INTERFAZ POR I2C DEL MICROPROCESADOR P8X32A CON EL SENSOR MP6050

Interfaz por I2C del Microprocesador P8X32A con el Sensor MP6050

MC. Francisco Alejandro Chávez Estrada[1,2]
frchaveze1400@alumno.ipn.mx
fachavez@ittla.edu.mx
Dr. Juan Carlos Herrera Lozada[1]
jcrls.ipn@gmail.com
Dr. Jacobo Sandoval Gutiérrez[3]
jsandoval@correo.ler.uam.mx
Ing. López Jiménez, Efren[1]
jmzelectronica@gmail.com
Ing. Carboney Palafox, Leslie[1]
leslie.cidetec@gmail.com
Ing. José Juan Torres Duarte[1]
josejuan_td@hotmail.com

[1]*Instituto Politécnico Nacional , Centro de Innovación y Desarrollo Tecnológico en Cómputo (IPN-CIDETEC)*
[2]*Instituto Tecnológico de Tlalnepantla (ITTLA) del tecnológico Nacional del México (TecNM)*
[3]*Universidad Autónoma Metropolitana Lerma (UAML)*

Resumen

Los nuevos sensores como el MP6050 fabricados por las técnicas de sistema micro electrónico mecánicos (MEMES) y los nuevos protocolos de comunicación como el I2C permiten simplificar además de mejorar los desarrollos en hardware y software de la interfaz este trabajo muestra que a partir de los datos técnicos de los dispositivos se desarrolla un programa en C++ que permite leer y escribir datos del acelerómetro y giroscopio en X, Y y Z del sensor MP6050 designado como esclavo en una interfaz I2C conectado a un procesador maestro P8X32A, la comunicación entre el procesador - sensor es muy sencilla y confiable lo que permite simplificar conexiones y desarrollar un software sencillo que establece la comunicación al direccionar al sensor y este responde reconociendo la llamada, además permite configurar los parámetros del sensor para aumentar la confiablidad de los datos capturados. Este proyecto tiene aplicación en robot para su localización y de acuerdo a su avance, se identifica la trayectoria de desplazamiento por medio del giroscopio.

Introducción

El término I2C proviene de Inter-Integrated Circuit, que sería abreviado IIC. A menudo se pronuncia I-cuadrado-C, aunque I-dos-C también se entiende. Es un protocolo de comunicación serial desarrollado por Phillips Semiconductors, se creó para poder comunicar varios chips al mismo tiempo dentro de los televisores.

Comentado [v1]: http://learn.teslabem.com/fundamentos-del-protocolo-i2c-aprende/2/

I2C integra lo mejor de los protocolos SPI y UARTI2C como indica Leens F. [1], es una interfaz de dos hilos compuesta por los datos en serie de señales (SDA) y el reloj en serie (SCL). Las líneas son de drenaje abierto y bidireccional. En una implementación de interfaz I2C generalizada, debe existir un dispositivo maestro y un esclavo. El dispositivo maestro coloca la

dirección del esclavo en el bus y el dispositivo esclavo con la dirección coincidente reconoce al maestro de acuerdo a lo que nos indica Mazidi et al. [2], se muestra Fig.1.

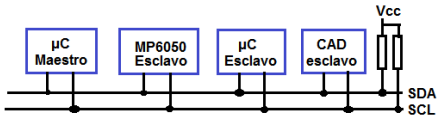

Figura 1. Conexionado de dispositivos en protocolo I2C.

Funcionamiento Protocolo Comunicación Del I2C

La comunicación en el bus I2C se inicia cuando el maestro pone la condición START (S) en el bus, es definida como una transición HIGH-a-BAJA de la línea SDA mientras que la línea SCL es HIGH (véase la figura siguiente). El bus considera ocupado hasta que el maestro pone una condición STOP (P) en el bus, que se define como LOW a HIGH en la línea SDA mientras que SCL está en ALTA (vea la figura abajo). Además, el bus permanece ocupado si se genera un START (Sr) repetido en lugar de una condición STOP.

Un paquete I2C típico tiene un primer byte que comienza con el procesador maestro (MM) enviando una señal de condición de inicio que toma la línea SDA baja, mientras que la línea SCL es alta. Después de eso, el MM tira repetidamente de la línea SCL y luego la libera, permitiendo que la resistencia de pull up hasta 3.3 V; esto envía una baja... alta ... baja ... alta ... señal de reloj. Cada vez que SCL es alta, el MM transmite el siguiente valor de datos binarios fijando la línea SDA baja para enviar un 0 binario, o fijando arriba por medio de la resistencia para enviar un 1 binario.

Funcionamiento de Sensor MP6050 (Sensor Acelerómetro, Giroscopio y de Temperatura)

Los sensores acelerómetros y giroscopios tienen aplicación en la industria automotriz en los sistemas de seguridad como indica Manresa Pedreño en [3] y en la robótica para determinar la ubicación y diseñar el control de desplazamiento de un robot como indica Zhang en [4]. El sensor MP6050 [5], se compone de los siguientes bloques:
Sensor giroscopio de tres ejes con un CAD de 16 bits de acondicionamiento de señal, sensor acelerómetro de tres ejes con un CAD de 16 bits de acondicionamiento de señal, sensor de temperatura, motor procesador de movimiento digital (DMP), interfaz de comunicaciones serie I2C y SPI, además registros de datos de sensores, memoria FIFO e interrupciones.

El Giroscopio MEMS de Tres Ejes con CAD De 16 Bits en Acondicionamiento de Señal
El MP6050 consta de tres giroscopios de velocidad MEMS vibratorios independientes, que detectan la rotación alrededor los ejes X, Y y Z. Cuando los giroscopios giran alrededor de cualquiera de los ejes sensoriales, el efecto Coriolis causa una vibración que es detectada por un capacitor seleccionado. La señal resultante es amplificada, desmodulada y filtrada para

producir una tensión que es proporcional a la velocidad angular. Este voltaje se digitaliza utilizando un chip individual CAD de 16 bits para muestrear cada eje. El rango completo de los sensores giroscópicos se pueden programar digitalmente a ± 250, ± 500, ± 1000 o ± 2000 grados por segundo (dps). La muestra del CAD es programable desde 8000 muestras por segundo, hasta 3.9 muestras por segundo, y los filtros pasa bajo seleccionable por el usuario que permiten una amplia gama de frecuencias de corte.

Acelerómetro MEMS de Tres Ejes con CAD de 16 Bits en Acondicionamiento de Señal
El acelerómetro de 3 ejes de la MP6050 utiliza masas de prueba separadas para cada eje. La aceleración en un eje particular induce desplazamiento sobre la masa de prueba correspondiente, y los sensores capacitivos detectan el desplazamiento diferencial. La arquitectura de la MP60X0 reduce la susceptibilidad de los acelerómetros así como a la deriva térmica. Cuando el dispositivo se coloca sobre una superficie plana, 0g en los ejes X e Y y + 1g en el eje Z. El factor de escala de los acelerómetros se calibra en fábrica y es nominalmente independiente de la tensión de alimentación. Cada sensor tiene un CAD sigma-delta dedicado para proporcionar salidas digitales. El rango completo de escala de la salida digital se puede ajustar a ± 2g, ± 4g, ± 8g o ± 16g.

Desarrollo

Interfaz Vía I2C del Microprocesador P8X32A [6] **al Sensor / MP6050**

Una vez descrito el funcionamiento y características técnicas del procesador además del sensor, se inicia el desarrollo de la interfaz de comunicación.

El hardware que muestra la interfaz de comunicación I2C se describe en la Fig.2.

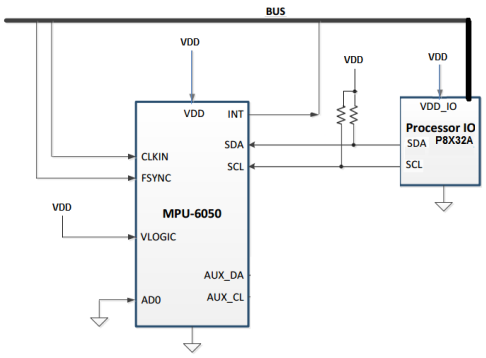

Figura 2. Interfaz entre el procesador y sensor MP6050 para protocolo I2C.

Los registros internos y la memoria de MPU6050 se pueden acceder usando I2C a 400 kHz o SPI a 1MHz (MP6000 solamente). SPI opera en modo de cuatro hilos e I2C con dos.

Debido a MP6050 siempre funciona como un dispositivo esclavo cuando se comunica con el procesador del sistema, el cual actúa como maestro. Las líneas SDA y SCL necesitan típicamente resistencias pull-up a VDD. La velocidad máxima del bus es 400 kHz.

La dirección del esclavo del MP6050 es b110100X tiene 7 bits de longitud. El bit LSB de la dirección de 7 bits es determinado por el nivel lógico en el pin AD0. Esto permite conectar dos MP6050 al mismo bus I2C como muestra tabla 1.

Tabla1. Registro de Identificación MP6050.

Registro 117, WHO_AM_I[6:1]?									
Registro HEX	**Registro DEC**	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
75	117	Valor por default =68H=1101000b							-

Cuando se utiliza en esta configuración, la dirección de uno de los dispositivos debe ser b1101000 y la dirección del otro debe ser b1101001.

Los primeros siete bits enviados son la dirección I2C del chip de destino, transmitida primero el bit más significativo. La dirección del sensor P6050 es 0b1101000, enviará primero el 1, luego el 1, otro 0, un1 y tres ceros. El último bit de datos en el primer byte se denomina bit de lectura / escritura:
Si este bit leído es 1, le dice al chip I2C que responda con datos, será la operación de lectura. Si es 0, significa que el chip I2C está a punto de recibir datos, será la operación de escritura de acuerdo a Fig.3.

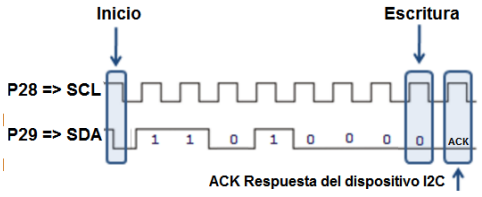

Figura 3. Operación de escritura.

De acuerdo a la figura, el bit de lectura / escritura se pone a 0, por lo que el MM va a enviar más información al sensor MP6050. Cuando el MM envía el 9º pulso CLK, libera el control de la línea SDA de modo que el chip I2C puede responder con un valor bajo para confirmar (ACK) que está listo para continuar o un valor alto para no reconocer (NACK).

En este punto, la función i2c_in ha establecido el puntero de dirección de memoria del dispositivo. Ahora, es hora de leer los bytes de datos del MP6050. Esto es iniciado por otro byte con la condición de inicio, dirección I2C y bit de lectura/escritura y ACK del dispositivo I2C. Esta

vez, el bit de lectura/escritura se establece en 1 para una operación de lectura se muestra Fig.4, significa que el sensor responderá enviando datos.

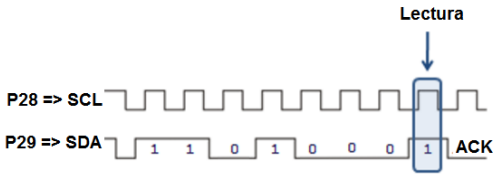

Figura 4. Operación de lectura.

Para una operación de lectura, el MM continúa suministrando impulsos SCL, pero ahora el sensor controla la línea SDA con 8 pulsos cuando responde con datos y el procesador tiene que responder con un ACK o NACK en SDA cuando se presenta el impulso SCL 9.

Desarrollo

Circuito eléctrico

De las hojas de datos de los fabricantes se consultan y se realiza el conexionado de los dispositivos como muestra la Fig.5.

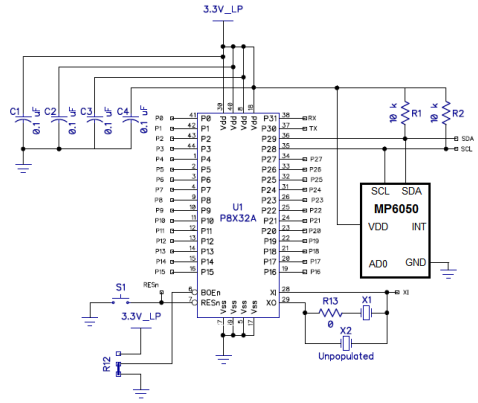

Figura 5. Diagrama eléctrico.

el Software en C++

Partiendo del hardware se diseña el algoritmo en lenguaje C++

// DECLARACION DE ESTRUCTURAS CLAVES DEL PROGRAMA

*i2c *eeBus2;* *//Identificador del bus I2C*

```
const int eeSCL=28; //Se asigna terminal 28 del uP para el SCL
const int eeSDA=29;              //Se asigna terminal 29 del uP para el SDA

//MPU-6050 guarda los datos de los sensores en dos bytes o 16 bits
//Asignación de variables Ax= Acelerómetro en x y Gy=Giroscopio en y,...

//CONFIGURACION DEL PROTOCOLO i2C
//Escritura al MP6050
i2c_out(eeBus2, 0b1101000, memAddr2, 1, (char*) &val2, 1); //Escritura al MP6050

//Formato de escritura de la función
int  i2c_out(i2c *busID, int i2cAddr,
             int memAddr, int memAddrCount,
             const unsigned char *data, int dataCount);

// i2c_out : función de escritura en el MP6050
// i2c_in : función de lectura del MP6050

// i2c *busID: Apuntador identificador de Bus i2C.
// i2cAddr : 0b1101000. 7 bits que indican Dirección del dispositivo MP6050
// memAddr: Valor de la dirección asignada al registro dentro del MP6050
// memAddrCount : Numero de bytes que usa para direccionar memAddr. Este valor puede
// ser cero para el caso que el dispositivo habilitado no tenga memoria interna. En este caso uno
// *data : (char*) &val2: Apuntador de datos para accesar al dispositivo vía I2C en este caso es un caracter
// dataCount : Numero de bytes de datos para accesar en este caso uno.
```

Las siguientes tablas muestran los registros del acelerómetro y giroscopio en el MP6050.

Tabla A. Registros del acelerómetro

Registro HEX	Registro DEC	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
3B	59	SAL ACCEL_X[15:8]							
3C	60	SAL ACCEL_X[7:0]							
3D	61	SAL ACCEL_Y[15:8]							
3E	62	SAL ACCEL_Y[7:0]							
3F	63	SAL ACCEL_Z[15:8]							
40	64	SAL ACCEL_Z[7:0]							

```
print("\nLectura de registros del acelerometro \n");
//Lectura de datos del acelerómetro en XYZ
i2c_in(eeBus2,0b1101000,60, 1, (char*)&val2,1);
pause(500);
AcX=val2;                                   //Dato recuperado del MP6050
print(" AcX =%f, ", AcX);                   //Imprime en pantalla

i2c_in(eeBus2,0b1101000,62, 1, (char*) &val2,1);
pause(500);
AcY=val2;                                   //Dato recuperado del MP6050
print(" AcY=%f, ", AcY);                    //Imprime en pantalla

i2c_in(eeBus2,0b1101000,64, 1, (char*)&val2,1);
pause(500);
AcZ=val2;                                   //Dato recuperado del MP6050
print(" AcZ=%f,\n", AcZ);                   //Imprime en pantalla
```

Tabla B. Registros del giroscopio.									
Registro HEX	**Registro DEC**	**bit 7**	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
43	67	SAL GYRO_X[15:8]							
44	68	SAL GYRO_X[7:0]							
45	69	SAL GYRO_X[15:8]							
46	70	SAL GYRO_X[7:0]							
47	71	SAL GYRO_X[15:8]							
48	72	SAL GYRO_X[7:0]							

```
print("\nLectura de registros del giroscopio \n");
//Lectura de datos del giroscopio en XYZ

i2c_in(eeBus2,0b1101000,68, 1, (char*) &val2,1);
pause(500);
GyX=val2;                                   //Dato recuperado del MP6050
print(" GcX=%f, ", GyX);                    //Imprime en pantalla

i2c_in(eeBus2,0b1101000,70, 1, (char*) &val2,1);
pause(500);
GyY=val2;                                   //Dato recuperado del MP6050
print(" GcY=%f, ", GyY);                    //Imprime en pantalla

i2c_in(eeBus2,0b1101000,72, 1, (char*) &val2,1); //change
pause(500);
GyZ=val2;                                   //Dato recuperado del MP6050
print(" GcZ=%f, ", GyZ);                    //Imprime en pantalla
```

Se muestra los resultados del programa en ejecución en la Fig.6.

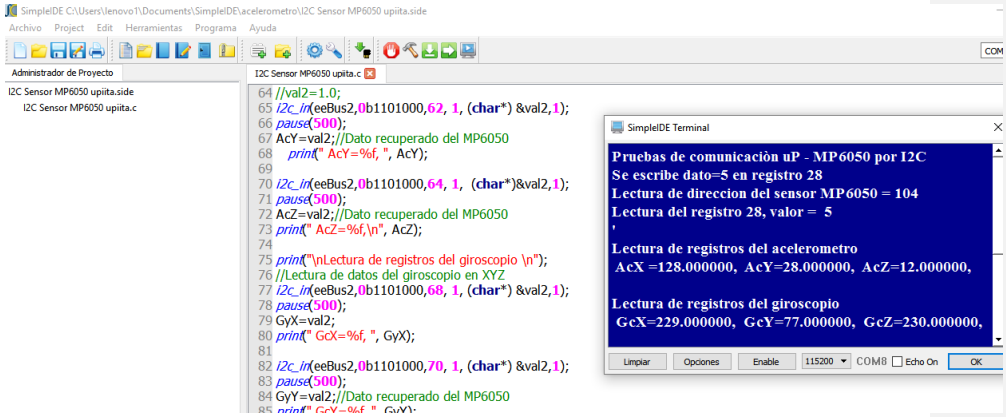

Figura 6. Resultados de programa en ejecución.

```
/*
 MP6050 comunicación  I2C lenguaje C++
 Realiza prueba de escritura en el sensor MP6050 por medio de I2C
  */
#include "simpletools.h"                      // Include simpletools header
#define A_R 16384.0  // aceleracion
#define G_R 131.0    // giroscopo
#define RAD_A_DEG 57.295779

I2c *eeBus2;              //Se habilita el bus I2C
const int eeSCL=28;  //Se asigna terminal 28 del uP para el SCL
const int eeSDA=29; //Se asigna terminal 29 del uP para el SDA
//int eeAddr=0b1010000;

//MPU-6050 guarda los datos de los sensores en dos bytes 0 16 bits
//Asignación de variables A= Acelerómetro y G=Giroscopio
float AcX, AcY, AcZ, AcX_H, AcY_H, AcZ_H, GyX, GyY, GyZ,Tp;

float Acc[2];
float Gy[2];
float Angle[2];
int val2;

char eeAddr1=0b1101000;
int i=0;
int memAddr2;
int main()                                    // Función Main
{
 eeBus2 = i2c_newbus(eeSCL, eeSDA,0);         // SeConfiguraciónbus  I2C bus

// Se realiza prueba de escritura de dato=0x64 en dirección 28 (0001 1100) del MP6050

 while (1)
 {

val2=0x64;
 memAddr2=28;
i2c_out(eeBus2, 0b1101000, memAddr2, 1,  (char*) &val2, 1); //change
pause(1000);
print("Val2(%f) = %f \n",i, val2);
 }
 }
```

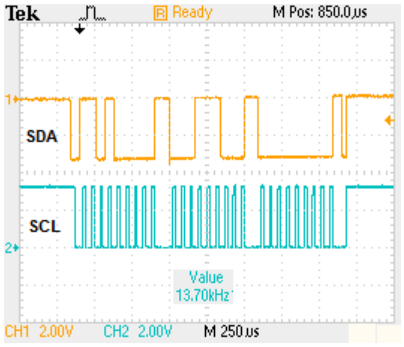

Figura 7. Programa cíclico para la escritura al MP6050 y medición de señales del bus I2C.

```
/*
 MP6050 comunicación  I2C lenguaje C++
 Realiza prueba de lectura en el sensor MP6050 por medio de I2C
  */
#include "simpletools.h"                      // Include simpletools header
#define A_R 16384.0  // aceleracion
#define G_R 131.0    // giroscopo
#define RAD_A_DEG 57.295779

I2c *eeBus2;              //Se habilita el bus I2C
const int eeSCL=28;  //Se asigna terminal 28 del uP para el SCL
const int eeSDA=29; //Se asigna terminal 29 del uP para el SDA
//int eeAddr=0b1010000;

//MPU-6050 guarda los datos de los sensores en dos bytes 0 16 bits
//Asignación de variables A= Acelerómetro y G=Giroscopio
float AcX, AcY, AcZ, AcX_H, AcY_H, AcZ_H, GyX, GyY, GyZ,Tp;

float Acc[2];
float Gy[2];
float Angle[2];
int val2;

char eeAddr1=0b1101000;
int i=0;
int memAddr2;
int main()                                    // Función Main
{
 eeBus2 = i2c_newbus(eeSCL, eeSDA,0);         // SeConfiguraciónbus  I2C bus
 while (1)
 {
  print("\nLectura de registros del acelerometro \n");
  //Lectura de datos del acelerómetro en X
  i2c_in(eeBus2,0b1101000,60, 1, (char*)&val2,1);
  pause(500);
  AcX=val2; //Dato recuperado del MP6050
  print(" AcX =%f, ", AcX);
 }
}
```

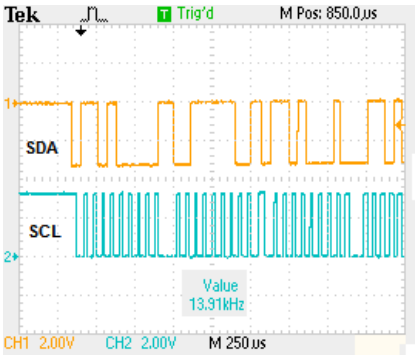

Figura 8. Programa cíclico para la lectura al MP6050 y medición de señales del bus I2C.

Conclusiones

El protocolo de comunicación I2C simplifica la comunicación del procesador con los dispositivos de entrada salida como se observa con la interfaz al sensor MP6050 debido a su direccionamiento sencillo permite leer/escribir los datos del acelerómetro y giroscopio por medio del microprocesador P8X32A por medio de dos cable lo cual simplifica el hardware del sistema embebido y se adapta perfectamente a las necesidades del sistema, el acceso a los datos es muy confiable por tener mecanismo de confirmación de datos en el protocolo de comunicación a frecuencias máximas de 400 KHz, en el proyecto el acceso de datos se realiza de manera ocasional y a una frecuencia de reloj de 14KHz.

La información que proporciona el fabricante sobre el sensor MP6050 es suficiente para interfazarlo con el procesador P8X32A el cual es un sistema abierto, ambas aperturas contribuyen en el diseño de la interfaz y darle aplicación en la robótica.

Agradecimientos

Los autores agradecen al editor ya los revisores por sus valiosos comentarios y sugerencias que nos permiten mejorar significativamente esta investigación. Los autores agradecen al Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC), al Instituto Tecnológico de Tlalnepantla del TecNM y al CONACyT por la beca en esta investigación.

Proyecto de Investigación en Curso
Proyecto SIP 20171021

Referencias

[1] Leens, Federic, "*An introduction to I 2 C and SPI protocols*" IEEE Instrumentation & Measurement Magazine, vol. 12, no. 1, pp. 8-13, 2009.
[2] Muhammad Mazidi, Sarmad Naimi and Sepehr Naimi, "*AVR microcontroller and embedded systems: using assembly and C*", Prentice Hall Press, 2010.
[3] J. M. Pedreño, "*Hécate: aplicación Android para notificación telefónica*", PFC/TFG-Escuela Técnica Superior de Ingeniería de Telecomunicación, Universidad Politécnica de Cartagena, 2013.
[4] Zhang Yingkun, Hao Cunming y Zhen Zhuo, "*The Design of Desktop Two Rounds of Self-balancing Robot*", International Conference on Computer Science and Electronic Technology, Hebei Shijiazhuang 050000, China, 2014.
[5] TDK InvenSense, www.invensense.com.
[6] Parallax Inc. www.parallax.com