Desarrollo de Aplicaciones utilizando placas de expansión con sensores y periféricos

Ipharraguerre Facundo

Instituto Universitario Aeronáutico, Facultad de Ingeniería
Tutores: Esp. Ing. Ducloux José, Mg. Ing. Riso Héctor

Noviembre de 2017

Resumen: En este trabajo se analizaron y desarrollaron aplicaciones en lenguaje C utilizando placas de expansión que poseen diversos sensores, conversores y otros dispositivos periféricos, los cuales posteriormente podrían ser conectados a la Computadora en Placa (SBC) del proyecto PIDDEF 44/12. Debido a la necesidad de tomar muestras de distintas condiciones en las que se desempeña una aeronave en vuelo para luego utilizar estos datos en simulaciones más precisas y teniendo en cuenta la disponibilidad comercial de sensores ambientales e inerciales a un precio accesible, a través de un análisis de prestaciones de estos dispositivos periféricos se ha intentado lograr un correcto funcionamiento de estos mismos en un ambiente de laboratorio para luego ser utilizados en futuras implementaciones y desarrollos. A través del estudio del hardware y del software preexistente, además del desarrollado en el laboratorio, se pudo observar la capacidad de los circuitos integrados para operar en un modo autónomo, confirmando de esta forma la posibilidad de implementación de sistemas de telemetría y adquisición de datos utilizando diseños electrónicos personalizados.

Introducción

El trabajo se divide esencialmente en dos partes, la primera trata de analizar el comportamiento de los dispositivos incluidos en la placa de expansión Sense HAT para Rapsberry Pi. Estro conllevó el desarrollo del código en lenguaje C de un controlador para permitir el acceso a las diferentes funciones de una Unidad de Medición Inercial (IMU, inertial measurement unit), para la cual se encontraban disponibles únicamente controladores en lenguaje Python y C++. La segunda parte del trabajo se basa en realizar mediciones sobre el rendimiento que se obtiene al realizar conversiones analógicas/digitales (y viceversa) utilizando la misma computadora Raspberry Pi 3 que se utilizó en la primera parte del trabajo, pero anexándole una placa de expansión para conversión AD/DA de alta precisión.

Análisis del módulo Sense HAT

Sense HAT (ver figura 1) es una placa de expansión para Raspberry Pi, la cual fue diseñada especialmente para la misión Astro Pi[1]. Actualmente el módulo Sense HAT está disponible para su compra a través de varios puntos de venta para el público en general.

La placa Sense HAT se compone de una matriz LED RGB 8×8, un joystick de 5 botones (on/off, no analógico) e incluye un set de sensores:

- Giróscopo/girómetro (mide velocidad angular) de 3 dimensiones
- Acelerómetro (mide aceleración lineal) de 3 dimensiones
- Magentómetro (mide densidad de flujo magnético) de 3 dimensiones
- Termómetros (son tres termómetros en total)
- Barómetro
- Higrómetro

La *Raspberry Pi Foundation* además creó una librería Python para proveer acceso de una manera más simple a los instrumentos de la placa (ver enlace en el anexo).

Figura 1

1 Se trata de una competencia de programación destinada a estudiantes menores de 19 años en la cual se proponen diversas aplicaciones para ser ejecutadas en la Estación Espacial Internacional. Más información en https://astro-pi.org/

Especificaciones de los sensores:

Presión / Temperatura (ST Micro LPS25H)
– Barómetro de 24-bit de resolución (desde 260 hPa hasta 1260 hPa)
– Termómetro de 16-bit de resolución (0 a 125° C)

Humedad / Temperatura (ST Micro HTS221)
– Higrómetro de 16-bit de resolución (0 - 100% RH)
– Termómetro de 16-bit de resolución (0 - 60°C)

Acelerómetro / Girómetro / Magnetómetro (ST Micro LSM9DS1)
– 9 grados de libertad (ejes X, Y, Z independientes para cada sensor)
– Rango de medición de aceleración lineal de hasta ±16 g
– Rango de medición de flujo magnético de hasta ±16 gauss
– Rango de medición de velocidad angular de hasta ±2000 dps (grados por segundo)
Cada uno de los canales de medición del LSM9DS1 tiene una resolución de 16 bit.

Cada uno de estos sensores tiene capacidad de muestro periódico del sensor e incluye un almacenamiento FIFO interno en el mismo IC. Los sensores LPS25H y HTS221 tienen tasas de muestreo de hasta 25 Hz, el LSM9DS1 logra una tasa de muestreo de hasta 952 Hz.

Matriz LED
La matriz de LED es controlada por una combinación de un controlador de LED de corriente constante y un microcontrolador Atmel ATTiny88 en el cual se ejecuta un firmware que provee al display de 8×8 de una profundidad de colores 15 bit (consulte el anexo para más información sobre el firmware AVR).

Joystick
El microcontrolador Atmel es el responsable de muestrear el joystick. El uso del microcontrolador en el display provoca que no queden disponibles los 5 pin necesarios para poder muestrear los ejes independientemente, por lo que se han unido los ejes a los pines de selección de fila de la matriz de LEDs. La frecuencia de actualización del joystick es de aproximadamente 80 Hz, la cual es la tasa de refresco de la matriz LED.

Todos los sensores y el firmware del Atmel son accesibles a través del bus I2C. Como extra, el bus SPI del controlador Atmel está conectado al conector GPIO, por lo tanto se puede reprogramar el firmware del microcontrolador aun estando en uso el bus I2C. El firmware stock está desprotegido y se puede sustituir.

En el anexo se encuentran las hojas de datos de los integrados y el esquema eléctrico de la placa.

Método experimental

Detección del dispositivo a través del bus I2C

Para listar fácilmente los dispositivos conectados al bus I2C se utilizó el programa “i2cdetect”.

```
sudo apt-get install i2c-tools

sudo i2cdetect -y 1
```

Debería entonces mostrar la lista de dispositivos conectados al bus I2C en forma de matriz. Por ejemplo:

```
pi@raspberrypi:~/$ i2cdetect -y 1

     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- 1c -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- UU -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- 5c -- -- 5f
60: -- -- -- -- -- -- -- -- -- -- 6a -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Donde las direcciones mostradas hacen referencia a los siguientes dispositivos:

0x5c: LPS25H

0x1c: LSM9DS1

0x5f: HTS221

0x46: LED2472G (puede aparecer como “UU” pero se respeta la fila y columna)

0x6a: LSM9DS1

Obtención de las Librerías en lenguaje C

Si bien los dispositivos I2C generalmente son controlados por el driver del kernel de Linux, es posible acceder a los dispositivos desde el espacio de usuario utilizando la interfaz /dev. Para esto hay que cargar la librería “i2c-dev.h” (consulte el anexo para más información sobre esta librería).

Puede resultar necesario instalar los siguientes paquetes: libi2c-dev, i2c-tools.

```
sudo apt-get install libi2c-dev i2c-tools
```

Se adjuntan con este archivo los ejemplos en lenguaje C para obtener datos de los sensores y escribir en el buffer de la matriz de LED.

Los ejemplos de control de sensores de presión, humedad y el control de la matriz de LED se obtuvieron del repositorio GitHub de Dave B. (dave.bird@dsl.pipex.com).

Desarrollo del controlador del IMU

El controlador del sensor inercial LSM9DS1 fue desarrollado en el laboratorio utilizando lenguaje C siguiendo como ejemplo el firmware de SparkFun y utilizando como referencia la hoja de datos del integrado (consulte el anexo).
Las funciones de inicialización, configuración y adquisición de datos se encuentran implementadas en el archivo “LSM9DS1.c”.

Si bien el archivo de cabecera “LSM9DS1.h” contiene las definiciones, las funciones de control de interrupciones no se encuentran disponibles actualmente al no haberlas incluido en el archivo LSM9D1S.c.

Un ejemplo del funcionamiento del sensor inercial LSM9DS1 se encuentra adjunto con el nombre “prueba.c”.

Para compilar el programa de prueba (o el que se haya creado para aprovechar las funciones que se encuentran en el código fuente “LSM9DS1.c”) se utiliza **GNU Compiler Collection (GCC)** especificando el *link* entre ambos. Por ejemplo, para compilar enlazando los códigos de las fuentes “LSM9DS1.c” y “prueba.c” para dar salida a un archivo “prueba” y que el compilador muestre las advertencias, se utiliza la siguiente forma:

```
gcc -Wall LSM9DS1.c prueba.c -o prueba
```

El parámetro “`Wall`” se refiere a “warn all”, no es necesario utilizarlo pero es recomendable para recibir notas y advertencias que no son errores fatales.

En general:

```
gcc <InputFile1> <InputFile2> -o <OutputFile>
```

El esquema de la figura 2 muestra la organización del controlador.

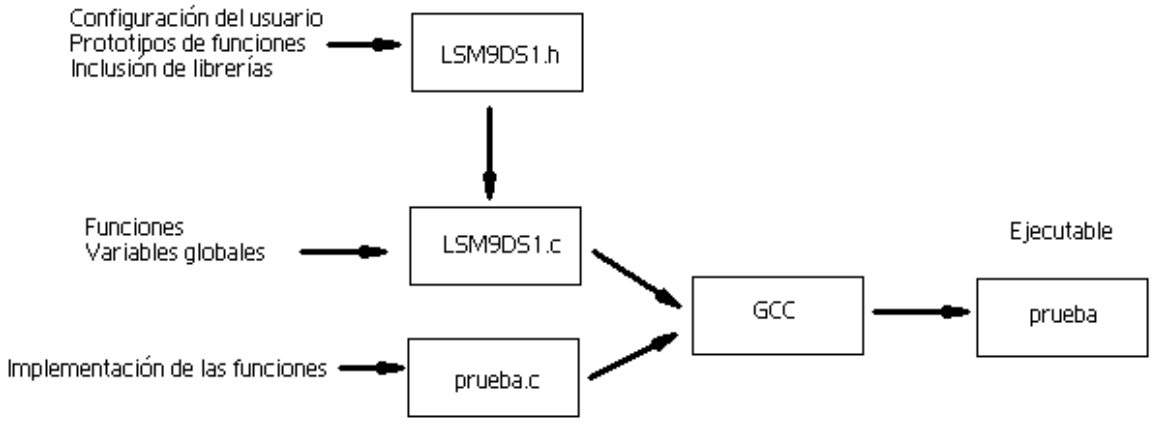

Figura 2

Mientras que en el archivo LSM9DS1.h se encuentran las opciones de configuración del IMU y este puede ser modificado para establecer la configuración que mejor se adapte a los requisitos, no es necesaria la modificación del archivo “registers.h” por parte del usuario.

Para el caso en que existan más controladores I2C en la placa controladora, utilizando “i2cdetect” se puede verificar la ruta al dispositivo. En ese caso se debe modificar la ruta en el archivo “LSM9DS1.h” en la siguiente definición:

```
#define DEV_PATH "/dev/i2c-1"
```

Se adjunta también un modelo de programa básico (con el nombre “main_solo.c”) que inicializa de forma básica el IMU e imprime los datos de los sensores en la consola. Este código fuente puede resultar útil para visualizar mejor la forma de ejecución de la inicialización y obtención de datos. Debido a su menor complejidad, es recomendable utilizar este programa para realizar cambios y pruebas en la forma de operar el sensor.

Resultados obtenidos

Los ejemplos proporcionados por Dave B. se ejecutaron sin inconvenientes. Se realiza la lectura de la presión y temperatura desde el LPS25H, se lee la humedad y temperatura desde el HTS221 y por último se prueba la escritura en la matriz de LED con una serie de demostraciones escribiendo directamente en el espacio de memoria del controlador.

Para realizaer las mediciones con el LSM9DS1, se ejecuta el programa de prueba estando el conjunto RaspberryPi + SenseHAT en reposo sobre una superficie horizontal. Se puede apreciar una salida similar a la de la figura 3, donde se puede apreciar la lectura secuencial de aceleración lineal y velocidad angular en las tres dimensiones del espacio, la temperatura a la que se encuentra el circuito integrado y una lectura normalizada del fujo de campo magnético.

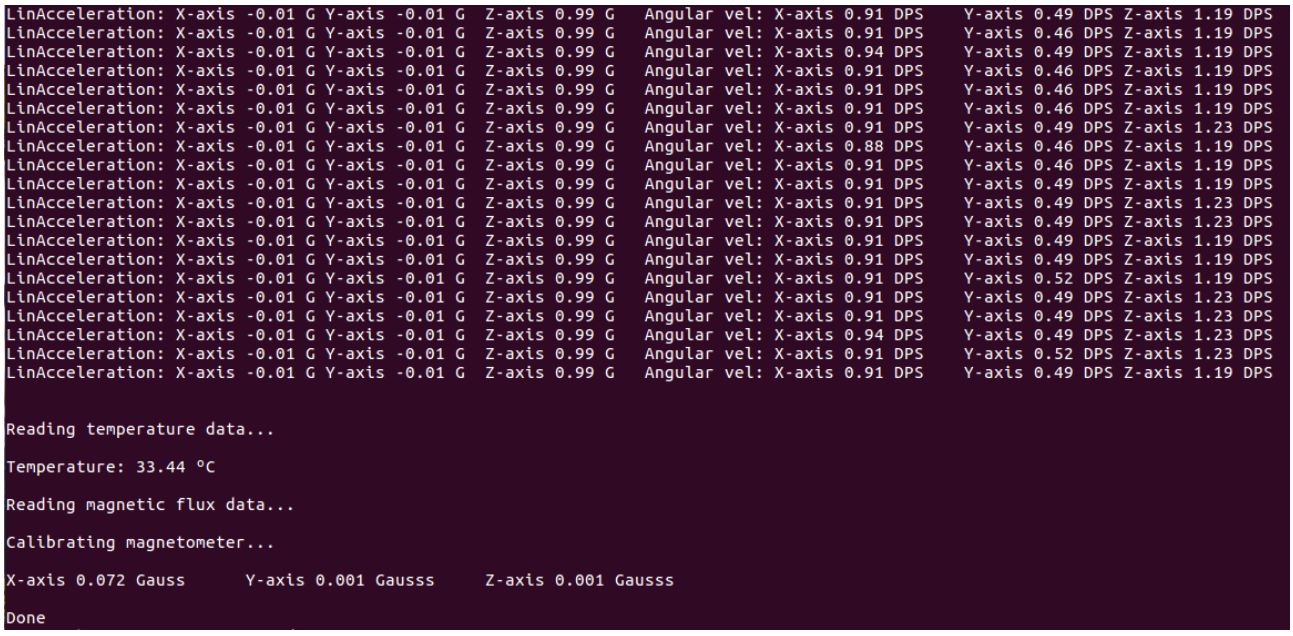

Figura 3

Nótese que si se intenta adquirir datos únicamente del girómetro (con la función “gyroRead”), probablemente devuelva datos erróneos. Una manera de evitar esto es adquirir conjuntamente los datos del acelerómetro y el girómetro utilizando la función “simultaneousRead” y si fuese necesario, descartar los datos del puntero del girómetro. Este comportamiento no se presenta cuando se ejecuta el archivo “main_solo”.

Vale aclarar que el proceso de calibración del medidor de flujo magnético puede realizarse de dos maneras:

- Referencia en cero: con la SenseHAT en reposo para tomar como referencia la posición actual.
- Promediando los valores que pueden aparecer en las 3 dimensiones: al momento de la calibración, mover aleatoriamente la SenseHAT para que el programa tome el valor medio de todas las mediciones en los 3 ejes. En este caso puede resultar necesario incrementar el número de muestras.

Consulte el anexo para mayor información sobre el proceso de calibración del magnetómetro.

No se debe ignorar que el primer bucle de obtención de datos puede contener datos erróneos como se muestra en la figura 4, por lo cual la muestra debería ser descartada.

```
pi@raspberrypi:~/LSM9DS1 $ gcc -Wall LSM9DS1.c prueba.c -o prueba
pi@raspberrypi:~/LSM9DS1 $ ./prueba


Reading linear acceleration data...

X-axis -0.01 G          Y-axis -0.01 G          Z-axis 0.98 G

Reading gyro data...

X-axis 1.23 DPS         Y-axis 1.86 DPS Z-axis -5.81 DPS

Reading both linear acceleration and angular velocity data...

LinAcceleration: X-axis -0.02 G Y-axis -0.14 G  Z-axis 0.95 G   Angular vel: X-axis 148.85 DPS  Y-axis 0.67 DPS Z-axis 147.91 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.98 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.98 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.98 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.98 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.46 DPS Z-axis 1.15 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.98 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.98 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.99 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.99 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.99 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.99 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.49 DPS Z-axis 1.19 DPS
LinAcceleration: X-axis -0.01 G Y-axis -0.01 G  Z-axis 0.99 G   Angular vel: X-axis 0.91 DPS    Y-axis 0.46 DPS Z-axis 1.19 DPS
```

Figura 4

Conclusiones

Se logró comprobar la capacidad de acceder a los registros de los distintos periféricos incorporados en la Sense HAT utilizando el bus I2C desarrollando programas en lenguaje C como se esperaba. En consecuencia esto habilita a futuros desarrollos sobre computadoras y microcontroladores más adecuados al fin específico del proyecto.

La mayor desventaja del uso de IMUs para la asistencia de navegación es que típicamente sufren de acumulación de errores. Esto se debe a que el sistema de guía se encuentra continuamente integrando la aceleración respecto al tiempo para calcular la velocidad y la posición, esto se conoce como *Dead Reckoning*, cualquier error de medición, por más pequeño que sea, se acumula en el tiempo. Esto produce un desplazamiento/drift que va incrementando la diferencia entre la posición

real y la posición indicada por el instrumento. Sistemas externos pueden utilizarse para corregir estos errores, como por ejemplo el uso de sistemas como los GNSS.

La temperatura reportada por los sensores no es precisamente la temperatura ambiente, sino la temperatura de la placa. Dado que el SoC Broadcom se encuentra directamente por debajo de la placa SenseHAT, esta suele estar más caliente que el aire alrededor, por lo cual se sugiere realizar las debidas correcciones.

Análisis del Conversor AD/DA

Utilizando una placa de conversión analógica-digital/digital-analógica formato “HAT” para RaspberryPi (véase la figura 5), se intenta estimar por medio de un generador, un osciloscopio y utilizando un bypass implementado en lenguaje C, el comportamiento en frecuencia del conversor AD/DA, así como el rendimiento de las librerías de Linux y el SoC Broadcom de la RaspberryPi,.

Atributos de la placa de expansión:

- Conversor AD ADS1256 con 8 canales de 24 bit (o 4 canales de entrada diferencial), tasa de muestreo de 30 Ksamples/s
- Conversor DA DAC8532, de 2 canales de 16 bit de resolución
- Pinheaders para conexión de entrada analógica (compatible con la forma del conector estándar de Waveshare)
- Interfaz de E/S a través de conectores con tornillos para señales digitales y analógicas
- Incluye un circuito para demostración de las capacidades de adquisición y conversión de muestras (potenciómetro, LDR y leds)

- Fabricada por Waveshare
- Factor-forma tipo HAT
- Requiere alimentación de 5 y 3,3 v
- Utiliza 8 pins GPIO
- La comunicación con el dispositivo se realiza a través del bus SPI

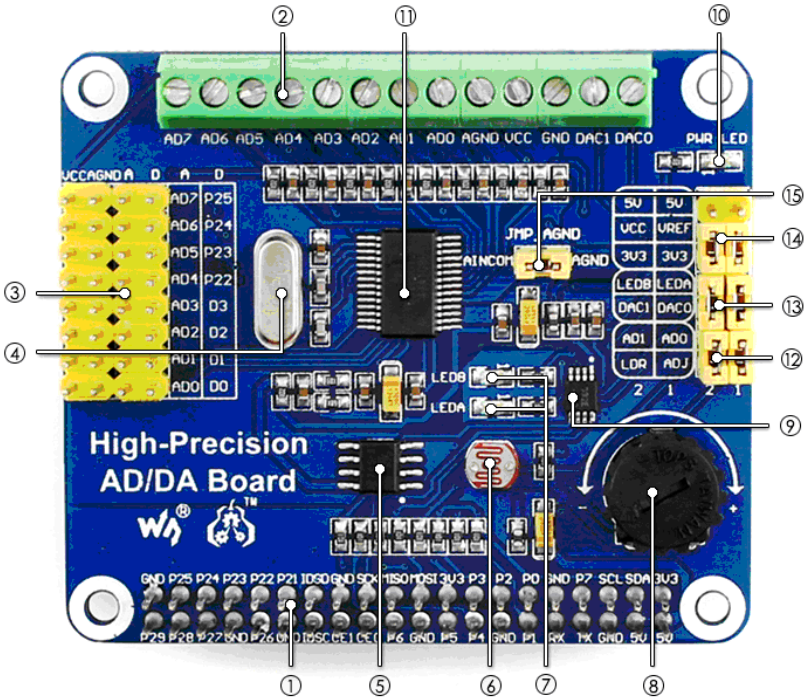

Figura 5

1. Interfaz GPIO
2. AD/DA E/S : terminales con tornillos
3. Entrada AD : pinheader compatible con los sensores de Waveshare
4. Oscilador de 7,68 MHz
5. LM285-2.5 : Voltaje de referencia del integrado del ADC
6. LDR / foto-resistor
7. LEDs de salida digital
8. Potenciómetro de 10Kohm
9. DAC8532 : DAC de 16 bit de resolución y 2 canales
10. Indicador de power
11. ADS1256 : ADC de 24 bit de resolución y 8 caales (4 de entrada diferencial)
12. Jumper de prueba del ADC
13. Jumper de prueba del DAC
14. Jumper de selección de voltaje
15. Jumper de referencia de tierra del ADC : cuando el AD tiene una sola entrada, AINCOM es el terminal de referencia que puede ser conectado al GND o a una referencia externa.

Método experimental

Para las pruebas se utilizó la siguiente configuración:

Configuración de Jumpers:

Alimentación a 3,3V: conectar el pin de 3,3V con el de VCC. (14)
Voltaje de entrada de referencia a 3,3V: conectar el pin de 3,3V con el de VREF. (14)
Los jumpers de prueba del AD (12) se desconectan
Los jumpers de prueba del DA (13) se desconectan
Se conecta AINCOM a AGND (15)
Se conecta la salida del generador al pin AD0 (12)
Se conecta una punta del osciloscopio al pin DA0 (13)
Se conectan los GND a cualquier pin GND del GPIO (1)

Instalación del software

Los ejemplos utilizados (que luego fueron modificados) requieren que se encuentre instalada la librería bcm2835 la cual provee funciones para leer las entradas digitales, así como poder configurar las salidas digitales utilizando SPI e I2C, además de posibilitar el acceso a los timers del sistema (ver anexo).

La instalación de las librerías se realiza de forma simple. Solamente es necesario descargar, descomprimir, construir e instalar el paquete

```
tar zxvf bcm2835-1.xx.tar.gz
cd bcm2835-1.xx
./configure
make
sudo make check
sudo make install
```

Ejecución de las pruebas

Se modificaron los ejemplos provistos por Waveshare para realizar las mediciones de respuesta en frecuencia y jitter de la conversión.
Los parámetros modificados fueron desde la frecuencia del bus SPI hasta la eliminación de algunas rutinas de comprobación de existencia de dispositivos en otros canales distintos al que se utilizó.

Resultados

Se realizaron pruebas en primera instancia del bypass con los parámetros por defecto del ejemplo “AD-DA_test.c” provisto por el fabricante y se obtuvo el siguiente resultado (véase figura 6) utilizando una entrada de 30 Hz montada sobre una componente de DC de 2,5V y una tasa de muestras de 30 KHz en el conversor.

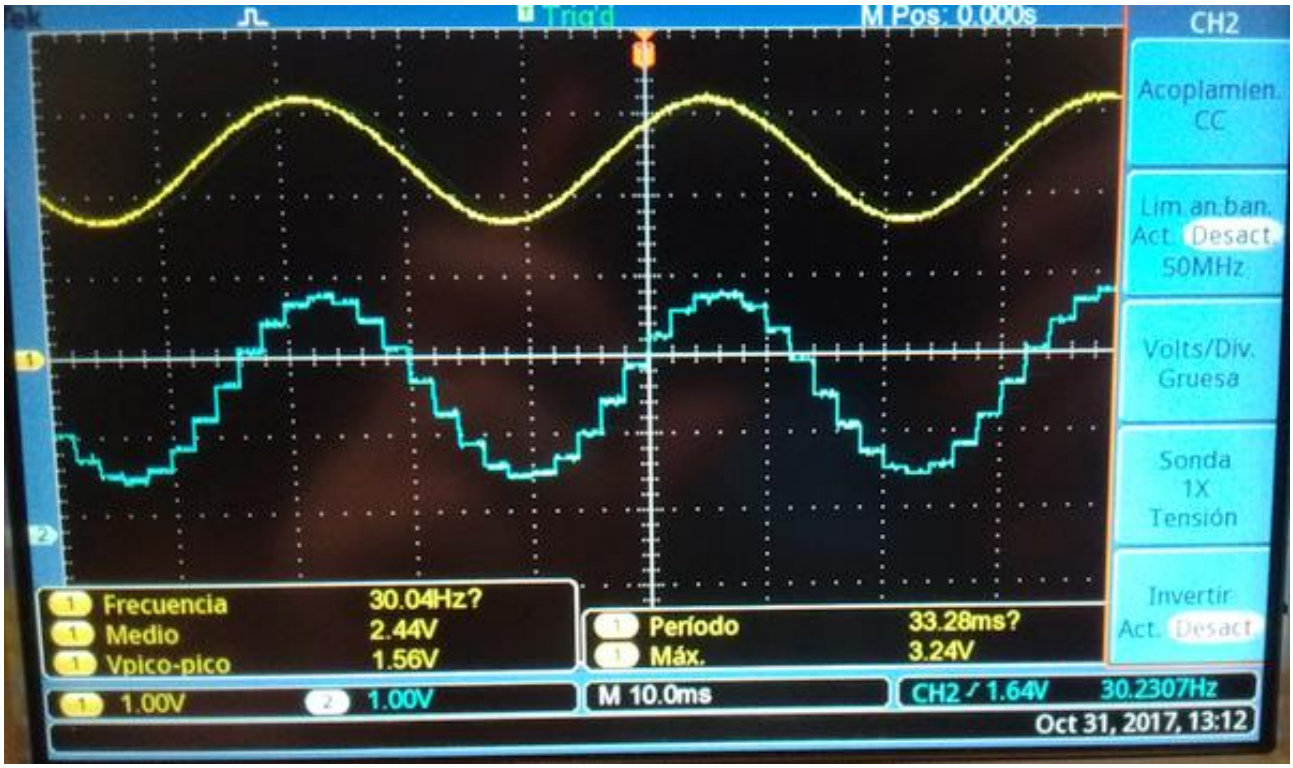

Figura 6

Para probar la capacidad de escritura del DAC se ejecuta un bit-shift de 8 bit para reducir al mínimo la influencia de la carga del CPU en el resultado. Se puede observar el resultado con los valores por defecto del divisor de frecuencia del bus SPI en la figura 7 y luego el resultado con el divisor de frecuencia en un valor más bajo, el cual fija la frecuencia del bus a 3,125 MHz, esto se puede apreciar en la figura 8. Se probaron valores más bajos del divisor pero no se recomiendan ya que no es estable su funcionamiento por encima de los 4 MHz en la Raspberry Pi 3.
El valor máximo de valores convertidos por el DAC que se encontró fue de 64000 muestras por segundo (ver figura 8).

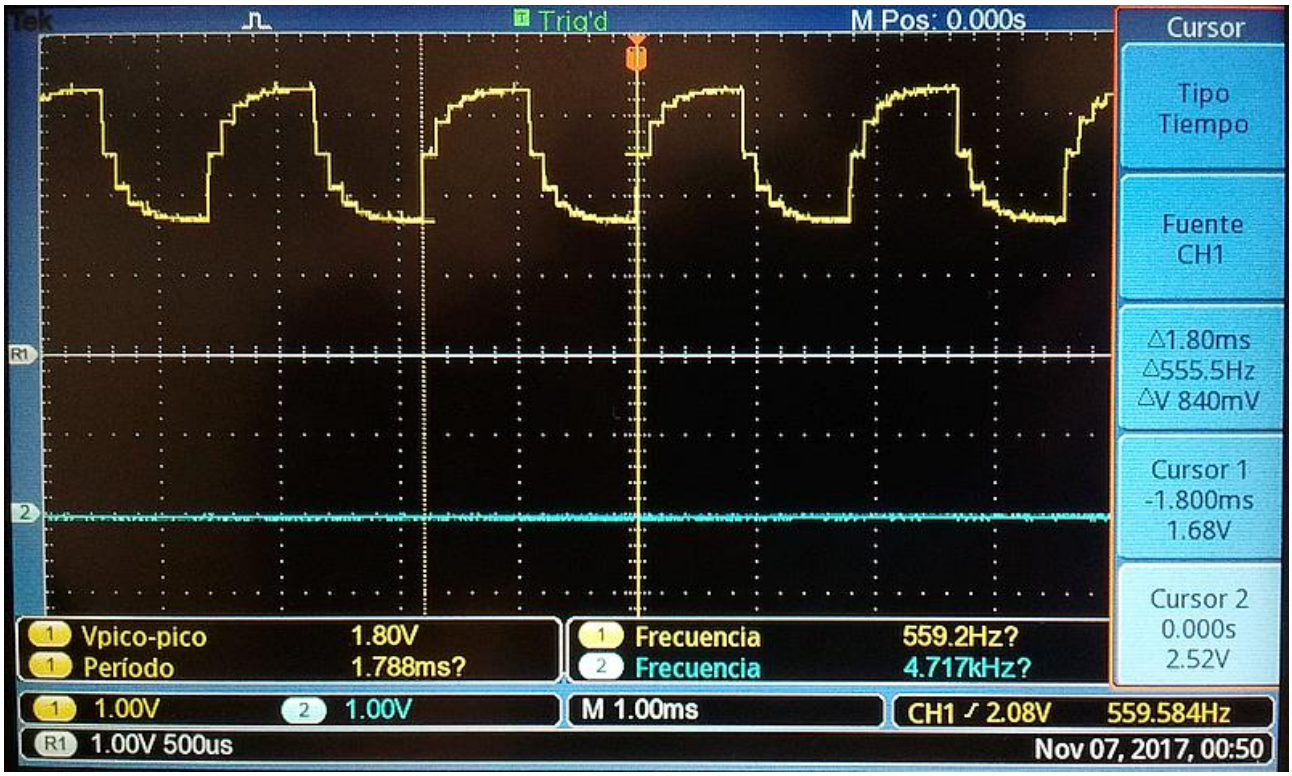

Figura 7

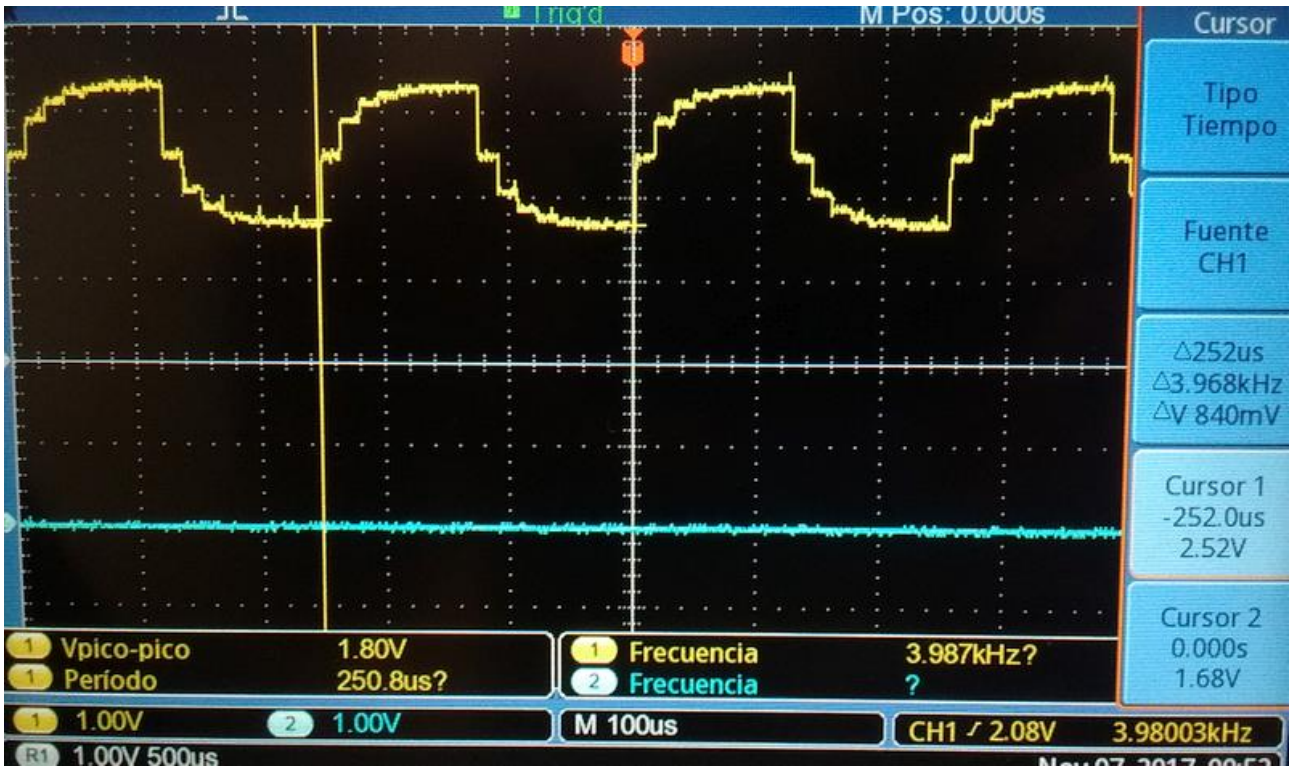

Figura 8

Se verificó la existencia de una latencia mayor que el tiempo entre muestras del ADC, en la cual se repetía el mismo valor cuatro veces, reduciendo de esta forma la tasa de muestreo efectiva. Esto se logró visualizar con un bucle que compara el valor de la conversión con la muestra contra el valor de la muestra del bucle anterior. Véase la figura 9.

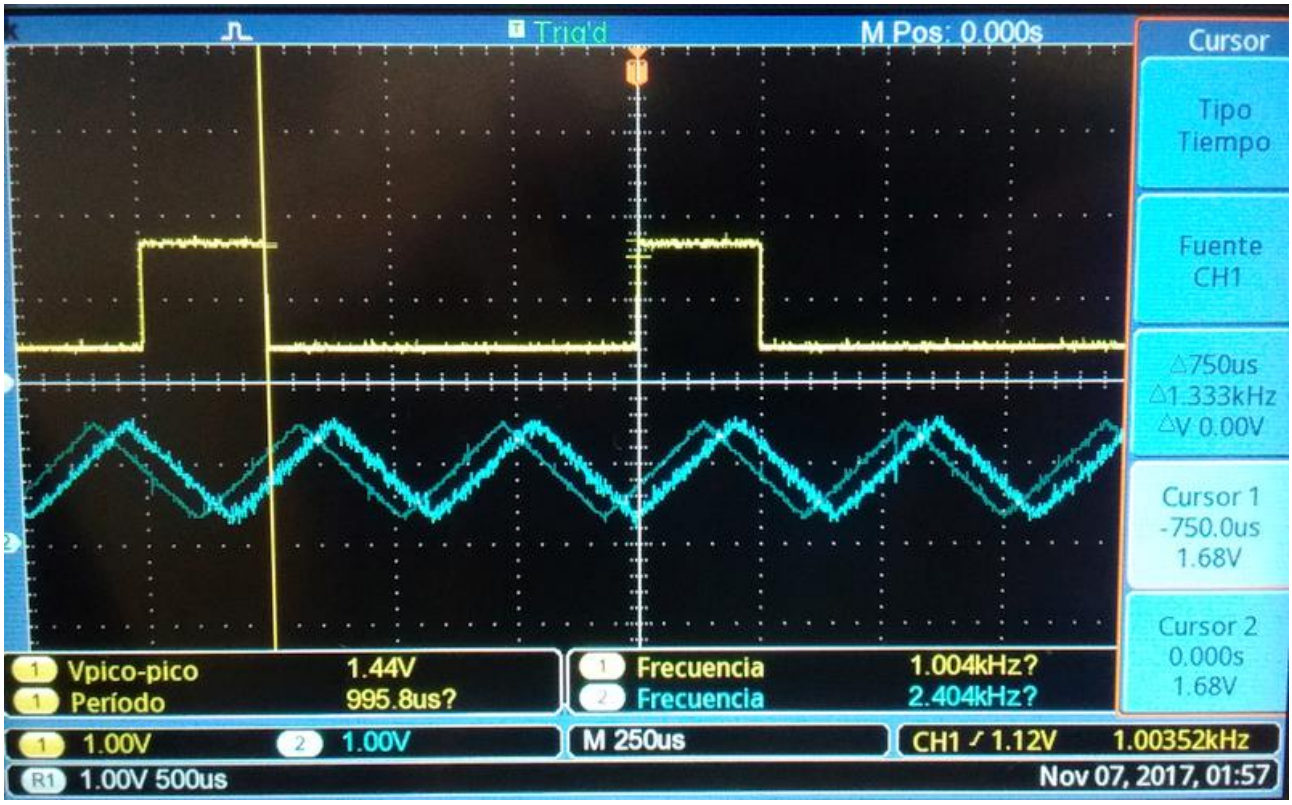

Figura 9

Este delay se originaba en la función ADS1256_ISR(), la cual esperaba un cambio de canal muestreado y una posterior sincronización. En la función ADS1256_Scan() se comentó la línea que llama a la función que producía la demora y que no es necesaria para la aplicación actual, logrando ahora una tasa de muestras de aproximadamente 24000 muestras por segundo (véase figura 10).

Figura 10

Se hace notar el efecto del muestreo sobre la respuesta en frecuencia, ya que a mayores frecuencias, la amplitud de la muestra de salida comienza a decrecer. Remítase a la hoja de datos del conversor Analógico-Digital que se encuentra en la dirección que muestra el anexo para más detalles sobre la respuesta en frecuencia del conversor.

Conclusiones

Se logró estimar el número de muestras máximo procedentes de un solo canal del conversor AD/DA que se puede obtener y procesar en la Raspberry Pi 3 utilizando un sistema operativo que no está optimizado para operaciones en tiempo real. Este número máximo de muestras es de aproximadamente 24000 muestras por segundo. Debido a la capacidad de tomar muestras de varios canales, se debe prestar atención a los canales habilitados para tomar las muestras, ya que el proceso de conmutar entre canales toma un periodo de tiempo relativamente largo. Esto no es crítico en implementaciones donde se utilice una tasa de muestreo menor a 10 muestras por segundo.
Se recomienda entonces que para conversiones de alta velocidad se utilice únicamente un solo canal de entrada analógica y un divisor de frecuencia del bus SPI que logre una frecuencia efectiva de alrededor de 4 MHz.
Cabe señalar también que la placa Raspberry Pi 2 soporta frecuencias de bus SPI ligeramente mayores que la Raspberry Pi 3, por lo que debería repetirse el conjunto de pruebas en el modelo anterior.
No resultó posible observar valores de jitter apreciables en las pruebas realizadas, pero se sugiere realizar las pruebas con un sistema operativo de tiempo real para minimizar la posibilidad de alteraciones en la latencia del sistema.
El integrado que realiza la conversión Analógica-Digital cuenta con un pin de sincronismo externo, que en el caso de la placa de expansión AD/DA para Raspberry Pi no se encuentra conectado, por lo tanto se debe usar el comando SYNC/RESET seguido por un comando de WAKEUP para completar la sincronización.

Anexos

Esquemático de la Sense-HAT V1:
https://www.raspberrypi.org/documentation/hardware/sense-hat/images/Sense-HAT-V1_0.pdf

La hoja de datos del sensor de presión/temperatura (LPS25H) provista por el fabricante (ST):
www.st.com/resource/en/datasheet/dm00066332.pdf

La hoja de datos del sensor de humedad/temperatura (HTS221) provista por el fabricante (ST):
www.st.com/resource/en/datasheet/hts221.pdf

La hoja de datos del sensor inercial (LSM9DS1) provista por el fabricante (ST):
http://www.st.com/resource/en/datasheet/lsm9ds1.pdf

La hoja de datos del controlador de LEDs (LED2472G) provista por el fabricante (ST):
www.st.com/resource/en/datasheet/led2472g.pdf

Para más detalles sobre el firmware AVR, el mismo se encuentra disponible en GitHub:
https://github.com/raspberrypi/rpi-sense

Más información sobre i2c-dev:
https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/i2c/dev-interface

Los ejemplos en lenguaje C desarrollados por Dave B. se encuentran disponibles en GitHub:
https://github.com/davebm1/c-sense-hat/

El código en C++ de SparkFun que se utilizó como base para el desarrollo del controlador del IMU en C, se encuentra disponible en GitHub:
https://github.com/sparkfun/LSM9DS1_Breakout/blob/master/Libraries/Particle/firmware/SparkFunLSM9DS1.cpp

Las librerías Python e instrucciones para la calibración del magnetómetro se encuentran en el sitio de documentación de Raspberry Pi:
https://www.raspberrypi.org/documentation/hardware/sense-hat/

Esquema de la placa AD/DA proporcionado por el fabricante (WaveShare):
http://www.waveshare.com/wiki/File:High-Precision-AD-DA--Schematic.pdf

La documentación y enlaces de descarga de las librerías de Broadcom pueden encontrarse en:
http://www.airspayce.com/mikem/bcm2835/

Los códigos de ejemplo para la placa AD/DA son proporcionados por el fabricante de la misma:
https://www.waveshare.com/wiki/File:High-Precision-AD-DA-Board-Code.7z

El esquemático del conversor Analógico-Digital es proporcionado por WaveShare:
https://www.waveshare.com/w/upload/0/03/ADS1256.pdf