Trabajo Práctico N2

Sensor Fusion

Estudiante:
Scala, Tobias 55391

Profesores:
Perfumo, Lucas Alberto
Fortunatti, Nelson Ariel

Materia:
Mecatrónica Aplicada

Facultad:
Instituto Tecnológico de Buenos Aires, ITBA

1. Introduccion

Sensor fusion consiste en la combinación de 2 o más sensores para lograr obtener con mayor calidad, robustez y exactitud la medición de una variable física determinada
El objetivo de dicho trabajo es poder realizar dicha combinación haciendo uso del sensor MPU6500. Dicho sensor consta de un giroscopio y acelerómetro por lo que el mismo presenta 6DOF.
Puesto a que no es posible conocer la orientación del sensor utilizando sólo uno de ambos sensores debido a que:

- Con giroscopio se obtiene deriva ya que el mismo presenta bias en su medición. Es por esto que el mismo es preciso a corto plazo.
- El acelerómetro es más sensible al ruido y es preciso a largo plazo.

Debido a que el bias es de baja frecuencia y el ruido es de alta frecuencia, es necesario usar filtros pasabajos y pasaaltos respectivamente para poder despresiarlos y así lograr una buena combinación de ambos sensores (sensor fusion).
Se usará el algoritmo de Madgwick para efectuar el sensor fusión. El mismo cuenta con los filtros antes mencionados. Cabe mencionar que el acelerómetro otorga una referencia externa el cual es la gravedad.
A continuación se observa la conexión entre el MCU (STM32F303RE) y el sensor (MPU6500).

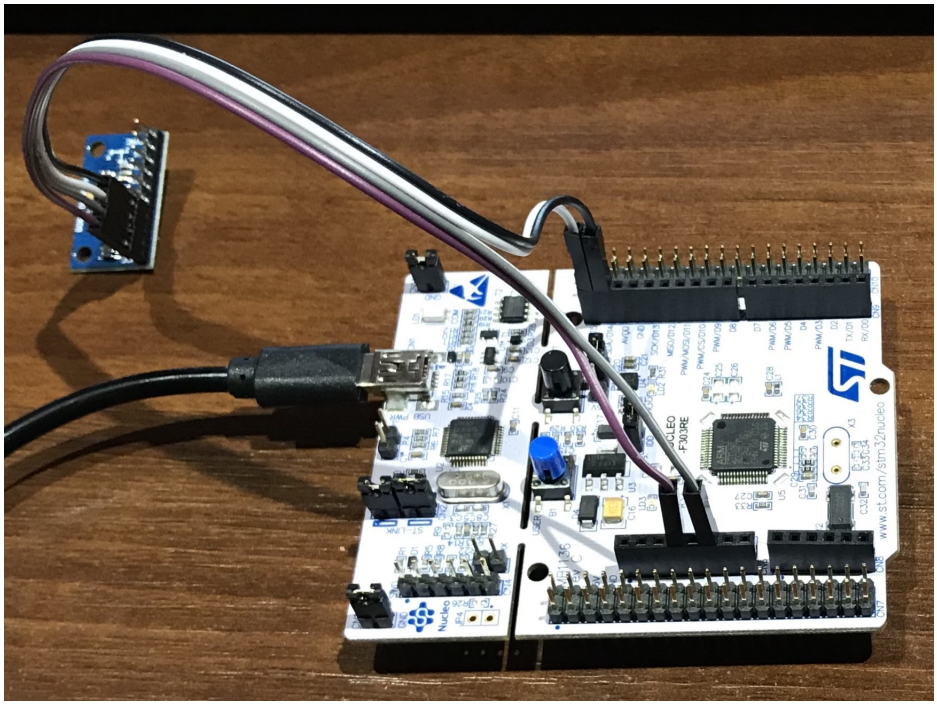

Figura 1: Conexión entre el MCU (STM32F303RE) y el Sensor (MPU6500).

2. Estructura e Implementación

Se ha implementado un par de códigos tanto en la PC (usando Python) como en el microcontrolador STM32F303RE (usando C) para poder obtener los valores del sensor MPU6500 y graficar los mismos en tiempo real. Ambos códigos son totalmente bolquentas con respecto a la transferencia de datos. A continuación se presentan los diagramas de flijo de ambos programas.

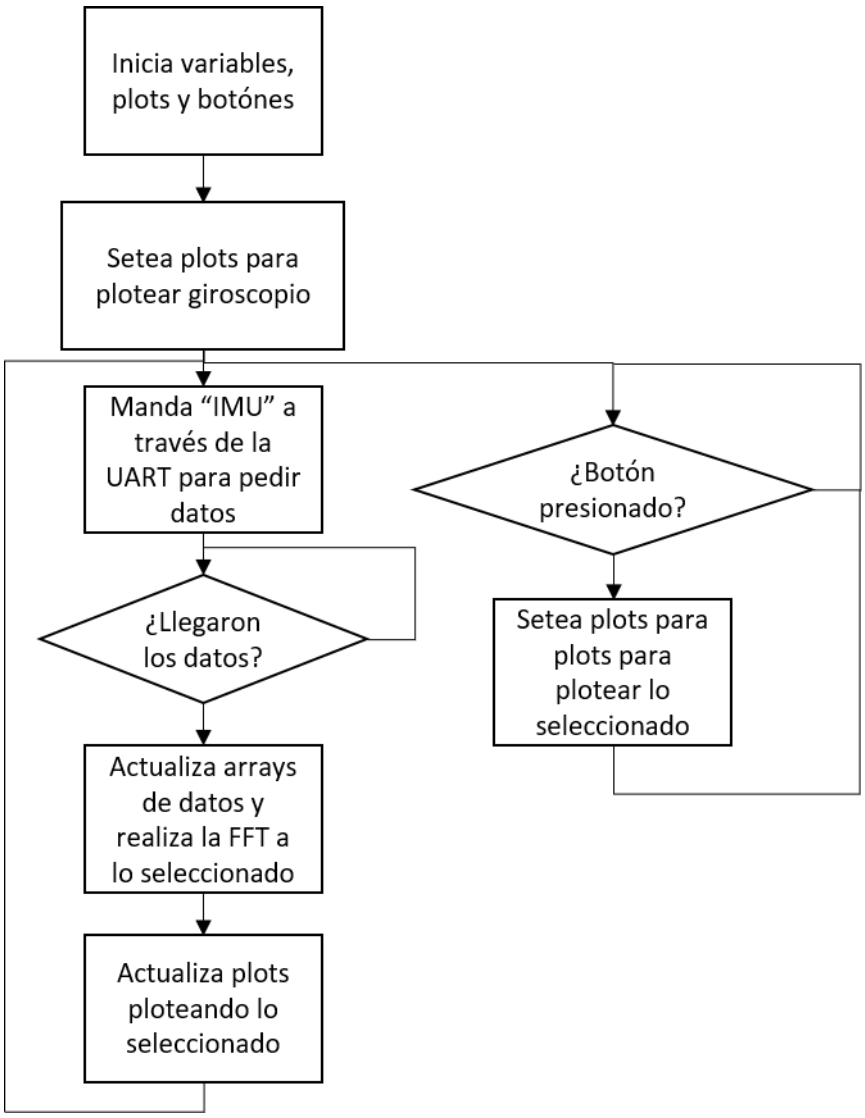

Figura 2: Diagrama de Flujo del Código Implementado en Python.

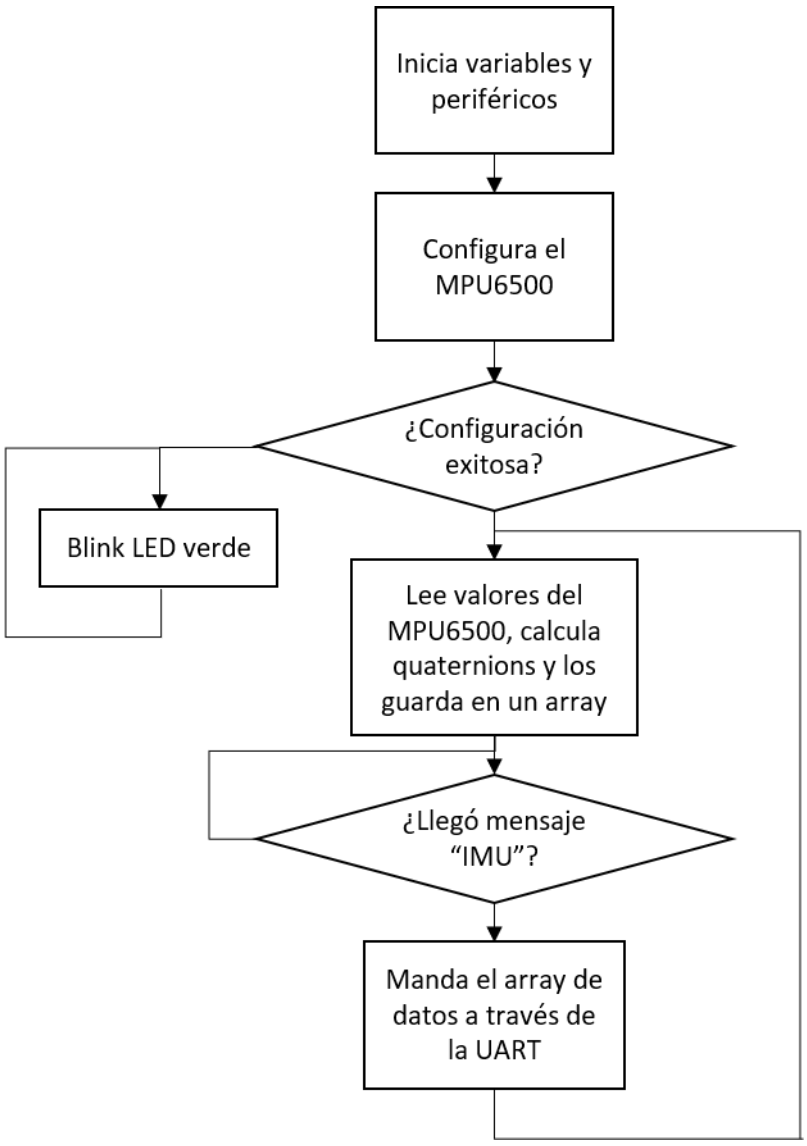

Figura 3: Diagrama de Flujo del Código Implementado en C.

2.1. Programa Implementado en Python

Este programa se encarga de pedir los valores del sensor MPU6500 al MPU, de calcular la FFT de las señales obtenidas y fija la frecuencia de sample.
A modo de GUI, se han usado botones de la librería "matplotlib" para que el usuario pueda elegir qué datos graficar. Para poder visuzlizar los gráficos en tiempo real se ha usado la función "FuncAnimation".
A continuación se observa una imagen de la GUI.

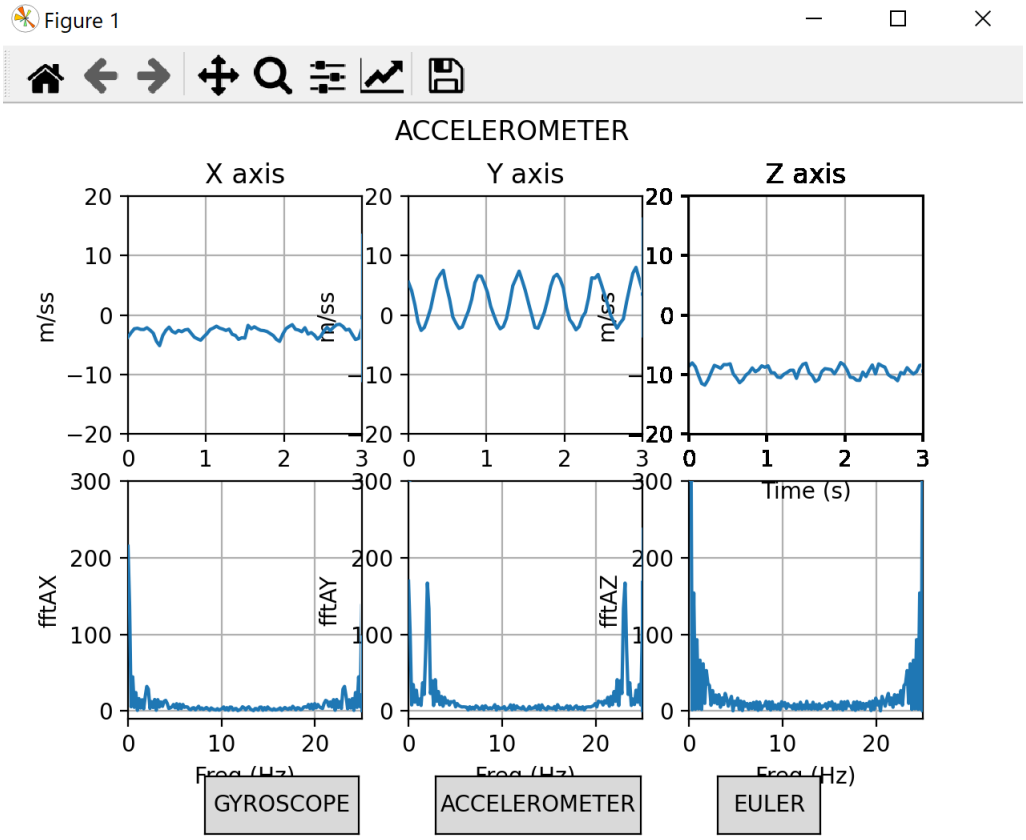

Figura 4: GUI diseñado con botones y los gráficos.

Puesto a que la función "FuncAnimation" es lenta, se ha medido el tiempo que tarda en plotear los gráficos obteniendose un valor de 0.04 segundos. Esto ha limitado a que la frecuencia de sample sea de 25 muestras por segundo. Puesto a que 25 puntos es muy poco para graficar, se optó por graficar 3 segundos dando un total de 75 puntos.

2.2. Programa Implementado en C

Este programa se encarga de hacer una lectura de los datos del sensor MPU6500 (a través de I2C), de calcular los quaternions y de enviar los datos a la PC (a través de UART).
Los periféricos UART y I2C se han configurado de forma fácil utilizando el propio IDE del microcontrolador. Para la configuración del sensor MPU6500 se ha utilizado el programa en C publicado en github cuyo link: `https://github.com/nelsonfort/RTOSyPROTOCOLOS`. El mismo fue editado para su uso con el sensor MPU6500 y con el microcontrolador STM32F303RE.
Como se puede ver en el diagrama de flujo (Figura 3) el programa es bastante simple.
Los quaternions se obtienen mediante el programa en C el cual implementa el filtro de Madgwick tanto para IMU como MARG. Se ha utilizado el filtro de Madgwick referido para IMU ya que el sensor MPU6500 no cuenta con un magnetómetro (compass), por lo tanto es de 6DOF.

2.3. Sincronización para la Transferencia de Datos

Se logró una sincronización entre la PC y el microcontrolador a través de un intercambio de mensajes. Es decir, la PC pide datos mandando un mensaje ÏMUz el MCU inmediatamente recibido el mensaje manda los datos obtenidos por el sensor justo con los quaternions.

3. Resultados

A continuación se presentan un par de gráficos correspondientes a las mediciones del giroscopio y del acelerómetro. Se puede notar el primer armónico en frecuencia en aquellos gráficos con medición oscilatoria.

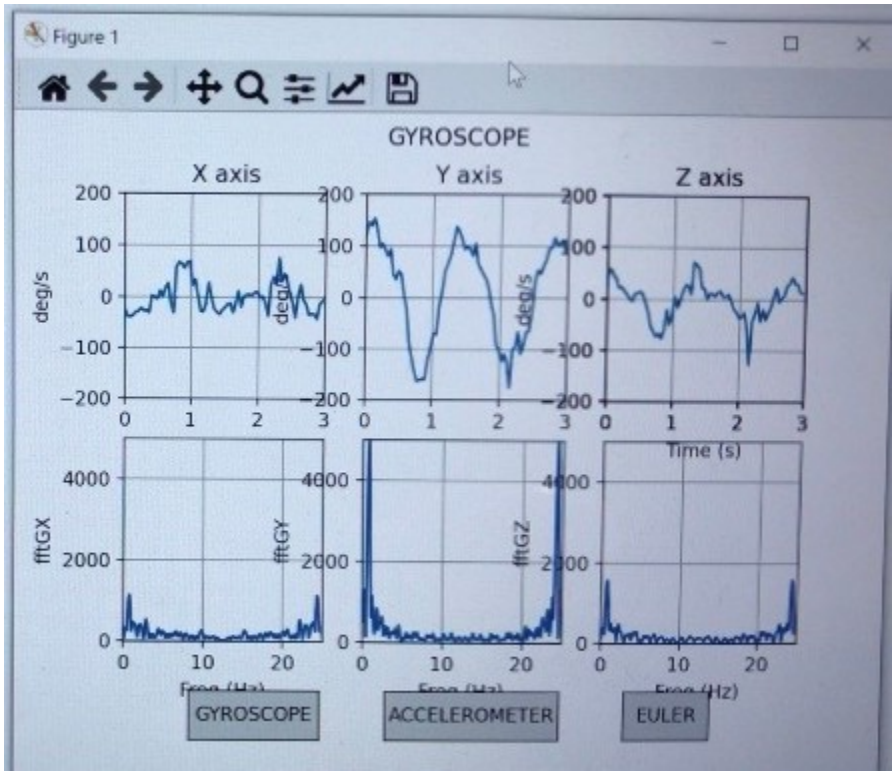

Figura 5: Gráficos del Giroscopio.

Figura 6: Gráficos del Acelerómetro.

4. Problemas Encontrados

Debido a que el sensor no es el MPU9250 (9DOF), no es posible obtener los ángulos de Euler de la forma que se propone en el paper de Madgwick. Ya que corresponde a una secuencia de orientación que empieza con respecto al eje Z (yaw). Al no tener un compass, no es posible obtener los águlos de Euler de forma correcta.
Una solución a este problema fue usar una forma cuya secuencia de orientación empiece con respecto al eje X (roll). Se encontró el código en Python del mismo en el link:
`https://math.stackexchange.com/questions/2975109/how-to-convert-euler-angles-to-quaternions-and-get-the-same-euler-angles-back-fr`.
Con este código se pudieron obtener los ángulos de Euler correspodientes a cada eje. Lamentablemente este no se obtuvieron valores precisos de los ángulos puesto a que los gráficos muestran valores de aproximadamente el doble de lo que realmente ha sido girado el sensor.
Un último problema encontrado ha sido con respecto al acelerómetro. Al obtener los valores (en crudo) desde el MPU y orientando el sensor de tal forma que el eje "Y" apunte hacia arriba, se obtuvo un cambio de señal en el gráfico correspondiente al eje "X". Este problema ha sido simplemente solucionado con el siguiente código:

```
1 data[3] = -mpu6500GetAccelY_mss();
2 data[4] = -mpu6500GetAccelX_mss();
```