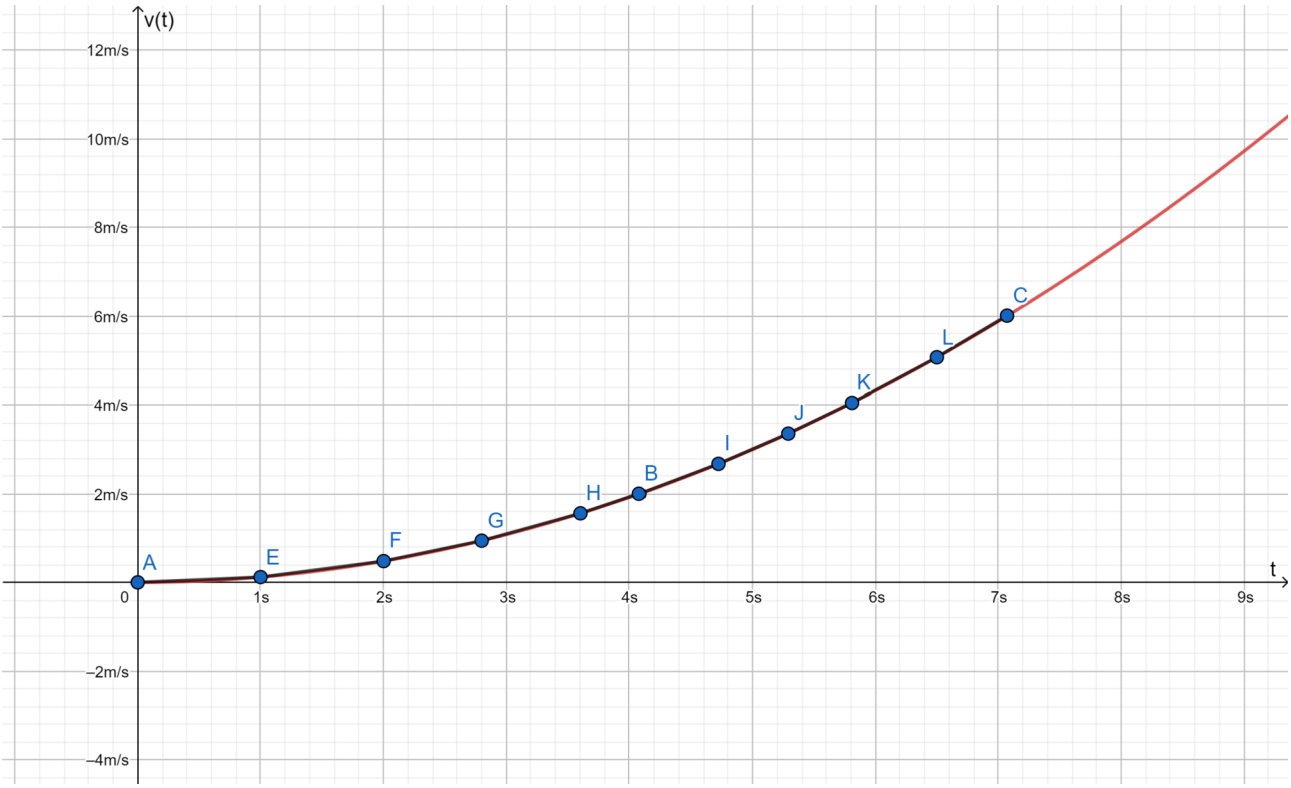

Fig. 3.2.1 Aproximación lineal de la curva de velocidad

La segunda solución, que es la que está actualmente implementada, es un paso más compleja. Supone efectuar una serie de aproximaciones lineales de la curva de aceleración y obtener la posición genérica asociada a dicha función de aceleración. En las secciones 4.1 y 4.2 se explicará este algoritmo.

La tercera solución, desarrollada en detalle en la sección 4.3, consiste en realizar una estimación estadística de la curva de aceleración. Aprovechando la potencia del microcontrolador Arduino, se pueden realizar mediciones a una frecuencia mucho más alta y obtener una gran cantidad de muestras en un intervalo de tiempo relativamente corto. Con dichas muestras *(t,a)*, se puede realizar un ajuste polinomial para obtener

una curva estimada que sea mucho más representativa de la aceleración real que la aproximación lineal propuesta en la solución anterior.

4. Desarrollo de la solución y algoritmos

4.1 Problema rotacional

Como ya se ha dicho, el algoritmo que se utilizó para resolver el problema rotacional ya estaba implementado. Dicho algoritmo consiste en una serie de transformaciones sobre los valores que arroja el sensor, para llevarlos a distintos formatos, cada uno de ellos con sus propias ventajas para operar y calcular. Finalmente se devuelven los datos en tres representaciones, y la interface visual se actualiza generando una rotación en base a estos resultados.

Los tres formatos utilizados son:

- Cuaterniones: estos son un objeto matemático concebido como una extensión al conjunto de los números reales con adición de tres unidades imaginarias, según la Ecuación 3.

$$i^2 = j^2 = k^2 = ijk = -1 \quad \text{(Ecuación 3)}$$

  Los cuaterniones son especialmente útiles para representar movimientos rotacionales y operar sobre ellos. La librería MPU6050 proporciona funciones que abstraen al programador de realizar la transformación, y permite obtener directamente del sensor los valores del cuaternión asociado a la rotación. Para más información sobre el álgebra de cuaterniones y su relación con las rotaciones, consultar por ejemplo [1].

- Ángulos de Euler: conforman un sistema para describir rotaciones relativas entre dos sistemas de coordenadas, utilizando tres ángulos, como se muestra en la Figura 4.1.

- Yaw-Pitch-Roll: es el sistema más sencillo e intuitivo para visualizar rotaciones; representa los giros en torno a tres ejes, como se muestra en la Figura 4.2

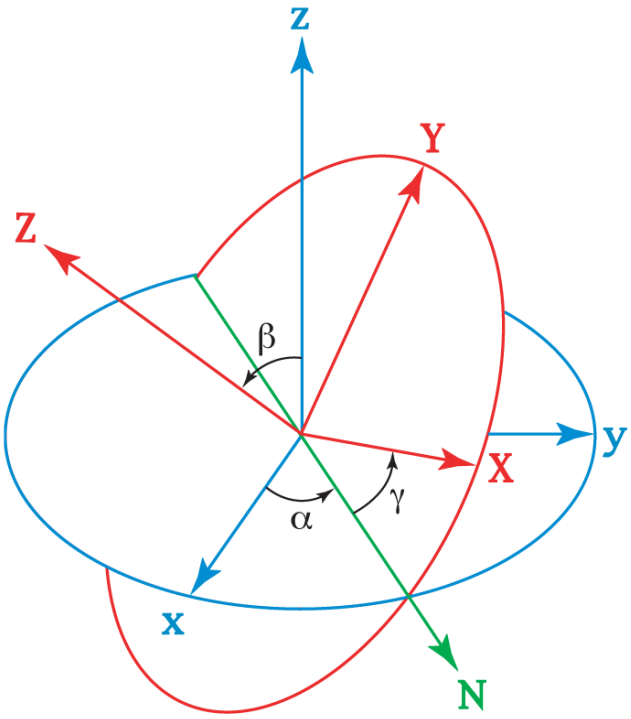

Fig. 4.1 Diagrama representativo de los Ángulos de Euler.

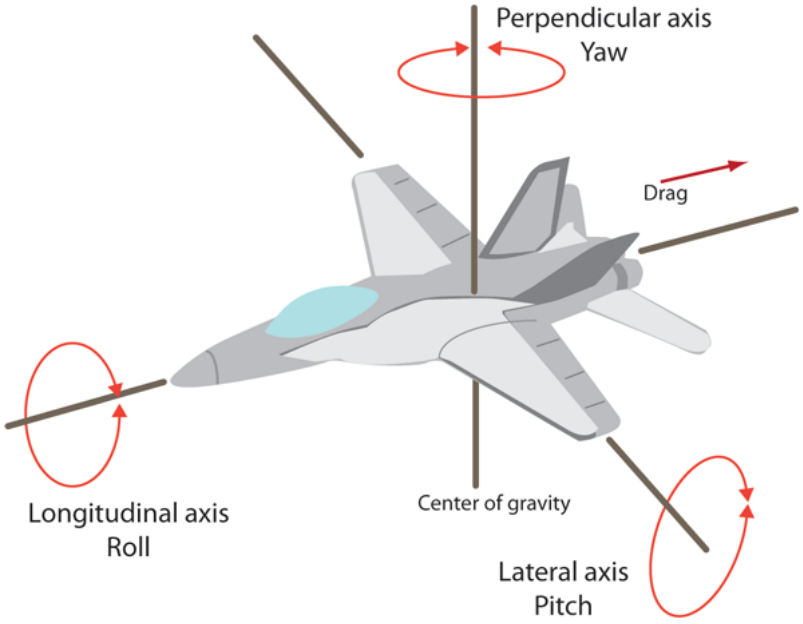

Fig. 4.2 Diagrama representativo de los ejes Yaw-Pitch-Roll

Las transformaciones entre los distintos sistemas ya se encontraban resueltas en el programa utilizado de base, y no forman parte del alcance

de este informe. Para ver una explicación profunda sobre el tema, ver por ejemplo el sitio web del Ing. Noel Hughes, [2].

4.2 Problema traslacional

Se describirá a continuación el algoritmo generado para resolver el problema del movimiento traslacional.

- Se toman las mediciones de aceleración con una frecuencia definida *f*. El período de muestreo es $T=1/f=t_2-t_1$.

- Dos mediciones sucesivas determinan un intervalo. En dicho intervalo, se aproximará la curva de aceleración por una recta; si la longitud del intervalo es pequeña, la aproximación será eficiente.

- La aproximación lineal en dicho intervalo es la recta secante que une sus dos extremos, los puntos en los que se ha medido la aceleración. Su pendiente equivale a la aceleración *media* en dicho intervalo, y está dada por la Ecuación 4:

$$m_s=\frac{a_2-a_1}{t_2-t_1}=a_{med} \qquad \text{(Ecuación 4)}$$

Y su ordenada al origen queda determinada por la aceleración inicial del intervalo.

- La aceleración y la posición en *cualquier* movimiento lineal, están relacionadas por la Ecuación 5:

$$\frac{d^2x}{dt^2}=a(t) \qquad \text{(Ecuación 5)}$$

Para calcular el desplazamiento, simplemente se integra dos veces la función aceleración y se reemplazan las condiciones iniciales. Para una explicación más detallada de este modelo, consultar cualquier libro elemental de física universitaria, por ejemplo [3].

4.3 Problema traslacional – solución estadística

Se propone una solución alternativa para el problema traslacional; dicha solución involucra algunos conceptos estadísticos que se presentarán a continuación.

Supóngase que se tiene una cierta familia de funciones $f(t,\vec{\theta})$, donde $\vec{\theta}$ es un vector de parámetros; para cada realización de este vector de parámetros queda definida una función de tal familia. Se dispone además de un conjunto de datos bivariados de la forma (t,a) , y se desea encontrar cual es la función f que mejor ajusta a dicho conjunto, es decir, encontrar el valor del vector de parámetros más adecuado. Para ello, lo primero es definir precisamente qué significa "ajustar bien". Un criterio posible es el siguiente:

- El vector de parámetros $\vec{\theta}$ correspondiente a la función que mejor ajusta los datos, es aquél tal que la función dada por la Ecuación 6 sea mínima:

$$S(\vec{\theta})=\sum_{i=1}^{n}[a_i-f(t_i,\vec{\theta})]^2 \qquad \text{(Ecuación 6)}$$

  Este criterio se conoce como el **criterio de los mínimos cuadrados**, y el método de estimación asociado recibe el nombre de **Método de Mínimos Cuadrados** (Least-Squares estimator).

Definida la familia de funciones $f(t,\vec{\theta})$, resta minimizar la función $S(\vec{\theta})$. Si dicha función es diferenciable, tendrá su mínimo (si es que éste existe) en el punto donde se satisfaga la Ecuación 7:

$$\frac{\partial S}{\partial \theta_1}=\frac{\partial S}{\partial \theta_2}=\ldots=\frac{\partial S}{\partial \theta_n}=0 \qquad \text{(Ecuación 7)}$$

Luego, se obtendrá un sistema de *n* ecuaciones (en general no lineales), que deberá resolverse numéricamente.

La propuesta es aprovechar la rapidez del procesador Arduino para efectuar una cantidad mayor de mediciones en un corto tiempo (el error estadístico del ajuste se reduce al aumentar la cantidad de datos, según

establece la Ley de los Grandes Números. Para más detalles sobre el Método de Mínimos Cuadrados y más cuestiones estadísticas, ver por ejemplo [4]), y realizar un ajuste de este tipo para obtener una curva de aceleración estimada. Como familia de funciones se propone utilizar un polinomio de grado 5, a saber:

$$f(t, a_0, a_1, a_2, a_3, a_4, a_5) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \quad \text{(Ecuación 8)}$$

El resto de la solución sigue el mismo patrón que fue explicado en la sección 4.2: empleando esta función estimada como curva de aceleración, se resuelve la ecuación diferencial integrando dos veces para obtener la función de posición, con las mismas condiciones iniciales especificadas en la solución anterior.

Para implementarlo, se desarrolló una pequeña librería en Java llamada Regression Calculator, la cual provee una función que recibe un conjunto de datos y devuelve un conjunto de coeficientes de regresión estimados. El ajuste se realiza empleando la librería Apache Commons Math. Regression Calculator se incluye como una librería adicional en la aplicación Processing, y se utiliza directamente.

5. Protocolo de comunicación

Adicionalmente al protocolo I2C que es el que utiliza el sensor para vincularse con Arduino, fue necesario definir un protocolo propio para la comunicación serial entre Arduino y la aplicación Processing.
Definimos entonces el siguiente protocolo:

- Los datos que se enviarán a través del puerto serie serán: las tres componentes del vector aceleración neta optimizada, es decir, sin la componente de la gravedad y la componente ocasionada por la rotación de la Tierra; y las cuatro componentes del cuaternión que representa el movimiento rotacional neto. La aceleración lineal se mide en metros por segundo cuadrado.

- Los datos de la aceleración lineal y del cuaternión se envían separados por el caracter ‘;’. A la izquierda, los datos de la aceleración lineal, y a la derecha los del cuaternión.

- Las mediciones de la aceleración lineal y las componentes del cuaternión respectivamente están separadas entre sí por el caracter ‘,’. Las componentes del vector aceleración están separadas por el carácter ‘:’, mientras que las del cuaternión se envían en el orden ‘w’ → ‘x’ → ‘y’ → ‘z’, separadas por comas. Las mediciones de las componentes de la aceleración lineal están separadas entre sí por comas.

- Cada trama enviada según este protocolo está separada de otras tramas por el carácter de fin de línea; no se incluye ningún separador adicional, puesto que las tramas se envían y se reciben línea por linea.

Ejemplo de trama: $a_{x1}, a_{x2}, \ldots, a_{x20} : a_{y1}, a_{y2}, \ldots, a_{y20} : a_{z1}, a_{z2}, \ldots, a_{z20} ; q_w, q_x, q_y, q_z$

6. Filtros utilizados

Para filtrar ruido en las mediciones, se utilizó un sencillo filtro basado en el promedio. La placa Arduino opera con una frecuencia mayor que la aplicación de Processing, de manera que cada vez que se envía una trama, ya fueron realizadas una cierta cantidad de mediciones, y no una sola. Los valores que se envían en dicha trama son promedios matemáticos de todas estas múltiples mediciones. De esta forma, se atenúa el ruido producido por *outliers*. El filtrado se hace del lado de Arduino, de manera que la aplicación Processing recibe los valores ya filtrados.

7. Ejemplos de uso

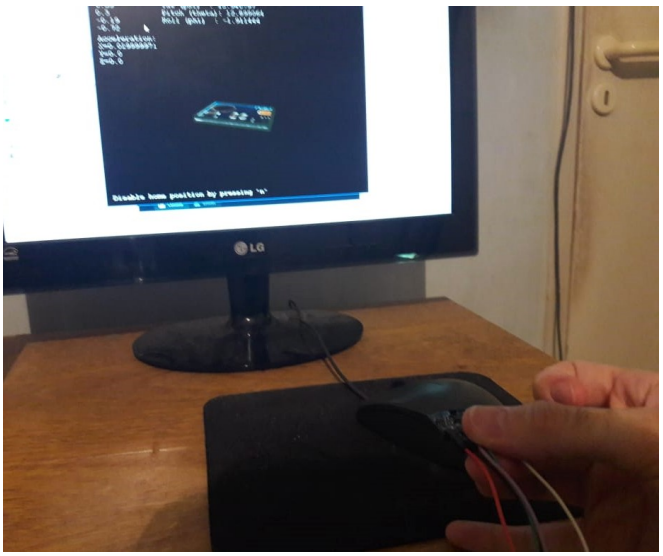

Fig. 7.1a

Fig 7.1b

Fotografías reales de la interface visual de la aplicación. En la figura 7.1a se observa como al sostener el dispositivo en una posición *lo más horizontal y firme posible*, y ajustar la posición de referencia de la simulación en dicho punto, en la pantalla se visualiza la representación gráfica del sensor en el centro geométrico de la interface. En la figura 7.1b se observa como al desplazarlo verticalmente en la dirección *-z*, el dibujo emula el movimiento real, más una cierta rotación proveniente del ruido mecánico introducido en la manipulación del sensor.

8. Conclusiones

En la finalización del trabajo completo, se obtienen las siguientes conclusiones:

- Medir magnitud de rotación utilizando el giróscopo es sencillo y no introduce tanto ruido como para hacerlo inviable.
- En contraste, no es **para nada recomendable** medir posición o desplazamiento lineal empleando un acelerómetro, tanto por su

extrema sensibilidad al ruido introducido por las rotaciones como por la cantidad de cálculos intermedios que hay que realizar. Existen formas muy superiores, más precisas y resistentes al ruido, como por ejemplo los sensores de ultrasonido.

- Existen múltiples formas de estimar la posición lineal. Se implemento una; y se sugiere como alternativa para un trabajo futuro la implementación del ajuste estadístico sobre múltiples valores de aceleración.
- Existe un problema intrínseco asociado al hecho de medir aceleración y rotación mediante el mismo dispositivo; este problema no puede resolverse por medios directos.
- Se podrían utilizar mejores filtros para disminuir el ruido, pero se optó por la alternativa más simple. Adicionalmente, se podría utilizar mayor cantidad de muestras al filtrar y al estimar para disminuir el error estadístico. Esta es la hipótesis que sustenta la propuesta del ajuste polinomial.
- El algoritmo realiza ocasionalmente una pequeña calibración para corregir el offset del sensor. Podría optimizarse la frecuencia y la forma en que se realiza dicha calibración.
- Es sencillo incluir código Java en el programa Processing, por lo cual mucho del código contenido en el sketch Processing podría reescribirse en Java, donde puede dársele un mejor diseño e incluso acompañarlo con tests. Este posible refactor puede ser una buena propuesta a futuro.
- **Resultados de la última implementación:** la última solución propuesta para el problema traslacional, implementada a partir de la versión 3 del programa, resultó ser extremadamente menos eficiente que la anterior. En detalle:
  - No se gana mayor precisión en la utilización del algoritmo estadístico, debido a que al realizar la estimación en intervalos pequeños, los términos de grado mayor a 2 del polinomio empleado (Ecuación 8) aportan muy poco al resultado general. Por ende, una aproximación lineal es una opción con el mismo rendimiento y mucho más sencilla.
  - Arduino puede operar y transmitir en ciclos del orden de microsegundos, pero Processing no puede operar en tiempos de orden inferior a los milisegundos. Por lo tanto, la tecnología utilizada desaprovecha la potencia de Arduino e impone una gran limitación de rendimiento.

- Los cálculos que hay que hacer para obtener los datos efectivos de aceleración, velocidad y posición son muchos más que en el caso de la aproximación lineal; entonces, los errores se acumulan con una rapidez mucho mayor.
- El envío de tramas de datos tan largas a través del puerto serie puede inducir errores muy fácilmente. Es recomendable, en lo posible, enviar tramas cortas o utilizar mecanismos de control más sofisticados.
- En la interface gráfica, el movimiento ya no resulta fluido sino lento e incluso errático. El autor de este informe no ha podido determinar la causa exacta de este comportamiento, pero maneja la suposición de que puede deberse a problemas con la frecuencia de operación del programa Processing en relación con la frecuencia a la que recibe la información, o bien a una limitación propia de la tecnología al procesar mayor cantidad de datos y efectuar mayor cantidad de cálculos.

9. Bibliografía

[1] Es.wikipedia.org. (2018). Cuaterniones y rotación en el espacio. [online] Available at: https://es.wikipedia.org/wiki/Cuaterniones_y_rotaci%C3%B3n_en_el_espacio [Accessed 28 Jun. 2018].
[2] noelhughes. (2018). Sitemap. [online] Available at: http://noelhughes.net/Sitemap.php [Accessed 28 Jun. 2018].
[3] Sears, F., Zemansky, M., Young, H., Adams, P. and Chastain, R. (n.d.). College physics.
[4] Dekking, F. (2005). A Modern Introduction to Probability and Statistics: Understanding Why and How. London: Springer.