Instituto Tecnológico Superior De Atlixco.

Ingeniería Mecatrónica.

IM140744.

7-A.

Microcontroladores.

PRÁCTICA 3 Control de Posición de servomotores con la IMU MPU6050.

CATEDRÁTICO: Dra. Mariana Natalia Ibarra Bonilla.

INTEGRANTE: José Ángel Balbuena Palma. IM140744

Fecha de realización:

25/11/2017

Fecha de entrega:

27/11/2017

Objetivo general.

El objetivo de esta práctica es el control de tres servomotores con los que se representa la posición angular en cada eje X, Y, Z referente a la IMU MPU6050 con el que cuenta la tarjeta de desarrollo Ophyra que posee un microcontrolador STM32F407VGTx.

Materiales.

- Protoboard.
- Tarjeta de desarrollo Ophyra.
- Servomotor.
- Jumpers.
- Computadora con el software Atollic TrueStudio, STM32CubeMX y STMFlashLoader .

MARCO TEÓRICO.

Comunicación UART.

Transmisor-Receptor Asíncrono Universal, se utiliza para poder trasmitir información de forma serial. Es una comunicación asíncrona no utiliza señal de reloj la trasmisión es modo full-dúplex.

Asíncrono (full-dúplex).

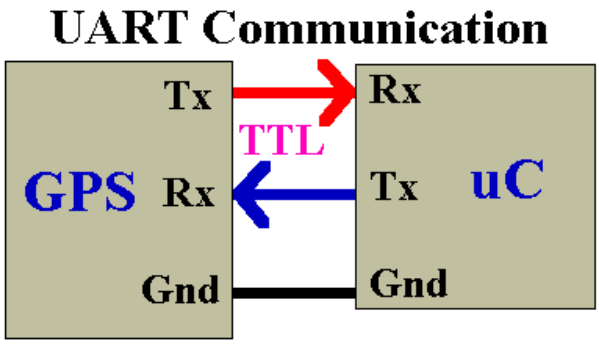
No envía señal de reloj pero emisor y receptor deben relojes a la misma frecuencia y fase se suele utilizar en distancia s largas utiliza dos líneas de comunicación.

Protocolos:

- RS-232.
- USB.

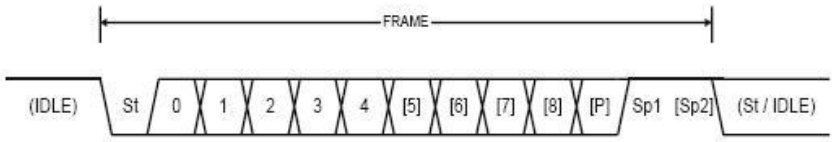
Características:

- Cada trama de datos tiene un tamaño fijo y posee un bit de inicio o arranque (start) y un bit de paro (stop) para realzar la sincronización.
- La trasmisión es de modo Full-Dúplex, que utiliza dos líneas de trasmisión Tx y otra receptora Rx, con datos en ambos sentidos.



El dato se trasmite de la siguiente forma:

- En estado inactivo la línea de datos permanece en estado alto.
- Cada trasmisión de datos inicia con un bit de arranque (START) siempre permanece en cero.
- Cada dato tiene un ancho de 8 a 9 bits desde el menos significativo LSB.
- Cada transmisión de datos termina con un bit (STOP) que siempre es un uno.



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

Comunicación USART (Un receptor, transmisor sincrónico, asíncrono universal)

Síncrono (Hall-Dúplex) Utiliza una señal de reloj y una línea de datos se utiliza cuando la distancia entre emisor y receptor es pequeña.

Protocolos:

- I2C
- SPI
- I2S

La trasmisión síncrona (USART) permite la transmisión continua de datos y no existe un límite de tamaño. No puede transferir información de forma simultánea.

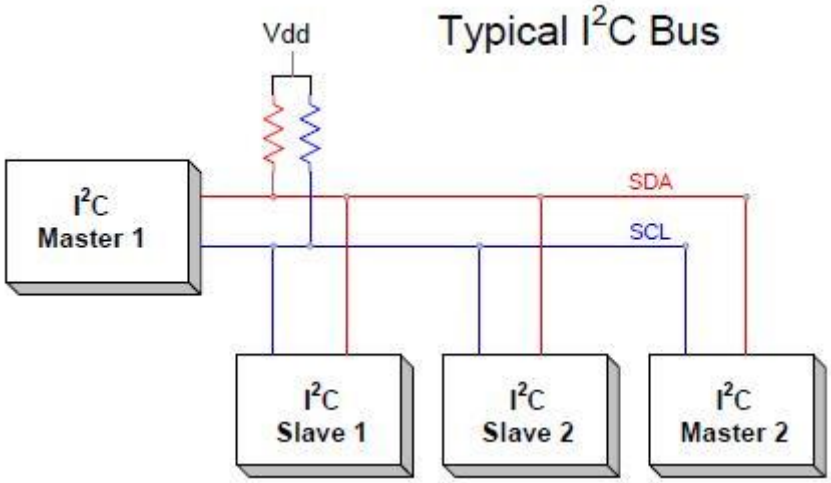
Características:

Un dispositivo actúa como maestro el cual genera una señal de reloj que inicia y finaliza la transmisión. Un dispositivo actúa como esclavo recibe la señal de reloj y depende del maestro para recibir y transferir información.

Interfaz inter-circuitos I2C.

El bus I2C es una comunicación de 2 líneas creado por Philips. Cada bus tiene una dirección .Puede configurarse como maestro-esclavo y puede haber más de un esclavo.

En ambas configuraciones el dispositivo maestro es que decide sobre la transferencia, decide quien la realiza, el sentido de la misma y cuando finaliza.



Se deben utilizar resistencias de colector abierto como resistencias externas de Pull-Up. Puede haber una velocidad de transferencia de hasta 400 Kbps.

Formato de transmisión de datos.

El maestro envía la condición de inicio (S) para iniciar la comunicación.

El maestro envía una trama de bits los cuales contienen la dirección del dispositivo con el que establecerá la comunicación.

El maestro envía un bit para decir que tipo de comunicación establecerá de lectura o de escritura, R/W un cero para lectura y un uno para escritura.

El esclavo con la dirección correspondiente envía un bit de confirmación (ACK).

EL maestro o el esclavo envían un byte de información. En caso del que el bit de confirmación termina la comunicación.

El maestro finaliza la comunicación con un bit de paro (stop).

Protocolo I2C:

LA IMU MPU6050.

Las unidades de medición inercial están compuestas por un conjunto de sensores que son capases de medir la aceleración y la velocidad, con un acelerómetro y un giroscopio de tres ejes. Por lo tanto, los valores que nos puede entregar el MPU6050 son las aceleraciones producidas por la gravedad en cada uno de nuestros ejes dadas en g (9.8 m/s^2) por parte de acelerómetro y el por parte del giroscopio nos entrega una velocidad angular presente en cada uno de los ejes en grados (°)/s . Esta IMU también nos puede entregar mediciones de temperatura pues pose integrado un sensor de temperatura.

Los valores que nos entrega el MPU6050 se almacenan en 14 registros.

Sensor	Registro en Hexadecimal	Dato Almacenado
Acelerómetro	3B	ACCEL_XOUT [15:8]
	3C	ACCEL_XOUT [7:0]
	3D	ACCEL_YOUT [15:8]
	3E	ACCEL_YOUT [7:0]
	3F	ACCEL_ZOUT [15:8]
	40	ACCEL_ZOUT [7:0]
Temperatura	41	TEMP_OUT [15:8]
	42	TEMP_OUT [7:0]
Giroscopio.	43	GYRO_XOUT [15:8]
	44	GYRO_XOUT [7:0]

	45	GYRO_YOUT [15:8]
	46	GYRO_YOUT [7:0]
	47	GYRO_ZOUT [15:8]
	48	GYRO_ZOUT [7:0]

Rangos de escalas, sensibilidad del acelerómetro y giroscopio.

Sensor	Rangos de escalas	Sensibilidad
Acelerómetro	$\pm 250°/s,\ \pm 300°/s\ ,$ $\pm 1000°/s\ y\ \pm 2000°/$ $s.$	1638 LSB/g, 8192 LSB/g, 4096 LSB/g, 2048 LSB/g.
Giroscopio	$\pm 2\ g, \pm 4g, \pm 8g\ y\ \pm$ $16g.$	131 LSB/dps, 65.5 LSB/dps, 32.8 LSB/dps, 16.4 LSB/dps.

Para obtener la medición de la temperatura en grados centígrados se aplica la siguiente formula.

Temperatura en °C = (Valor de registro) /340+36.53

Power Manager

Este registro tiene la dirección de 0x6B permite al usuario configurar el modo de energía y una fuente de reloj, Posee un bit para resetear todo el dispositivo(DEVICE_RESET), así como un bit para deshabilitar el sensor de temperatura(TEMP_DISP). En este registro podemos desactivar el bit de SLEEP en cero, el cual tiene la función de deshabilitar el sensor de temperatura.

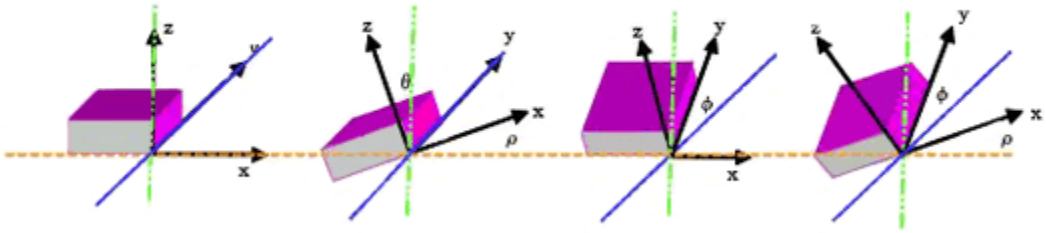
Frecuencia de muestreo

Esta frecuencia es la frecuencia a la que obtenemos los datos de este módulo, se encuentra en el registro 0x19, para trabajar a una frecuencia de muestreo de 1kKz se escribe el dato 0x07 en el registro.

La comunicación que el microcontrolador STM32F407VGx establece con el MPU6050 es por medio del protocolo I2C donde el maestro es el microcontrolador y el esclavo es modulo MPU6050 la dirección del dispositivo es 0x68 en lectura y 0x69 en escritura.

Determinación del ángulo de inclinación con el uso del acelerómetro.

EL acelerómetro nos puede proporcionar las medidas dadas en aceleraciones respecto a cada uno de los ejes, pero no en posiciones con las cuales se pueden determinar por medio de la geometría respecto a los vectores de sus aceleraciones.



Angulo en X.

$$AnguloX_{acelerometro} = tan^{-1}(\frac{Ax}{\sqrt{Ay^2+Az^2}})$$

Angulo en Y.

$$AnguloY_{acelerometro} = tan^{-1}(\frac{Ay}{\sqrt{Ax^2+Az^2}})$$

Determinación del ángulo de inclinación con el uso del giroscopio.

El giroscopio nos da la velocidad angular que esta presente en cada uno de los ejes al realizar algún movimiento. La ecuación para obtener el Angulo con la velocidad es la siguiente:

$$Angulo_{Giroscopio} = Angulo_{Inicial} + VelocidadAngular * \Delta t$$

Donde Δt es el tiempo transcurrido cada ves que se aplica la ecuación.

Para cada uno de los ejes:

$$AnguloX_{Giroscopio} = AnguloX_{Inicial} + VelocidadAngularX * \Delta t$$

$$AnguloY_{Giroscopio} = AnguloY_{Inicial} + VelocidadAngularY * \Delta t$$

$$AnguloZ_{Giroscopio} = AnguloZ_{Inicial} + VelocidadAngularZ * \Delta t$$

La aplicación de un filtro es necesaria poder lograr una reducción de errores en las mediciones del MPU6050 se puede aplicar algo que se conoce como filtro

complementario. Este filtro es resultado de complementar los dos ángulos obtenidos en las mediciones de giroscopio y el acelerómetro como se muestra en la siguiente ecuación.

$$Angulo = 0.9 * Angulo_{Giroscopio} + 0.1 * Angulo_{acelerometro}$$

Configuración en STM32CubeMX para la comunicación UART(RS-232).

Para poder utilizar la comunicación UART (protocolo RS-232) con nuestra tarjeta de desarrollo Ophyra se necesita activar el USAR3 que se encuentra disponible para su uso en forma serial, de manera Asíncrona.

Los parámetros que se le deben configurar es la velocidad de transición a 115200 bits/s cuya velocidad será la de sincronización.

Funciones para recibir y trasmitir RS232 en Atollic TrueStudio.

HAL_UART_Receive ();

Es una función recibimos los datos provenientes de la comunicación serial (RS-232), requiere como parámetros: el puntero al UART a utilizar, una variable de tipo uint8_t para almacenar el dato a recibir, la cantidad de datos a recibir de tipo uint16_t y el tiempo de espera del puerto en ms de tipo uint32_t.

HAL_UART_Transmit ();

Es una función enviamos datos desde el micro controlador por comunicación serial (RS-232), requiere como parámetros: el puntero al UART a utilizar, una variable o un apuntador de tipo uint8_t que contiene los datos a trasmitir, la cantidad de datos a enviar de tipo uint16_t se puede recurrir a la función strlen() que devuelve la cantidad de datos contenidos en un buffer, el tiempo de espera en ms de strlen.

Configuración en STM32CubeMX para la comunicación USART(I2C).

Para poder utilizar la comunicación USART (protocolo I2C) con nuestra tarjeta de desarrollo Ophyra se necesita activar //añadir

Funciones para recibir y trasmitir con el protocolo I2C en Atollic TrueStudio.

HAL_I2C_Master_Trasmit ();

Es una función que permite realiza la comunicación maestro esclavo, de manera que el maestro envía información al esclavo. Los parámetros que recibe esta función son el puntero al I2C que emplearemos en la comunicación, la dirección del dispositivo a comunicarse, el puntero al dato que vamos a trasmitir de tipo uint8_t, el tamaño del dato a trasmitir de tipo uint16_t y tiempo de espera del puerto en ms de tipo uint32_t.

HAL_I2C_Master_Receive ();

Esta función nos permite recibir los datos provenientes del dispositivo esclavo, Los parámetros que recibe esta función son el puntero al I2C que emplearemos en la comunicación, la dirección del dispositivo a comunicarse, el puntero al dato que emplearemos para recibir los datos provenientes del esclavo de tipo uint8_t, tamaño de datos a recibir de tipo uint16_t y tiempo de espera del puerto en ms de tipo uint32_t.

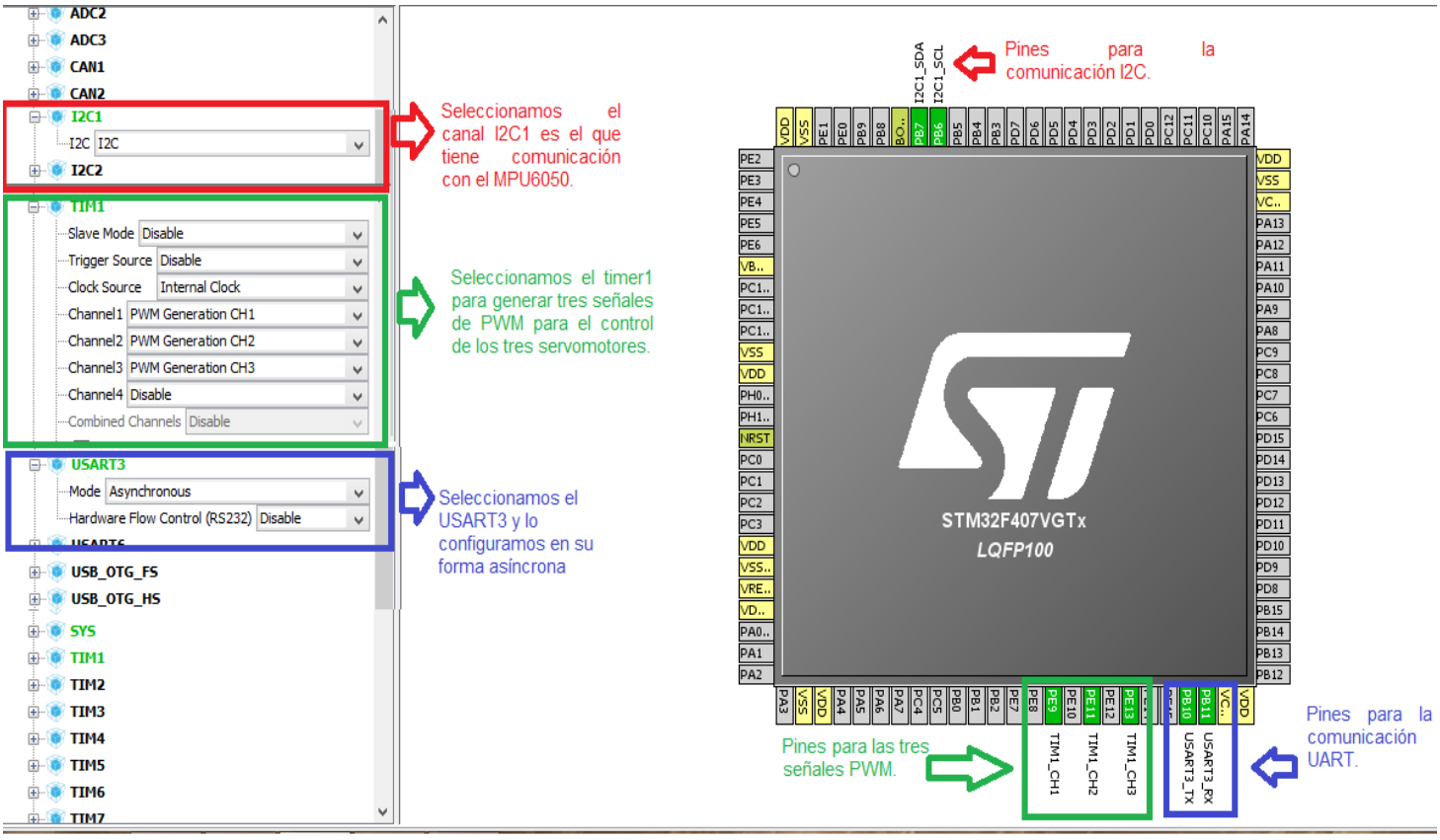
Estas sentencias automáticamente adicionan y cambian el bit AD0, realizan un corrimiento a la izquierda.

	Dirección del dispositivo MPU6050	Corrimiento al a Izquierda	Dirección correcta para la instrucción
Escritura	1101000	110100AD0	0xD0
Lectura	1101001	110100AD0	0xD0

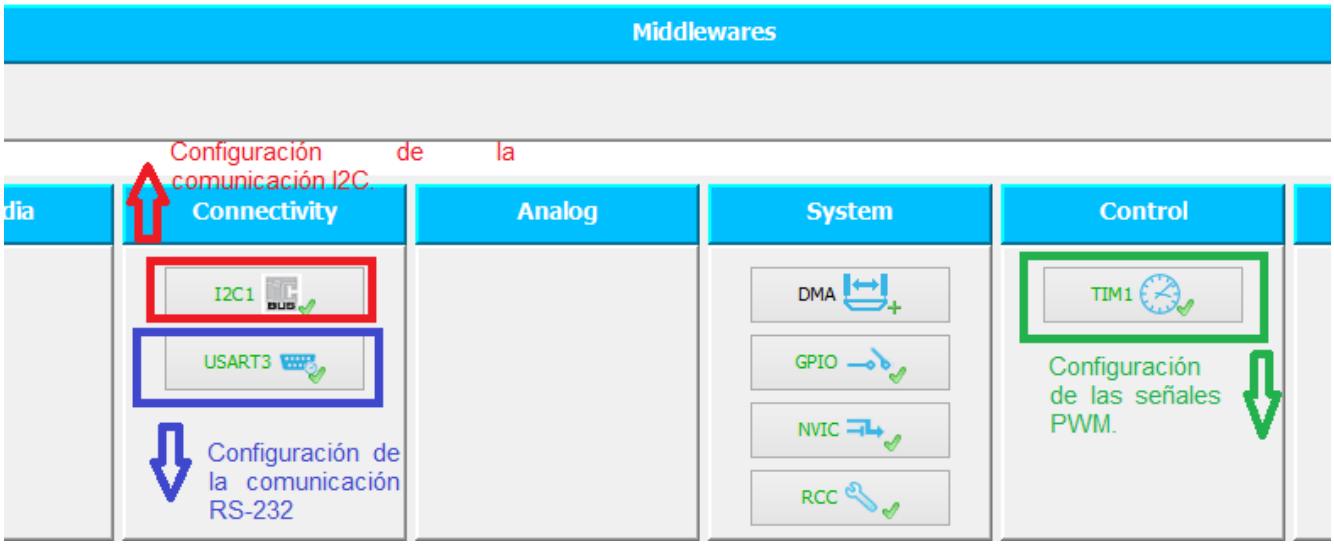
Desarrollo.

Configuración del proyecto en el software STM32CubeMx.

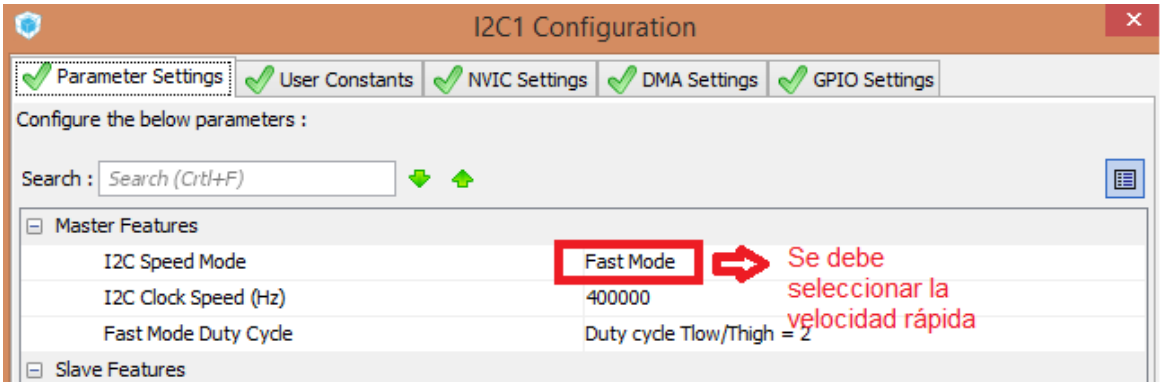
Selección de componentes a emplear.



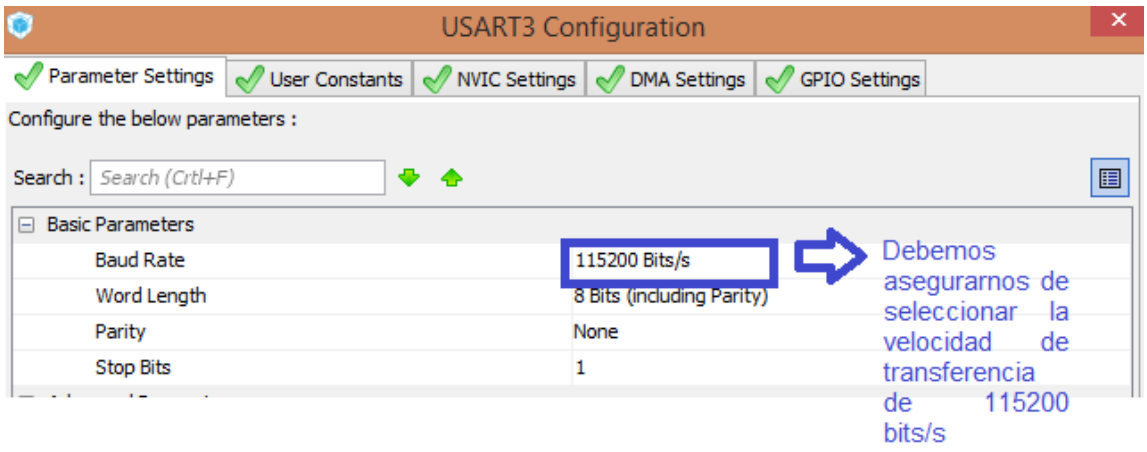
Configuración de los elementos seleccionados.



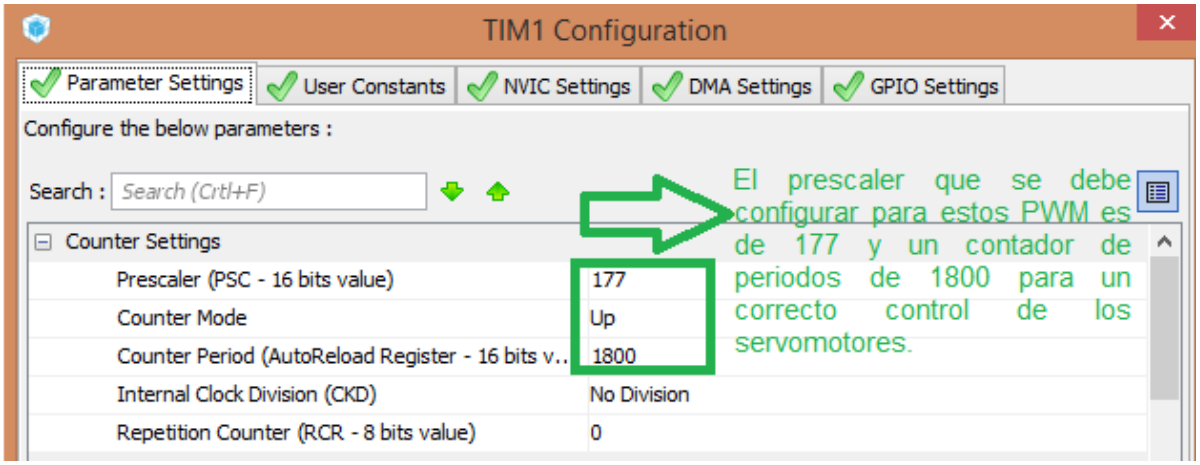
En la comunicación I2C solo se debe configurar el parámetro.



El parámetro que debemos cambiar para la comunicación UART es el siguiente.



Configuración del Timer1.



Realización del programa en Atollic TrueStudio.

```
/* USER CODE BEGIN Inclu
#include"stdio.h"
#include"string.h"
#include"math.h"
#define MPU6050_AD 0xd0
#define RAD 180.0/3.14//
```

Inclusión de las librerías a utilizar la librería estándar de entra y salidas de C, la librería para el manejo de string y la librería para operaciones matemáticas, se define MPU6050_AD como la dirección del dispositivo 0xD0 y el factor RAD para pasar de radianes a grados.

Declaración de variables a emplear en la función principal.

```
int main(void)
{

uint8_t DatoRecibido[15];
uint8_t DatoEnviado[2];
char bufer[100];
float AX,AY,AZ,GX,GY,GZ,T;//VARIABLES
float  Ang[3]={0,0,0};
float AcANG[2]={0,0};
float angulos[3]={0,0,0};
```

Declaración del arreglo de tipo uint8_t que almacenara los datos recibidos del MPU6050.

Declaración del arreglo de tipo uint8_t que almacenara los datos enviados al MPU6050.

Declaración del bufer que concatenara los datos amostrar por la terminar serial

Declaración de las variables tipo flotante que contendrán los valores de la aceleración en X,Y y Z, la velocidad angular en X,Y y Z también una para la temperatura.

Este arreglo contendrá las posiciones angulares para cada uno de los ejes.

Este arreglo almacena las posiciones angulares de X y Y obtenidas a partir del acelerómetro.

Este arreglo contendrá los valores que serán enviados a los servomotores para accionarlos.

Inicio de la comunicación I2C para la configuración del Registro Power Manage

Almacenamos la dirección del registro del Power Manager en la primera posición el array del Dato enviado

```
93   DatoEnviado[0]=0x6B;
```

Almacenamos el valor de 0x00 en la segunda posición el array DatoEnviado, para deshabilitar la opción sleep del sensor de temperatura.

```
95   DatoEnviado[1]=0x00;
```

```
97   HAL_I2C_Master_Transmit(&hi2c1,(uint16_t)MPU6050_AD,DatoEnviado,2,100);
```

Con esta función podemos realizar comunicación I2C con el MPU6050 trasmitiéndole el arreglo dato enviado para realizar las configuraciones del Power Manager.

```
101  DatoEnviado[0]=0x19;
```

Almacenamos la dirección del registro del Sample Rate Divide en la primera posición del array DatoEnviado.

```
102  DatoEnviado[1]=0x07;
```

Almacenamos el valor 0x07 en la segunda posición del array DatoEnviado, para configurar una frecuencia de muestreo de 1KHz .

```
104  HAL_I2C_Master_Transmit(&hi2c1,(uint16_t)MPU6050_AD,DatoEnviado,2,100);
```

Con esta función se puede enviar al MPU6050, el arreglo de DatosEnviar con el cual se configura el registro de la frecuencia de muestreo.

```
109  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_1);
110  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);
111  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_3);
112
113
114   while (1)
115   {
```

Inicializamos los tres canales de PWM generados por Timer.

Almacenamos la dirección del registro del acelerómetro Ax más significativo y lo guardamos en la primer posición del array.

```
DatoEnviado[0]=0x3B;
```

No almacenamos algun valor significativo en la segunda posicion del arreglo.

```
DatoEnviado[1]=0x00;
```

```
HAL_I2C_Master_Transmit(&hi2c1,(uint16_t)MPU6050_AD,&DatoEnviado[0],1,100);
```

Trasmitimos únicamente la primera posición del arreglo que contiene la dirección del acelerómetro.

```
HAL_I2C_Master_Receive(&hi2c1,(uint16_t)MPU6050_AD,DatoRecibido,14,100);
```

Esta función nos permite almacenar los datos trasmitidos por el MPU6050, en el arreglo DatoRecibido, el cual posee las lecturas en bruto provenientes de los registros para el acelerómetro, sensor de temperatura y giroscopio

```
AX=(float)((int16_t)(DatoRecibido[0]<<8|DatoRecibido[1]))/(float)1638);
AY=(float)((int16_t)(DatoRecibido[2]<<8|DatoRecibido[3]))/(float)1638);
AZ=(float)((int16_t)(DatoRecibido[4]<<8|DatoRecibido[5]))/(float)1638);
T=(float)((int16_t)(DatoRecibido[6]<<8|DatoRecibido[7]))/(float)340+(float)36.53);
GX=(float)((int16_t)(DatoRecibido[8]<<8|DatoRecibido[9]))/(float)131);
GY=(float)((int16_t)(DatoRecibido[10]<<8|DatoRecibido[11]))/(float)131);
GZ=(float)((int16_t)(DatoRecibido[12]<<8|DatoRecibido[13]))/(float)131);
```

El valor que se obtenido del MPU6050 para la aceleración en el eje x, se encuentra almacenado en dos posiciones del arreglo , la primera se tiene que desplazar 8 bits al valor obtenido el valor del acelerómetro del eje x más significativo y se concatena con el otro registro de la lectura menos del valor significativos para obtener su valor total , este valor se debe dividir por la sensibilidad del dispositivo en el caso del acelerómetro es de 1638, para el giroscopio de una sensibilidad de 131 y para el sensor de temperatura se aplica la fórmula para obtener la temperatura en grados.

```
AcANG[0]=atan(AX/sqrt(pow(AY,2) + pow(AZ,2)))*RAD;
AcANG[1]=atan(AY/sqrt(pow(AX,2) + pow(AZ,2)))*RAD;
```

Una vez que se ha obtenido las aceleraciones para cada uno de los ejes se es posible determinar el grado de inclinación para el eje X y Y.

```
Ang[0]= 0.9*(Ang[0]+GX*0.01) + 0.1*AcANG[0];
Ang[1]= 0.9*(Ang[1]+GY*0.01) + 0.1*AcANG[1];
Ang[2]= Ang[2]+GZ*0.01;
```

Para el ángulo X y Y es posible ser encontrado aplicando el filtro complementario pero para Z solo se puede calcular el ángulo en función de la velocidad del giroscopio cuya lectura podría presentar diversos errores.

```
if(Ang[0]>=0&&Ang[0]<90){
    angulos[0]=Ang[0]+45+90;
   __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,angulos[0]);
}
```

Para un ángulo en x mayor o igual que cero y menor que noventa se le debe sumar al ángulo obtenido más 45, más 90 que desplaza el ángulo cero a 90° en el servomotor este valor es cuál puede ser enviado la función __HAL_TIM_SET_COMPARE() donde se establece el Angulo.

```
else if(Ang[0]<0&&Ang[0]>-90){
    angulos[0]= Ang[0]*(-1)+45;
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,angulos[0]);
}
```

Para un ángulo en x menor que cero y mayor que noventa se le debe multiplicar por menos uno pues estos ángulos son negativos sumar al ángulo obtenido más 45 el servomotor, estos valores serán representados del valor de 90 en el servomotor hasta cero, su valor es cuál puede ser enviado con la función __HAL_TIM_SET_COMPARE() donde se establece el Angulo en el servomotor.

De igual manera se configuran los otros dos servomotores.

```
if(Ang[1]>=0&&Ang[1]<90){
        angulos[1]=Ang[1]+45+90;
       __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_2,angulos[1]);
    }
    else if(Ang[1]<0&&Ang[1]>-90){
        angulos[1]= Ang[0]*(-1)+45;
        __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_2,angulos[1]);
    }
if(Ang[2]>=0&&Ang[2]<90){
        angulos[2]=Ang[2]+45+90;
       __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_3,angulos[2]);
    }
    else if(Ang[2]<0&&Ang[0]>-90){
        angulos[2]= Ang[2]*(-1)+45;
        __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_3,angulos[2]);
    }
```

```
sprintf(bufer, "Temp=%.2f\n\r,Angulo_X=%.2f\n\r,Angulo_Y=%.2f\n\r,Angulo_Z=%.2f\n\r\n",T,Ang[0],Ang[1],Ang[2]);
```

Esta función de la librería string te permite concatenar una serie de valores y palabras en un solo bufer y poder ser enviado para mostrarse por serial, en este caso mostramos la temperatura y los ángulos en X,Y y Z.

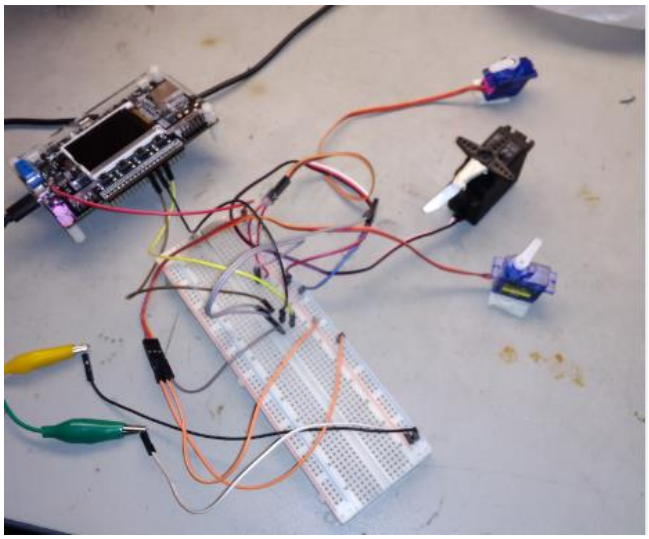
```
HAL_UART_Transmit(&huart3,(uint8_t*)bufer,(uint16_t)strlen(bufer),(uint32_t)100);
```

Esta función nos permite enviar el bufer donde se encuentran nuestros valores de los ángulos para poder ser vistos por la terminal serial.

```
    HAL_Delay(10);
  }

}
```

Aplicamos una pequeña pausa de 10ms para que vuelva a repetir el ciclo.



```
,Angulo_Y=21.37
,Angulo_Z=-27.93

Temp=22.55
,Angulo_X=37.03
,Angulo_Y=21.30
,Angulo_Z=-27.94

Temp=22.55
,Angulo_X=36.99
,Angulo_Y=21.28
,Angulo_Z=-27.96

Temp=22.55
,Angulo_X=36.90
,Angulo_Y=21.28
,Angulo_Z=-27.98
```

Conclusión.

Para poder utilizar una IMU como el MPU6050 es necesario poder conocer de diversos temas como son protocolos de comunicación , el caso del I2C que se empleó para trasmitir datos de las aceleración y velocidades angulares que nos proporciona el MPU6050 pero que dichos datos debes ser trasmitidos de esclavo a maestro , estos se encuentran almacenados en registros estos deben ser desplazados, concatenados y divididos por su sensibilidad para obtener su valor de velocidad o aceleración , el maestro puede realizar configuraciones sobre el dispositivo a partir de una configuración maestro esclavo donde trasmite la dirección del registro y el valor a modificar para cambiar algún parámetro del dispositivo, los valores que nos proporciona el giroscopio y el acelerómetro pueden ser empleados para determinar la posición angular con ayuda de la geometría, y aplicación de conceptos físicos, estos valores de los ángulos obtenidos para poder ser enviados a los servomotores de pendiendo del origen elegido para nuestros sistema de referencia en la ubicación de los servomotores, los valores de los ángulos también puede mostrarse en la ventana de la terminar serial atreves de una comunicación UART donde mandamos información en atreves de puerto serial y se visualiza en la terminal.

Referencias.

Perles Àngel; ARM Cortex-M práctico. 1 - Introducción a los microcontroladores STM32 de St, Univercidad Politécnica de Valencia, Valencia 2017,206 págs.