El magnetómetro

Es el dispositivo que sirve para cuantificar en fuerza o dirección de la señal magnética de una muestra, el magnetómetro AK8975 es capaz de detectar el magnetismo terrestre en el eje X, eje Y, y eje Z. La comunicación se realiza mediante el bus de datos I2C, este magnetómetro cuenta con la posibilidad de cambiar la dirección del dispositivo mediante los pines CAD0 y CAD1, cuando ambos pines se encuentran conectados a GND la dirección del dispositivo es 12 en decimal o 0C en hexadecimal como lo es en este caso.

Si intentamos ingresar directamente al dispositivo apuntando a la dirección 12, tendremos un error de comunicación por parte de Python, debido a que el magnetómetro no se encuentra conectado directamente al bus de datos, si no que se emplean los pines auxiliares del MPU6050 XDA y XCL. Como se muestra en la imagen 1.

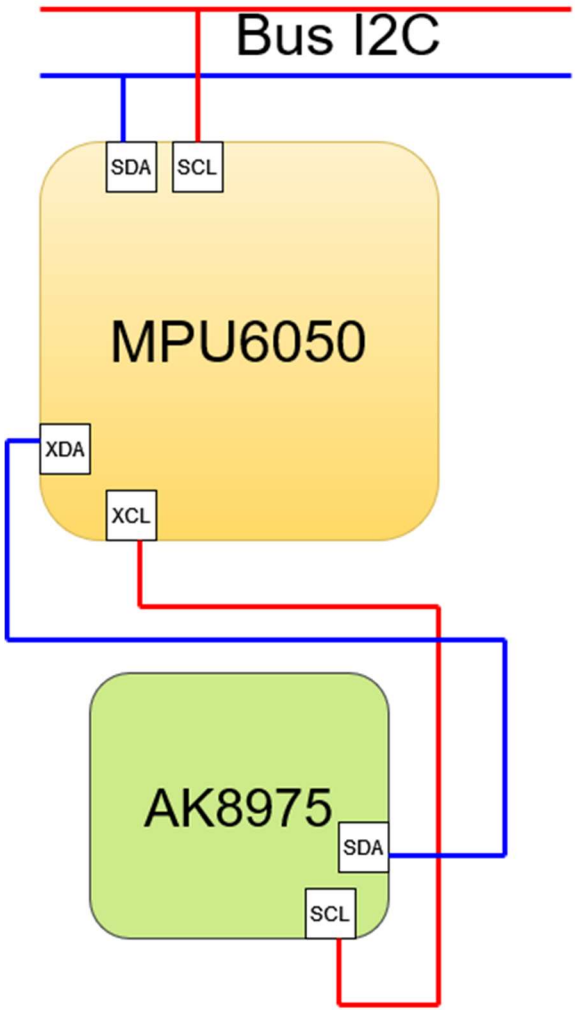

Imagen 1. Conexión del magnetómetro.

Debido a la conexión mostrada en la imagen 1, es necesario habilitar los registros 55 y 107 para habilitar los pines auxiliares del Bus de datos I2C.

Biblioteca kairos_AK8975.py

Desarrollo de la biblioteca se comienza importando las bibliotecas I2C, delay y array.

```
from pyb import I2C
from pyb import delay
from array import array
```

Se crea el objeto i2c y se establece el modo de operación y la velocidad del bus de datos.

```
i2c = I2C(1)
i2c.init(I2C.MASTER, baudrate = 100000)
```

Se crea la clase llamada AK8975, esta clase en su interior contendrá 3 métodos, el método init, el método get_x y el método get_y. Es importante aclarar que el magnetómetro es capaz de leer el eje z, pero para la aplicación requerida solo es necesario obtener dos ejes, si se desea leer el eje z, se puede agregar un nuevo método y leer los registros 7 y 8 del magnetómetro y realizar los mismos pasos con respecto al eje x y al eje y.

```
class AK8975:
```

El método init, se encarga de escribir en los registros 55 (0x37) y 107 (0x6B) del MPU 6050. El registro 55 es el encargado de activar el bypass del MPU6050, de debe de habilitar el bit 1 MSB (Bit Mas Significativo), para un mejor entendimiento se utiliza el código binario en la escritura de registros. El bypass habilita los pines auxiliares XDA y XCL del MPU6050 habilitando el magnetómetro y haciendo posible su escritura y lectura sin la necesidad de utilizar registros del MPU6050. Por su parte el registro 107 define la frecuencia de oscilación del MPU6050, es muy importante habilitar este registro ya que a pesar de activar el registro del bypass si no se habilita el registro 107 el magnetómetro no es detecto por el bus i2c. La frecuencia de oscilación se establece en 8 Mhz. La dirección del MPU6050 es 104 (0x68).

```
    def init():
        i2c.mem_write(0b00000010, 0x68, 0x37)
        i2c.mem_write(0b00000000, 0x68, 0x6B)
```

Una vez realizada es configuración ya es posible acceder al magnetómetro directamente desde el bus de datos I2C.

El método get_x se encarga de leer dos bytes del magnetómetro concatenarlos y retornarlos, como se leen dos bytes la cantidad leída se encuentra en el rango de 0 a 65535.

El método comienza declarando un arreglo de dos bytes del tipo entero, se escribe en el registro 10 (0x0A) el modo de operación, este se establece como Single measurement mode, para activarlo se habilita el bit 0 (MSB). La dirección del magnetómetro es 12 (0x0C). Un pequeño inconveniente de este magnetómetro es que solo cuenta con ese modo de operación, en donde solo se lee un valor único y no de manera continua, para tener lecturas continuas es necesario escribir cada en el registro 10 cada vez que sea necesario leer el eje x. Para obtener los valores del eje x es necesario leer el registro 3 (0x03) y el registro 4 (0x04). Ambos bytes se concatenan y si el resultado es mayor a 32768 se restan 65536. Esto se hace para tener un valor de salida del eje x entre -32768 y +32768.

```
def get_x():
    datos = array('B', [0] * 2)

    i2c.mem_write(0b00000001, 0x0C, 0x0A)
    delay(10)
    datos = i2c.mem_read(2, 0x0C, 0x03)
    eje_x = (datos[1] << 8) | datos[0]

    if (eje_x > +32768):
        eje_x = eje_x -65536
```

Para obtener los datos del eje y el procedimiento es el mismo, solo cambian los registros a leer, estos serán el registro 5 (0x05) y el registro 6 (0x06).

```
def get_y():
    datos = array('B', [0] * 2)

    i2c.mem_write(0b00000001, 0x0C, 0x0A)
    delay(10)
    datos = i2c.mem_read(2, 0x0C, 0x05)
    eje_y = (datos[1] << 8) | datos[0]

    if (eje_y > 32768):
        eje_y = eje_y -65536

    return(eje_y)
```

Con estos 3 métodos es posible realizar la brújula digital. Para invocarlos se utiliza la instrucción en el archivo “main.py”.

```
from kairos_AK8975 import AK8975
```

Polo norte geográfico y magnético.

El planeta tierra cuenta dos polos norte, el polo norte geográfico y el polo norte magnético. El polo norte geográfico está situado en medio del Océano Ártico y es el punto de convergencia para el origen de los meridianos, este polo norte es al que siempre se refieren en los libros de texto.

El polo norte magnético es un punto en la Isla Ellesmere en el norte de Canadá donde las líneas de atracción del norte entran a la tierra. Esto significa que el polo norte magnético y el polo norte geográfico se encuentran en diferentes lugares existe una diferencia de aproximadamente 500 km.

Una brújula análoga apuntara al polo norte magnético, la diferencia entre el polo norte magnético y polo norte geográfico se llama declinación magnética.

Ángulo Azimut

Es el ángulo de una dirección contando en el sentido de las agujas del reloj a partir de del norte magnético. Cuando la aguja de brújula apunta hacia el norte es decir 0° entonces el azimut magnético es de 0°. Para obtener este ángulo se utiliza la función trigonométrica tangente inversa o arco tangente.

$$\acute{a}ngulo\ azimut = \tan^{-1}\frac{y}{x} + declinacion\ magnetica$$

Brújula digital

Abrimos el archivo "main.py" y procedemos a importar las siguientes librerías.

```
from kairos_AK8975 import AK8975
from pyb import delay
import math
```

Invocamos al método para inicializar las configuraciones correspondiente al magnetómetro y declaramos la declinación magnética, para obtener la declinación ingresamos a la siguiente página web.

https://www.magnetic-declination.com/

Dentro de la página web elegimos nuestra ubicación y guardamos el parámetro "Magnetic Declination", convertimos ese valor a radianes en la siguiente página web.

https://es.planetcalc.com/71/

Una vez realizada la conversión a radianes se guarda dicho valor en la variable declinación.

```
AK8975.init()

declinacion = 0.0668
```

En el ciclo while se invocan los métodos "get_x" y "get_y" para obtener los valores del eje x y del eje y. El valor del eje x se guarda en la variable x, y el valor del eje y se guarda en la variable y.

Se calcula el angulo del eje y con respecto al polo norte magnético y se suma la declinación magnética, para encontrar el norte geográfico.

Para mostrar la información como si se tratara de un brújula análoga se convierten los radianes negativos a positivos y se convierten los radianes a grados.

```
while True:

    x = AK8975.get_x()
    y = AK8975.get_y()

    ang_rad = math.atan2(y, x) + declinacion

    if(ang_rad < 0):
        ang_rad = ang_rad + 2*math.pi

    grados = int(ang_rad * 180/math.pi)

    print(grados)

    delay(100)
```

Los grados se imprimen en la terminal

Una vez que ejecutamos el código se mostraran grados que van desde los 0 grados hasta los 360 grados. La interpretación es la siguiente.

Norte ---> 0°

Este ---> 90°

Sur ---> 180°

Oeste ---> 270°

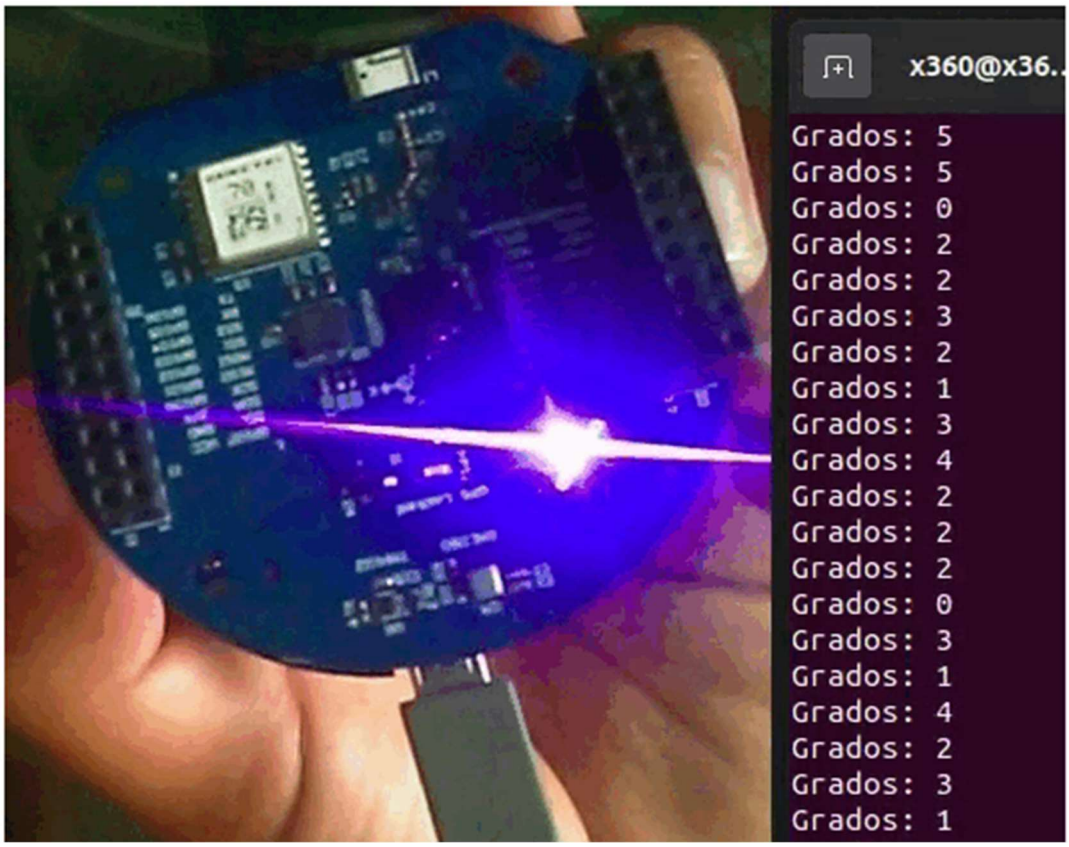

Imagen 2. Brújula apuntando hacia el norte.