Universidad De Málaga

E.T.S Ingeniería Informática

Ingeniería de Computadores, 3A

PROYECTO FINAL
Manual de uso de acelerómetro Adafruit MMA8451

Diseño con Microcontroladores

Francisco Javier Cano Moreno

1 de Junio de 2023

Contents

1 Introducción

El acelerómetro es un dispositivo que se encarga de medir la aceleración lineal y angular y se utiliza en muchísimos dispositivos de nuestro día a día. Gracias a ello es capaz de detectar golpes, movimiento o impacto. Algunos de los usos que se la da son:

- **Relojes Fitness** $\longrightarrow$ Ayudan a medir el ejercicio físico de la persona que lo lleva.
- **Smartphones/tablets** $\longrightarrow$ Detección de la orientación de la pantalla
- **Navegación** $\longrightarrow$ Ayudan al GPS a tener una mejor precisión en el seguimiento.

Para poder utilizar el acelerómetro, se ha creado una API que se encarga de realizar la comunicación con el acelerómetro y de ofrecernos los registros necesarios para poder leer y escribir en ellos de una forma sencilla.

El acelerómetro dispone de 8 pines:

- **Vin** $\longrightarrow$ Pin de alimentación. Se puede conectar a los 5V de la placa directamente.
- **GND** $\longrightarrow$ Pin de tierra que se conecta al GND de la placa.
- **3v3** $\longrightarrow$ Pin de alimentación pero de 3.3V.
- **I1** $\longrightarrow$ Línea de interrupción 1.
- **I2** $\longrightarrow$ Línea de interrupción 2.
- **SDA** $\longrightarrow$ Canal I2C por el que se realiza el envío de datos. Se conecta al pin SDA de la placa.
- **SCL** $\longrightarrow$ Canal por el que se envía la señal de reloj de I2C, la cual genera la placa. Se conecta al pin SCL de la placa.
- **A** $\longrightarrow$ Este pin se utiliza para darle otra dirección I2C al acelerómetro (0x1C) ya que por defecto utiliza como dirección 0x1D. Para cambiarla basta con conectar este pin al GND.

Las líneas de interrupción se pueden utilizar para saber cuando los datos están listos para leer, cuando se detecta un impacto o si hay movimiento. Gracias a las dos líneas de interrupción, se pueden tener configuradas dos interrupciones diferentes.

2 Comunicación con MMA8451

2.1 I2C

El acelerómetro utiliza un protocolo de comunicación serie especial llamado I2C. Este se encarga del envío de datos entre dispositivos digitales utilizando el mismo bus, una de sus principales ventajas. Utiliza dos cables de comunicación:

- **SCL**: Línea por la que se transmite la señal de reloj, que se encarga de sincronizar la transferencia de los datos.
- **SDA**: Línea por la que se envían los datos.

Cada dispositivo tiene una dirección asignada, en el caso del MMA8451 es la 0x1D pudiéndose cambiar a 0x1C si lo deseamos, y tiene un funcionamiento basado en maestro y esclavo. El maestro se encarga de controlar la señal de reloj y de iniciar y de parar la comunicación. Por otro lado, el esclavo se encarga de recibir las órdenes del maestro, que pueden ser escribir en cierto registro o leer de cierto registro.

En el STM32, debemos configurar el I2C que queremos en la interfaz gráfica de configuración activando los pines de I2C. Ello nos aporta un manejador con el que podemos utilizar diferentes funciones de la API de STM32 que se encargan de gestionar toda la comunicación I2C en el nivel más bajo, es decir, inicio de la comunicación y fin, control de llegada de paquetes, envío de paquetes...Además, la API que se ha creado, permite abstraer aún más esta gestión, como veremos en el siguiente apartado.

2.2 Funciones de Comunicacion

2.2.1 write_register

- **Nombre de la función**: void write_register(uint8_t reg, uint8_t value, I2C_HandleTypeDef hi2c1, UART_HandleTypeDef huart2)
- **Descripción**: Recibe como parámetros el registro en el que se quiere escribir, el valor que se quiere escribir, el manejador I2C y el manejador de envío por el puerto serie. La función se encarga de escribir dicho valor en el registro indicado del MMA8451.
- **Valor de retorno**: Ninguno.

2.2.2 read_register

- **Nombre de la función**: uint8_t read_register(uint8_t reg, I2C_HandleTypeDef hi2c1, UART_HandleTy huart2)
- **Descripción**: Recibe como parámetros el registro que se quiere leer, el manejador I2C y el manejador de envío por el puerto serie. La función se encarga de leer el registro indicado en el MMA8451.
- **Valor de retorno**: Devuelve el byte leído.

3 API

3.1 Configuración del MMA8451

El MMA8451 tiene tres modos de funcionamiento, los cuales se configuran en el registro 0x2A (defindo en el header como MMA8451_REG_CTRL_REG1) modificando el primer bit, que indica si es el modo activo o no, y en el registro 0x2B (defindo en el header como MMA8451_REG_CTRL_REG2) modificando el tercer bit, que indica si se activa el auto sleep.

- **Standby (00)**: En este modo, el MMA8451 está encendido pero no tiene todas las funcionalidades activas. En este caso, las partes analógicas y los relojes internos están deshabilitadas.
- **Wake (01)**: Es un modo donde el MMA8451 tiene un funcionamiento normal y no se desconecta hasta que se cambie el modo o se desconecte.

- **Sleep (10)**: Este modo es igual que el anterior pero con una adición. Cuando el MMA8451 lleva un tiempo sin recibir interrupciones, se duerme para ahorrar energía. Estas interrupciones son las comentadas en el apartado de introducción.

Por otro lado, tenemos configuraciones más concretas:

- **Frecuencia de muestreo**: Es el periodo de muestreo de los datos. Adopta 3 bits de periodos diferentes que van desde los 1.56 Hz a los 800 Hz. Por defecto, utiliza 800 Hz y se configura en el 0x2A (defindo en el header como MMA8451_REG_CTRL_REG1).
- **Escalado de los datos**: Es el rango máximo de aceleración que puede tomar el acelerómetro. En función de las aceleraciones que se vayan a medir, elegiremos un rango u otro, por ejemplo, si queremos medir aceleraciones pequeñas, elegiremos un rango más pequeño para que sea más preciso. Se configura en el registro 0x0E (definido en el header como MMA8451_REG_XYZ_DATA_CFG).
- **Ruido**: El modo de ruido reducido se puede activar en el mismo registro de la frecuencia de muetreo y sirve para reducir el ruido en las mediciones para obtener más precisión en ellas.

3.2 Funciones de la API

3.2.1 Header

El header de la API define datos útiles para utilizar cuando se trabaje con el MMA8451. Estos datos pueden ser los valores de la gravedad en distintos planetas, factores de conversión de grados a radianes, los registros que se han utilizado en la elaboración de las funciones, distintos valores que pueden tomar esos registros, el valor por defecto de la dirección I2C del MMA8451, los prototipos de las diferentes funciones y la estructura **mma8451_t**, que guarda los valores de la velocidad y la aceleración en los ejes del acelerómetro y su orientación.

3.2.2 init_MMA8451

- **Nombre de la función**: void init_MMA8451(I2C_HandleTypeDef hi2c1, UART_HandleTypeDef huart2)
- **Descripción**: Inicializa el MMA8451 a una configuración por defecto.
- **Valor de retorno**: Ninguno.

3.2.3 get_orientation

- **Nombre de la función**: void get_orientation(mma8451_t *data, I2C_HandleTypeDef hi2c1, UART_HandleTypeDef huart2)
- **Descripción**: Modifica la estructura data con la nueva orientación leída.
- **Valor de retorno**: Ninguno.

3.2.4 read

- **Nombre de la función**: void read(mma8451_t *data, I2C_HandleTypeDef hi2c1, UART_HandleTypeDef huart2)
- **Descripción**: Actualiza la estructura data con los nuevos valores de la velocidad y la aceleración leídas del acelerómetro.
- **Valor de retorno**: Ninguno.

3.2.5 set_range

- **Nombre de la función**: void set_range(mma8451_range_t range, I2C_HandleTypeDef hi2c1, UART_HandleTypeDef huart2)
- **Descripción**: Actualiza el rango del MMA8451 con el nuevo rango.
- **Valor de retorno**: Ninguno.

3.2.6 get_range

- **Nombre de la función**: mma8451_range_t get_range(I2C_HandleTypeDef hi2c1, UART_HandleTypeDef huart2)
- **Descripción**: Lee el rango del MMA8451.
- **Valor de retorno**: Devuelve el rango leído.

3.2.7 print_info

- **Nombre de la función**: void print_info(mma8451_t data, UART_HandleTypeDef huart2)
- **Descripción**: Muestra la información de la estructura data.
- **Valor de retorno**: Ninguno.

3.3 Mejoras

Algunas de las mejoras que se pueden introducir es la capacidad de gestión de un evento como el movimiento o la detección de caída. Esta implementación sería muy útil para dispositivos de gestión de un airbag.

3.4 Prueba de funcionamiento

Esto sería un registro del funcionamiento del MMA8451:

```
POSICION

x = -614

y= 212
```

z= 1941

ACELERACION

x = -2.940080

y= 1.015141

z= 9.294291

ORIENTACION -> Landscape Left Front

POSICION

x = 224

y= -1310

z= 1713

ACELERACION

x = 1.072602

y= -6.272809

z= 8.202535

ORIENTACION -> Portrait Up Front