Adquisición de datos de una IMU por medio de Python y Matlab

Charitz Lizeth Bejarano Bolaños
Ingeniería Mecatrónica
Universidad ECCI
Bogotá, Colombia
charitzl.bejaranob@ecci.edu.co

Daniel Santiago Caycedo Rivera
Ingeniería Mecatrónica
Universidad ECCI
Bogotá, Colombia
daniels.caycedor@ecci.edu.co

Erick Santiago Soto Barrantes
Ingeniería Mecatrónica
Universidad ECCI
Bogotá, Colombia
ericks.sotob@ecci.edu.co

I. Resumen

En este documento se muestra la adquisición de datos de una IMU MPU 6050 describiendo el proceso de desarrollo del algoritmo y las herramientas utilizadas, mostrando la información de los acelerómetros y los giroscopios integrados en el componente que describen la aceleración y velocidad angular en los ejes coordenados. Por medio del uso de una tarjeta de desarrollo STM32 programando los periféricos necesarios para realizar la adquisición de datos usando el protocolo de comunicación I2C y enviar los datos por medio de comunicación serial para su procesamiento en un computador. Para calibrar la información obtenida se genera un código utilizando la media aritmética para generar los datos Offset y realizar las operaciones necesarias para tener los datos correctos. Finalmente se utiliza el software Matlab en un computador con Windows y el lenguaje de programación Python en una Raspberry Pi 3 B+, adquiriendo la respuesta del sensor IMU por medio del protocolo de comunicación serial enviados previamente desde la tarjeta de desarrollo STM32, para ser procesados y graficados la información obtenida de los acelerómetros y giroscopios, permitiendo conocer la efectividad de la aplicación de estas herramientas.

***Palabras claves*—IMU, Matlab, STM32, Comunicación Serial, Comunicación I2C, Protocolo VNC.**

II. Introducción

Una unidad de medición inercial es uno de los sensores electrónicos más usados en la actualidad, permite conocer la orientación y movimiento de un objeto en los tres ejes coordenados, el uso de esta información permite el control de diversos sistemas donde se implemente este tipo de sensor.

La adquisición y procesamiento de datos es uno de los retos que conlleva la aplicación de una IMU, para ello se pueden utilizar sistemas embebidos como las tarjetas de desarrollo STM32 adquiriendo los datos por medio del protocolo de comunicación I2C para luego ser enviados a una computadora y ser procesados por medio de lenguajes de programación como Python o en el ambiente de desarrollo de Matlab.

III. Marco Teórico

III-A. Python

Es un lenguaje de programación interpretado, multiparadigma, multiplataforma, gratuito y con una sintaxis legible, se ha convertido en uno de los lenguajes de programación con mayor auge en el siglo 21 debido a su facilidad para aprender y a la diversidad de librerías que se pueden encontrar para diversas aplicaciones [1].

III-B. Matlab

Es una herramienta de software que permite realizar cálculos basado en matrices integrando un conjunto de librerías y complementos que facilitan el tratamiento de datos en aplicaciones científicas y de ingeniería [2][3].

III-C. MEMS

Los MEMS o también conocidos por sus siglas en inglés Microelectromechanical systems se refiere a la tecnología electromecánica de dispositivos microscópicos que van desde sensores hasta actuadores de tamaño reducido, entre los cuales están componentes como las unidades de medición inercial [4].

III-D. IMU

El sensor de medición inercial IMU combina un giroscopio y un acelerómetro por lo que permite obtener información acerca de la orientación y desplazamiento que puede presentar un objeto [5].

III-E. Comunicación I2C

El protocolo de comunicación I2C es una manera de comunicar distintos dispositivos en una misma red estableciendo relaciones de maestro esclavo, véase Figura1 donde el maestro puede solicitar o escribir información a un esclavo, esto se logra asignando una dirección al esclavo de forma que se indica primero el esclavo con el cual se desea comunicar, luego se establece si se desea leer o escribir datos y por ultimo se finaliza la comunicación con ese esclavo o dispositivo [6].

III-F. STM32

La tarjeta de desarrollo STM32f746zgtx es un sistema embebido que cuenta con múltiples entradas y salidas de voltaje, módulos ADC y DAC, así como también módulos de comunicación I2C, serial, Ethernet y USART. Esta gran cantidad de funciones permiten el desarrollo de múltiples aplicaciones [7].

III-G. *Raspberry*

Es un ordenador de bajo costo con un tamaño reducido y de bajo costo, el software que maneja es de código abierto permitiendo debido a la gran variedad de periféricos permitiendo realizar todo tipo de aplicaciones en la industria y de forma casual [8].

III-H. *Comunicación VNC*

Es un protocolo de comunicación que permite la conexión remota a un equipo por medio de la conexión a un servidor para poder visualizar y controlar el computador desde otro cliente Fig.1 [9].

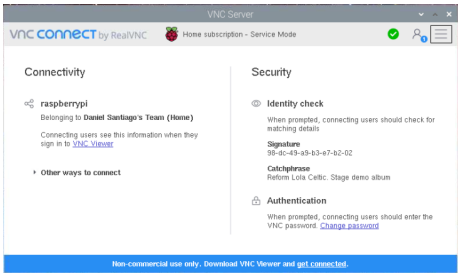

Figura 1. Servidor VNC en Raspberry

III-I. *STM32CubeIDE*

Es un IDE de programación ideado para la programación de tarjetas de desarrollo STM32, permite la generación de código para la configuración de los GPIO requeridos [10].

IV. Metodología

IV-A. *Programación STM32*

Se programa la tarjeta STM32 por medio de STM32cubeIDE Fig.2 empezando con la configuración de los GPIO que se van a usar y la frecuencia que va a trabajar el microcontrolador de la STM32. se configura la comunicación USART2 de forma asíncrona con una tasa de bits de 9600, el modulo de comunicación I2C1 y se coloca el oscilador interno a 216 MHz.

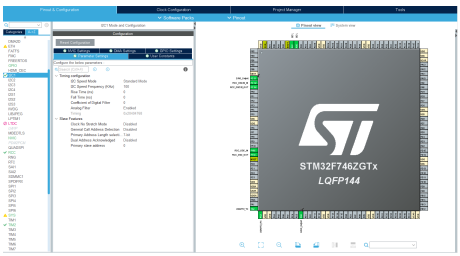

Figura 2. Configuración de GPIO eb STM32CubeIDE

Se programa dentro de las interrupciones la comunicación con la IMU primero realizando un test Who I am para verificar la conexión, después se realiza la adquisición de datos de los 3 giroscopios y de los 3 acelerómetros de cada eje, teniendo los 6 datos se envían por el puerto serial. Los datos se envían separados por coma cuando se vayan a analizar en Matlab y por espacio si se van a leer en la Raspberry por Python.

IV-B. *Cableado del circuito*

Se conecta los puertos transmisor y receptor del conversor USB-serial a los pines de comunicación serial de la STM32 de forma cruzada. Para la conexión de la IMU se energiza a VCC y a GND, y se conecta los puertos de SDA y SCL de la IMU a los puertos correspondientes de la comunicación I2C de la STM32.

IV-C. *Calibración de la IMU*

Para la calibración de los datos obtenidos de la IMU se realiza la adquisición de 100 datos en posición neutra, hallando los valores máximos y mínimos de la muestra, con estos datos se calcula la media aritmética obteniendo los offset necesarios para calibrar la medida. Después de tener los offset se utilizan para adquirir los datos en el programa principal restando los datos offset a los adquiridos.

IV-D. *Programación Matlab*

Primero se realiza el código para realizar la conexión con el puerto USB que se esta utilizando para enviar los datos, configurando a 9600 baudios para la adquisición de datos. Se programa un bucle con la cantidad de iteraciones necesarias, se utiliza la función de Matlab "strslplitz values para separar los datos y convertir los datos de string a tipo flotante.

IV-E. *Configuración de la Raspberry*

Se Configuró y habilitó la comunicación del protocolo VNC en la Raspberry con el fin de poder trabajar y programar de forma remota en otro equipo sin necesidad de conectar un monitor para poder visualizar. Finalmente se descarga el lenguaje de programación Ptython, el IDLE y las librerias que se van a necesitar.

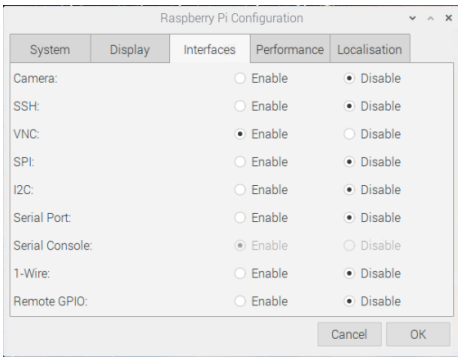

Figura 3. Configracion comunicación VNC en Raspberry

IV-F. *Programación Python*

Para el codigo de adquisición en Python se configura el puerto serial de la IMU en el que se conecta el conversor USB-serial. Se realiza el algoritmo necesario para la adquisición de datos guardándolos en tres matrices. Se guardan los datos

inicialmente en uno de los arreglos para separarlos por medio de una función de Python y pasarlos de variables tipo string a tipo flotantante y guardarlos en las otras dos matrices donde irán los datos calibrados y no calibrados. Finalmente se grafican para poder comparar los resultados.

V. Resultados

Primero se realizó el análisis de los resultados obtenidos en los acelerómetros, en las gráficas se observa la medición obtenida en el eje z.[en] función del tiempo en el eje "x". Se observa en el caso de las gráficas tomadas por medio de Matlab y Python que el eje z esta en un valor cercano a 1 debido a la posición en la que se encuentra la IMU mirando hacia arriba Fig.6 Fig.4. Se evidencia un error en la medición de los ejes ” x” y ”y” en las señales no calibradas Fig.5 Fig.7, tienden a verse un poco separadas una de la otra sabiendo que tienen que estar cercanas a 0.

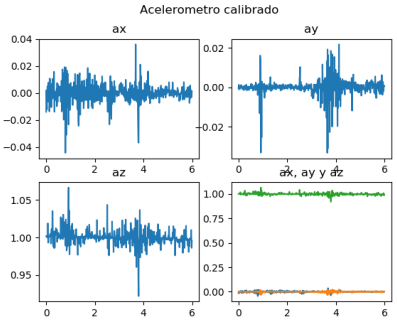

Figura 4. Datos calibrados del acelerómetro en la Raspberry por medio de Python

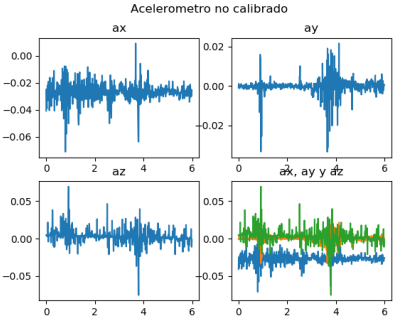

Figura 5. Datos no calibrados del acelerómetro en la Raspberry por medio de Python

Se analizan los datos obtenidos de los giroscopios en los cuales se puede observar una mayor cantidad de ruido en la señal debido al movimiento involuntario del lugar en el que se dispuso el sensor Fig.10 Fig.8. También se observa que a diferencia de los acelerómetros los datos se ven más distorsionados a su valor real si no se calibran anteriormente Fig.11 Fig.9.

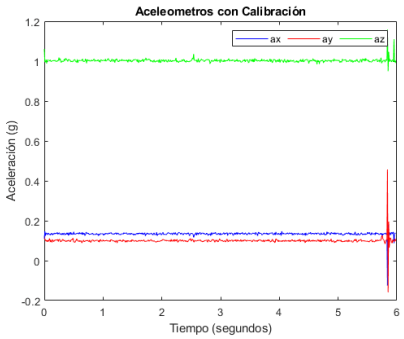

Figura 6. Datos calibrados del acelerómetro Matlab

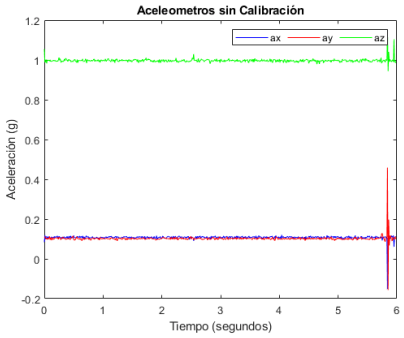

Figura 7. Datos no calibrados del acelerómetro Matlab

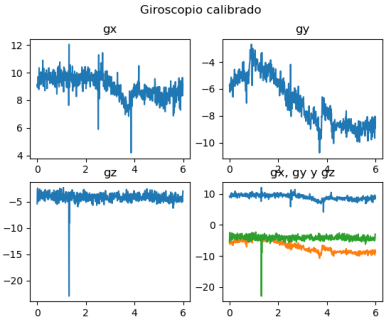

Figura 8. Datos calibrados del Giroscopio en la Raspberry por medio de Python

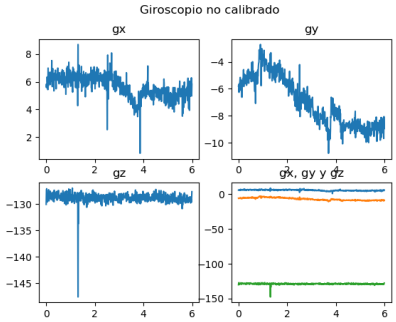

Figura 9. Datos no calibrados del Giroscopio en la Raspberry por medio de Python

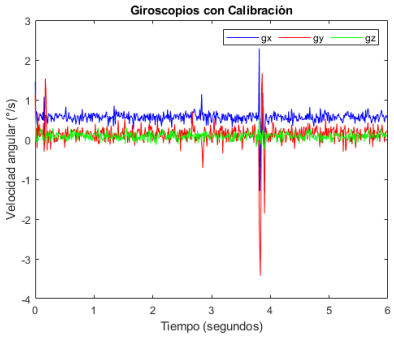

Figura 10. Datos calibrados del Giroscopio Matlab

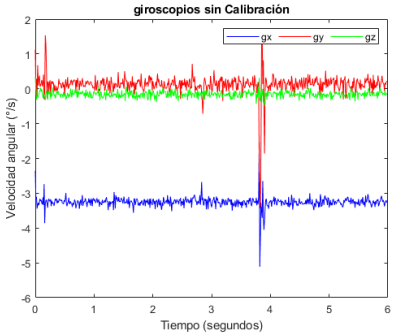

Figura 11. Datos no calibrados del Giroscopio Matlab

VI. Conclusiones

Se presenta el uso de dos métodos para la adquisición y procesamiento de datos obtenidos de una unidad de medición inercial, permitiendo analizar su respuesta en el tiempo y los cambios que pueden llegar a afectar la medición. Para una correcta adquisición de datos es necesario calibrar correctamente los datos de la IMU debido a que presenta un error en los valores netos arrojados y es necesario corregirlos por medio de unos offset obtenidos previamente.

Se puede observar la eficiencia para el procesamiento y graficación de datos tanto en Matlab en un computador como Python en una Raspberry Pi, mostrando el uso de diferentes herramientas con diferente costo y nivel de dificultad para una misma aplicación.

Referencias

[1] R. A. B.-G. Ivet Challenger-Pérez, Yanet Díaz-Ricardo, *El lenguaje de programación Python*. Universidad de Holguín, 2014.

[2] L. P. C. A. G. A. A. P. V. Óscar Reinoso García, Luis Miguel Jiménez García, *MATLAB: conceptos básicos y descripción gráfica*. Universitas Miguel Hernandez, 2018.

[3] C. O. J. Tintaya, *PROCESAMIENTO DIGITAL DE SEÑALES SÍSMICAS CON MATLAB*. Revista de Investigación de Física Instituto Geofísico del Perú, 2007.

[4] C. W. d. S. Farbod Khoshnoud, *Recent advances in MEMS sensor technology-mechanical applications*. IEEE, 2012.

[5] "Mpu-6050 datasheet," https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf, consultado: 2021-04-06.

[6] A. M. Fernández, *TEMA 5, EL BUS I2C*. Universidad de Córdoba, 2004.

[7] STmicroelectronics, "Stm32 st-link utility user manual."

[8] G. H. E Upton, "Raspberry pi user guide," 2014.

[9] "Realvnc," https://www.realvnc.com/pt/, consultado: 2021-04-06.

[10] "Stm32cubeide integrated development environment for stm32," https://www.st.com/en/development-tools/stm32cubeide.html, consultado: 2021-04-06.