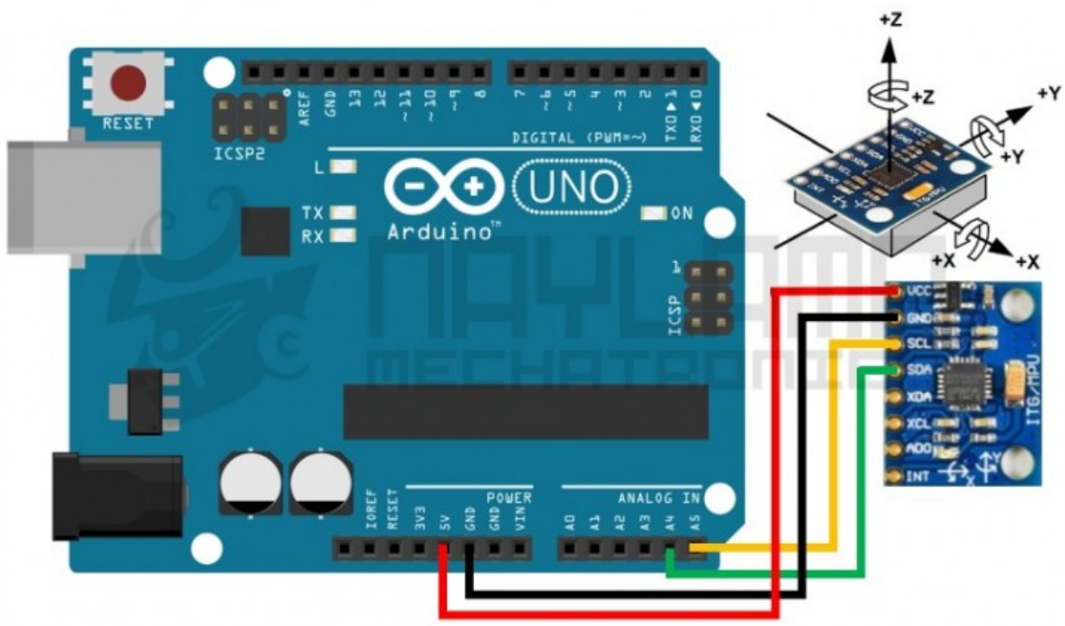
Tutorial MPU6050, Acelerómetro y Giroscopio

Fuente: ***https://naylampmechatronics.com/*** *(Versión 4-7-19)*



EL MPU6050 es una unidad de medición inercial o IMU (Inertial Measurment Units) de 6 grados de libertad (DoF) pues combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes. Este sensor es muy utilizado en navegación, goniometría (medición de ángulos), estabilización, etc.

En este tutorial revisaremos los principios generales de acelerómetros y giroscopios, luego estudiaremos el sensor MPU6050 con mayor detalle y finalmente implementaremos ejemplos del MPU6050 con Arduino.

Aceleración y acelerómetros

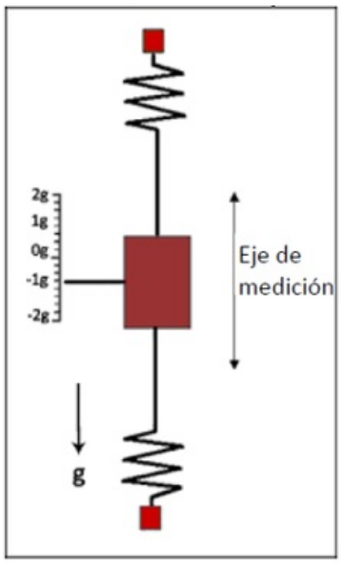
La aceleración es la variación de la velocidad por unidad de tiempo es decir razón de cambio en la velocidad respecto al tiempo:

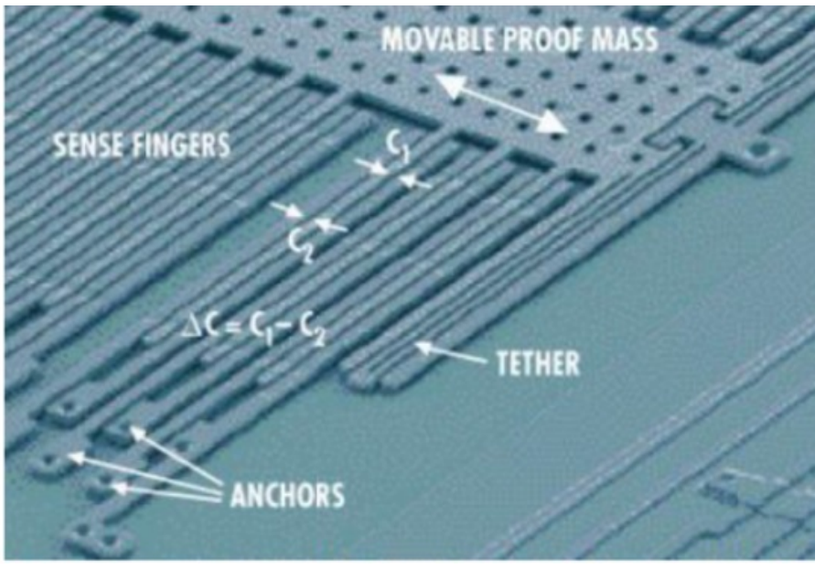
$$a=dV/dt$$

Así mismo la segunda ley de Newton indica que en un cuerpo con masa constante, la aceleración del cuerpo es proporcional a la fuerza que actúa sobre él mismo:

$$a=F/m$$

Este segundo concepto es utilizado por los acelerómetros para medir la aceleración. Los acelerómetros internamente tienen un MEMS (MicroElectroMechanical Systems) que de forma similar a un sistema masa resorte permite medir la aceleración.



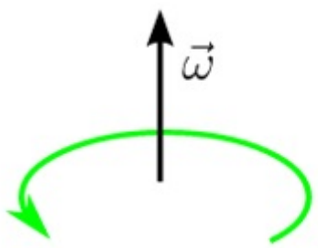


Con un acelerómetro podemos medir esta aceleración, teniendo en cuenta que a pesar que no exista movimiento, siempre el acelerómetro estará sensando la aceleración de la gravedad.

Con el acelerómetro podemos hacer mediciones indirectas como por ejemplo si integramos la aceleración en el tiempo tenemos la velocidad y si la integramos nuevamente tenemos el desplazamiento, necesitando en ambos casos la velocidad y la posición inicial respectivamente.

Velocidad Angular y giroscopio

La velocidad angular es la tasa de cambio del desplazamiento angular por unidad de tiempo, es decir que tan rápido gira un cuerpo alrededor de su eje:



$$\omega = \frac{d\theta}{dt}$$

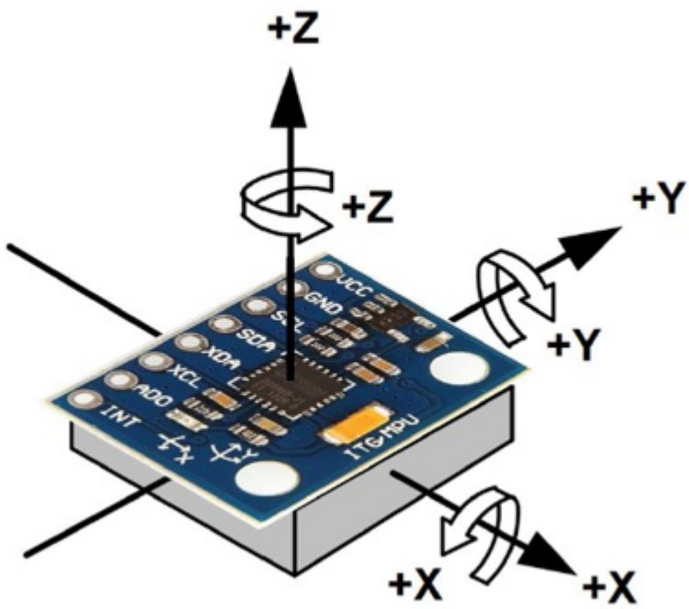
Los giroscopios utilizan un MEMS (MicroElectroMechanical Systems) para medir la velocidad angular usando el efecto Coriolis

Con un giroscopio podemos medir la velocidad angular, y si se integra la velocidad angular con respecto al tiempo se obtiene el desplazamiento angular (posición angular si se sabe dónde se inició el giro)

Módulo Acelerómetro y giroscopio MPU6050

EL módulo Acelerómetro MPU tiene un giroscopio de tres ejes con el que podemos medir velocidad angular y un acelerómetro también de 3 ejes con el que medimos los componentes X, Y y Z de la aceleración.

La dirección de los ejes está indicado en el módulo el cual hay que tener en cuenta para no equivocarnos en el signo de las aceleraciones.



La comunicación del módulo es por I2C, esto le permite trabajar con la mayoría de microcontroladores. Los pines SCL y SDA tienen una resistencia pull-up en placa para una conexión directa al microcontrolador o Arduino.

Tenemos dos direcciones I2C para poder trabajar:

Pin AD0	Dirección I2C
AD0=HIGH (5V)	0x69
AD0=LOW (GND o NC)	0x68

El pin ADDR internamente en el módulo tiene una resistencia a GND, por lo que si no se conecta, la dirección por defecto será 0x68.

El módulo tiene un regulador de voltaje en placa de 3.3V, el cual se puede alimentar con los 5V del Arduino.

Librería Para el PMU6050

En este tutorial trabajaremos con la librería desarrollada por Jeff Rowberg, la librería se descargar en:

https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050

Esta librería trabaja con una librería adicional para la comunicación I2C:

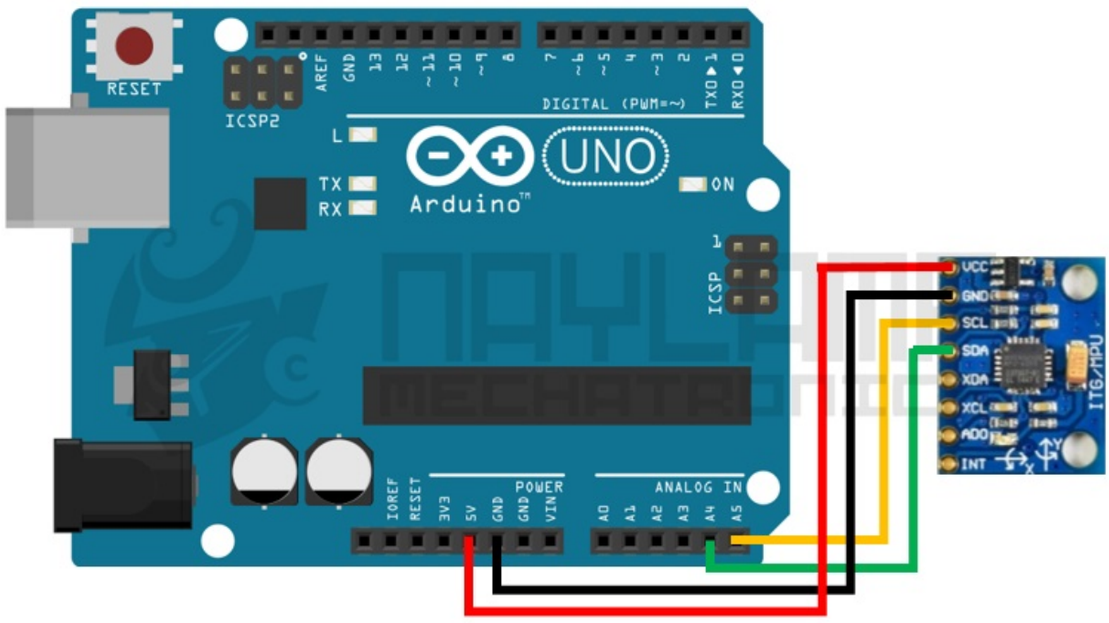
https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/I2Cdev

Para trabajar los siguientes ejercicios es necesario instalar las librerías en el IDE Arduino.

Conexiones entre Arduino y el MPU6050

Las conexiones son del modo I2C estándar:

MPU6050	Arduino Uno, Nano, Mini	Arduino Mega, DUE	Arduino Leonardo
VCC	5V	5V	5V
GND	GND	GND	GND
SCL	A5	21	3
SDA	A4	20	2



Ej.1: Realizar mediciones del MPU6050:

En este ejemplo realizaremos lecturas tanto del acelerómetro como del giroscopio.

El sketch para este ejercicio:

```
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;

void setup() {
  Serial.begin(57600);      //Iniciando puerto serial
  Wire.begin();             //Iniciando I2C
  sensor.initialize();      //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
  else Serial.println("Error al iniciar el sensor");
}

void loop() {
  // Leer las aceleraciones y velocidades angulares
  sensor.getAcceleration(&ax, &ay, &az);
  sensor.getRotation(&gx, &gy, &gz);

  //Mostrar las lecturas separadas por un [tab]
  Serial.print("a[x y z] g[x y z]:\t");
  Serial.print(ax); Serial.print("\t");
  Serial.print(ay); Serial.print("\t");
  Serial.print(az); Serial.print("\t");
  Serial.print(gx); Serial.print("\t");
  Serial.print(gy); Serial.print("\t");
  Serial.println(gz);

  delay(100);
}
```

Repasemos las funciones de la librería utilizada en este ejercicio.

Inicialmente como se indico es necesario incluir las siguientes 3 librerías

```
#include "I2Cdev.h"
#include "MPU6050.h"
```

```
#include "Wire.h"
```

Posteriormente crear la variable u objeto para el MPU6050, que en nuestro caso tiene el nombre de: “sensor”

```
MPU6050 sensor;
```

En este caso la dirección I2c es 0x68, pero si deseamos trabajar con otra dirección debemos modificar la linea anterior:

```
MPU6050 sensor(0x69);
```

Posteriormente en el void loop() inicializamos tanto la comunicación I2C como el MPU6050 y en nuestro caso la comunicación serial puesto que la usaremos más adelante:

```
  Serial.begin(57600);      //Iniciando puerto serial
  Wire.begin();               //Iniciando I2C
  sensor.initialize();      //Iniciando el sensor
```

Al inicializar el sensor, los rangos por defecto serán:

- acelerómetro -2g a +2g

- giroscopio: -250°/sec a +250°/sec

Teniendo en cuenta que la resolución de las lecturas es de 16 bits por lo que el rango de lectura es de -32768 a 32767.

En el void loop() realizamos las lecturas y las guardamos en sus variables respectivas, esto se hace con las siguientes funciones:

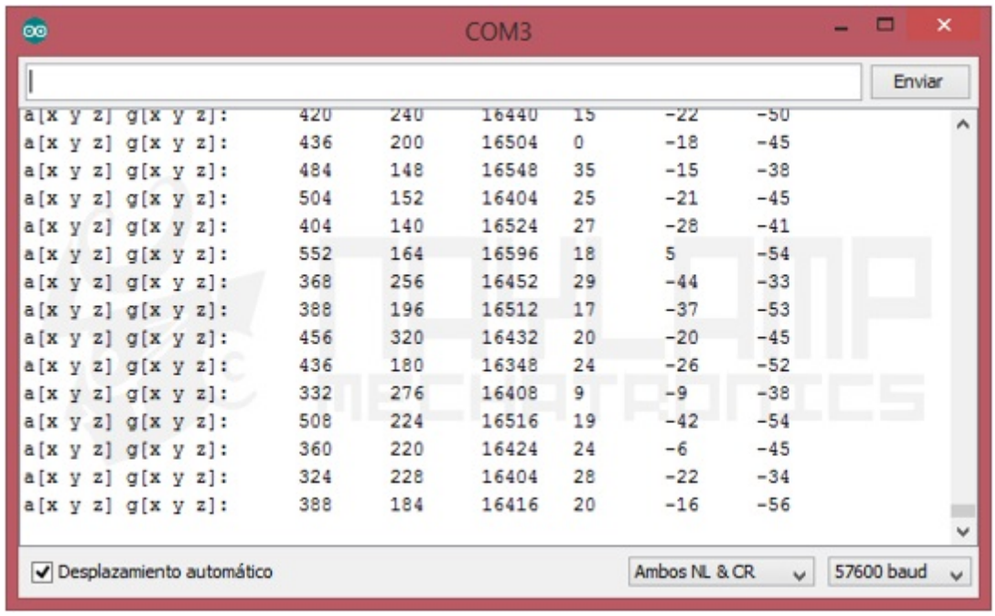
```
  sensor.getAcceleration(&ax, &ay, &az);
  sensor.getRotation(&gx, &gy, &gz);
```

La primera función lee la aceleración de los componentes x-y-z como parámetro, es necesario dar la dirección de las variables como argumento, para lo que se usa: &variable.

La segunda función realiza la lectura de la velocidad angular y guarda las lecturas en sus respectivas variables.

Finalmente enviamos por el puerto serial los datos leídos.

Si ubicamos el sensor en posición horizontal obtendremos medidas similares a los que mostramos a continuación.



Conforme movamos el sensor los componentes de aceleración irán cambiando en función del ángulo del sensor, puesto que la gravedad siempre es paralela al eje "Z" y se descompondrá en las componentes x-y-z del sensor.

Ej.2: Calibrando nuestro MPU6050

En este ejemplo realizaremos la calibración del MPU6050, esto es necesario ya que el sensor MPU6050 probablemente no se encuentre 100% en una posición horizontal, esto debido a que el sensor al ser soldado en el módulo puede estar desnivelado agregando un error en cada componente. De igual forma cuando instalemos el módulo en nuestro proyecto, puede estar desnivelado a pesar que a simple vista lo notemos correctamente nivelado.

Para solucionar este problema, se puede configurar en el módulo MPU6050 OFFSETS y de esta forma compensar los errores que podamos tener.

El sketch para realizar la calibración es el siguiente:

```
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;

//Variables usadas por el filtro pasa bajos
long f_ax,f_ay, f_az;
int p_ax, p_ay, p_az;
long f_gx,f_gy, f_gz;
int p_gx, p_gy, p_gz;
int counter=0;

//Valor de los offsets
int ax_o,ay_o,az_o;
int gx_o,gy_o,gz_o;

void setup() {
  Serial.begin(57600);     //Iniciando puerto serial
```

```
  Wire.begin();                //Iniciando I2C
  sensor.initialize();      //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");

  // Leer los offset los offsets anteriores
  ax_o=sensor.getXAccelOffset();
  ay_o=sensor.getYAccelOffset();
  az_o=sensor.getZAccelOffset();
  gx_o=sensor.getXGyroOffset();
  gy_o=sensor.getYGyroOffset();
  gz_o=sensor.getZGyroOffset();

  Serial.println("Offsets:");
  Serial.print(ax_o); Serial.print("\t");
  Serial.print(ay_o); Serial.print("\t");
  Serial.print(az_o); Serial.print("\t");
  Serial.print(gx_o); Serial.print("\t");
  Serial.print(gy_o); Serial.print("\t");
  Serial.print(gz_o); Serial.print("\t");
  Serial.println("nnEnvie cualquier caracter para empezar la calibracionnn");
  // Espera un caracter para empezar a calibrar
  while (true){if (Serial.available()) break;}
  Serial.println("Calibrando, no mover IMU");

}

void loop() {
  // Leer las aceleraciones y velocidades angulares
  sensor.getAcceleration(&ax, &ay, &az);
  sensor.getRotation(&gx, &gy, &gz);

  // Filtrar las lecturas
  f_ax = f_ax-(f_ax>>5)+ax;
  p_ax = f_ax>>5;

  f_ay = f_ay-(f_ay>>5)+ay;
  p_ay = f_ay>>5;

  f_az = f_az-(f_az>>5)+az;
  p_az = f_az>>5;

  f_gx = f_gx-(f_gx>>3)+gx;
  p_gx = f_gx>>3;

  f_gy = f_gy-(f_gy>>3)+gy;
  p_gy = f_gy>>3;

  f_gz = f_gz-(f_gz>>3)+gz;
  p_gz = f_gz>>3;

  //Cada 100 lecturas corregir el offset
  if (counter==100){
    //Mostrar las lecturas separadas por un [tab]
    Serial.print("promedio:"); Serial.print("t");
    Serial.print(p_ax); Serial.print("\t");
    Serial.print(p_ay); Serial.print("\t");
    Serial.print(p_az); Serial.print("\t");
    Serial.print(p_gx); Serial.print("\t");
    Serial.print(p_gy); Serial.print("\t");
    Serial.println(p_gz);

    //Calibrar el acelerometro a 1g en el eje z (ajustar el offset)
    if (p_ax>0) ax_o--;
    else {ax_o++;}
    if (p_ay>0) ay_o--;
    else {ay_o++;}
    if (p_az-16384>0) az_o--;
    else {az_o++;}

    sensor.setXAccelOffset(ax_o);
    sensor.setYAccelOffset(ay_o);
    sensor.setZAccelOffset(az_o);

    //Calibrar el giroscopio a 0º/s en todos los ejes (ajustar el offset)
    if (p_gx>0) gx_o--;
    else {gx_o++;}
    if (p_gy>0) gy_o--;
    else {gy_o++;}
```

```
    if (p_gz>0) gz_o--;
    else {gz_o++;}

    sensor.setXGyroOffset(gx_o);
    sensor.setYGyroOffset(gy_o);
    sensor.setZGyroOffset(gz_o);

    counter=0;
  }
  counter++;
}
```

El programa básicamente está modificando constantemente los offset intentando eliminar el error con la medida real que deseamos, en esta caso ax=0,ay=0,az=1g y gx=0,gy=0,gz=0.

Inicialmente leemos los offsets actuales y esperamos que el usuario envía un carácter por el puerto serie.
Antes de enviar el carácter es necesario ubicar el sensor en posición horizontal y evitar moverlo durante la calibración, dicha posición será nuestro nivel para futuras mediciones.

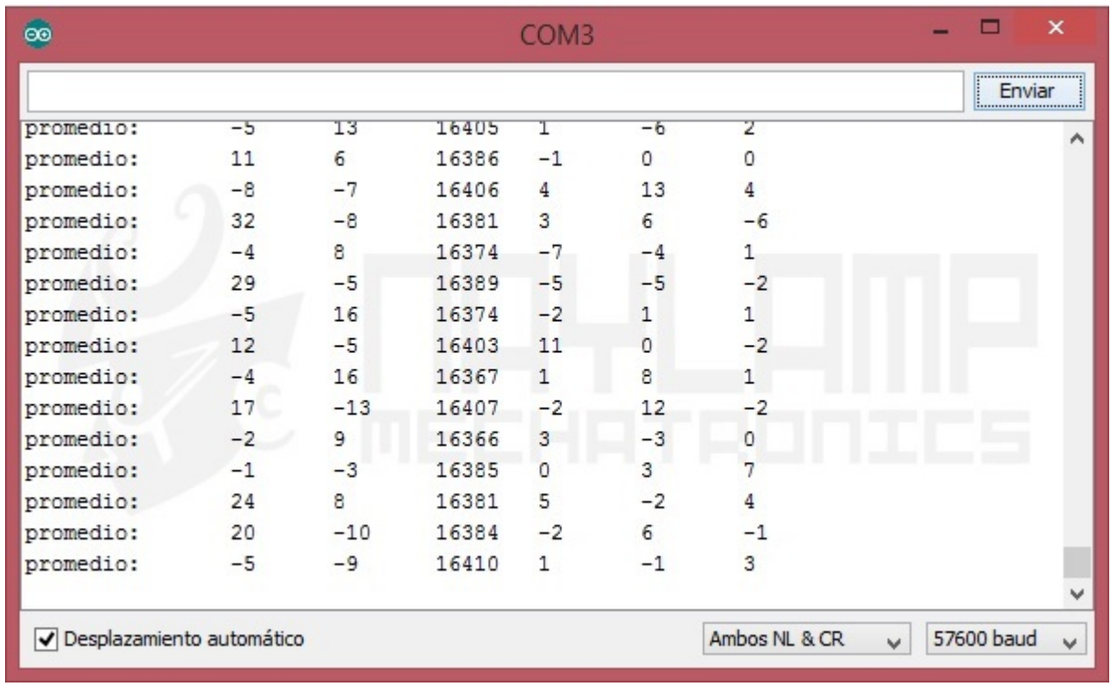
Después de enviar el carácter el programa realiza las lecturas tanto del acelerómetro como del giroscopio, usando un filtro estabilizamos un poco las lecturas y cada 100 lecturas comprobamos si los valores son cercanos a los valores que deseamos leer, dependiendo de esto se aumenta o disminuye los offsets. Esto hará que las lecturas filtradas converjan a:

-aceleración: p_ax=0 , p_ay=0 , p_az=+16384

-Velocidad angular: p_gx=0 , p_gy=0 , p_gz=0

Cuando en el monitor serial se observen valores cercanos a los anteriores debemos desconectar o reiniciar nuestro Arduino. Con esto el MPU6050 quedará configurado con el último offset calculado en el programa de calibración.

A continuación mostramos la salida del monitor serial después de enviar el carácter y esperar que los valores se acerquen a: 0 0 +16384 0 0 0



Cuando tengamos estos valores debemos reiniciar el Arduino o simplemente cerrar y abrir el monitor serial. Posteriormente podemos volver a cargar el primer ejemplo para probar las lecturas con los nuevos offsets.

Ej.3: Escalado de lecturas

En este ejemplo vamos a escalar las lecturas a valores con las unidades de aceleración y velocidad angular.

Primero tenemos que saber los rangos con los que está configurado nuestro MPU6050, dichos rangos pueden ser 2g/4g/8g/16g para el acelerómetro y 250/500/1000/2000(°/s) para el giroscopio.

Para este ejemplo trabajaremos con los rangos por defecto (2g y 250°/s):

Variable	valor mínimo	valor central	valor máximo
Lectura MPU6050	-32768	0	+32767
Aceleración	-2g	0g	+2g
Velocidad angular	-250°/s	0°/s	+250°/s

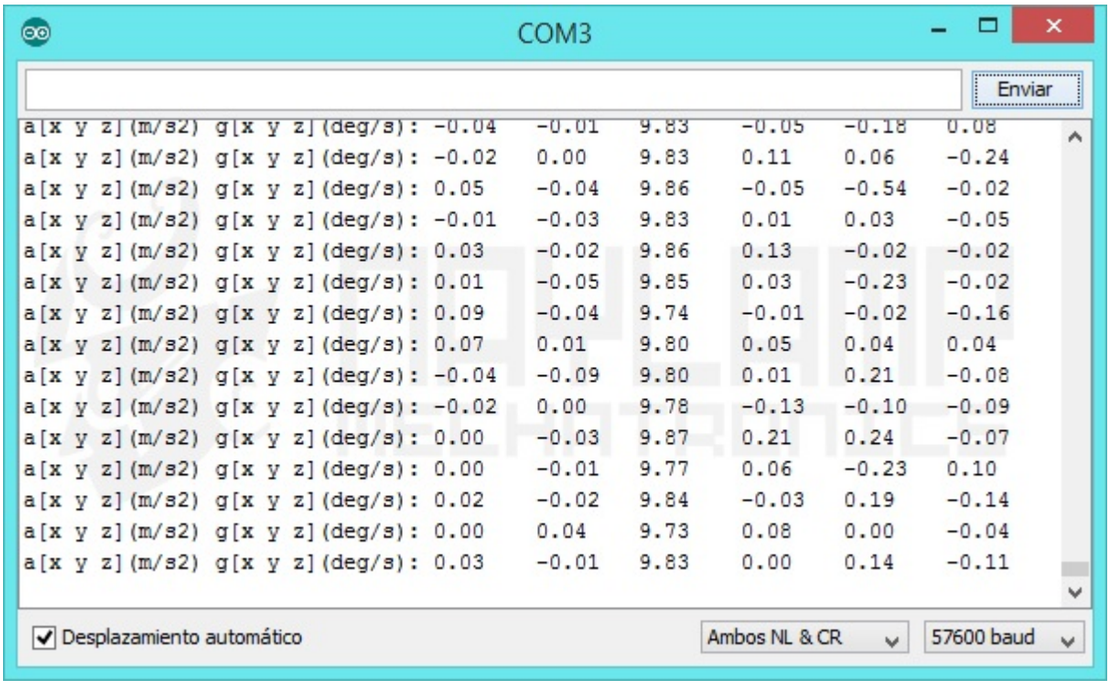
Para escalar simplemente debemos usar una ecuación para convertir el valor leído en un valor de aceleración o velocidad angular.

A continuación se muestra el sketch con la ecuación correspondiente para escalar los valores:

```
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;

void setup() {
  Serial.begin(57600);      //Iniciando puerto serial
  Wire.begin();             //Iniciando I2C
  sensor.initialize();      //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
  else Serial.println("Error al iniciar el sensor");
}

void loop() {
  // Leer las aceleraciones y velocidades angulares
  sensor.getAcceleration(&ax, &ay, &az);
  sensor.getRotation(&gx, &gy, &gz);
  float ax_m_s2 = ax * (9.81/16384.0);
  float ay_m_s2 = ay * (9.81/16384.0);
  float az_m_s2 = az * (9.81/16384.0);
  float gx_deg_s = gx * (250.0/32768.0);
  float gy_deg_s = gy * (250.0/32768.0);
  float gz_deg_s = gz * (250.0/32768.0);
  //Mostrar las lecturas separadas por un [tab]
  Serial.print("a[x y z](m/s2) g[x y z](deg/s):\t");
  Serial.print(ax_m_s2); Serial.print("\t");
  Serial.print(ay_m_s2); Serial.print("\t");
  Serial.print(az_m_s2); Serial.print("\t");
  Serial.print(gx_deg_s); Serial.print("\t");
  Serial.print(gy_deg_s); Serial.print("\t");
  Serial.println(gz_deg_s);

  delay(100);
}
```

Los valores que tenemos ahora ya están escalados a unidades de aceleración y velocidad angular. En nuestro caso hemos convertido la aceleración a valores en m/s^2 por lo que se reemplazó el valor de g=9.81, si se desea trabajar en unidades "g" no es necesario este último paso.

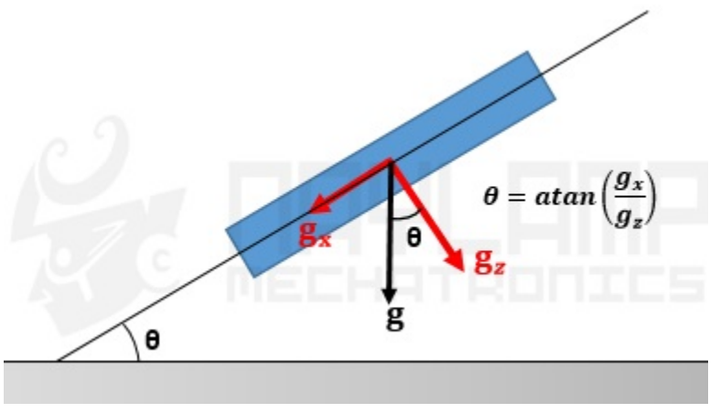
Si el sensor está en posición horizontal se debe obtener mediciones cercanas a 9.8 m/s^2 (aceleración de la gravedad terrestre) en la componente z de la aceleración.



Ej.4: Calculando el ángulo de inclinación con el acelerómetro del MPU6050

Si tenemos en cuenta que la única fuerza que actúa sobre el sensor es la fuerza de la gravedad. Entonces los valores que obtenemos en las componentes del acelerómetro corresponden a la gravedad y los ángulos de la resultante serán la inclinación del plano del sensor, puesto que la gravedad siempre es vertical.

Para entenderlo mejor, asumiremos que estamos en un plano X-Z e inclinamos el MPU6050 un ángulo θ, dicho ángulo se calcula de la siguiente forma:



Lo anterior nos sirve para calcular el ángulo en un plano 2D, pero para calcular los ángulos de inclinación en un espacio 3D tanto en X como en Y usamos las siguientes formulas:

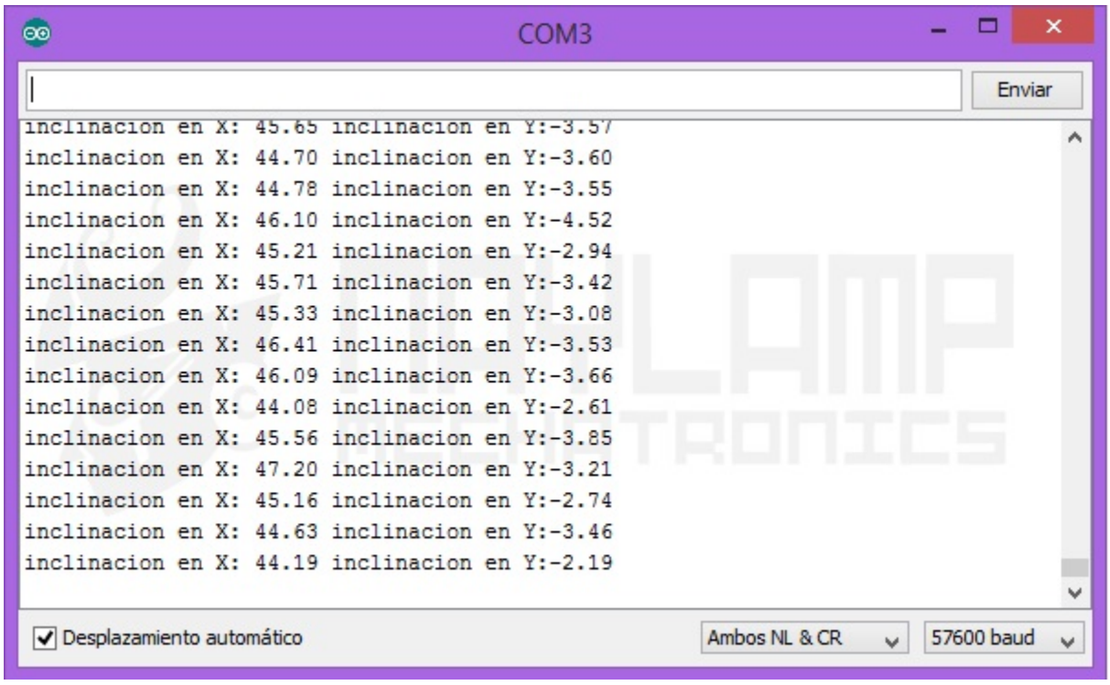
$$\theta_x = \tan^{-1}\left(\frac{a_x}{\sqrt[2]{a_Y{}^2 + a_z{}^2}}\right)$$

$$\theta_y = \tan^{-1}\left(\frac{a_y}{\sqrt[2]{a_x{}^2 + a_z{}^2}}\right)$$

Tener en cuenta que estamos calculando el ángulo de inclinación, si deseáramos el ángulo de rotación es decir por ejemplo el ángulo que rota el eje x en su mismo eje, entonces en las formulas necesitamos cambiar el a_y por el a_x y viceversa.
El sketch para calcular los ángulos de inclinación es el siguiente:

```
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro  en los ejes x,y,z
int ax, ay, az;

void setup() {
  Serial.begin(57600);      //Iniciando puerto serial
  Wire.begin();             //Iniciando I2C
  sensor.initialize();      //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
  else Serial.println("Error al iniciar el sensor");
}

void loop() {
  // Leer las aceleraciones
  sensor.getAcceleration(&ax, &ay, &az);
  //Calcular los angulos de inclinacion:
  float accel_ang_x=atan(ax/sqrt(pow(ay,2) + pow(az,2)))*(180.0/3.14);
  float accel_ang_y=atan(ay/sqrt(pow(ax,2) + pow(az,2)))*(180.0/3.14);
  //Mostrar los angulos separadas por un [tab]
  Serial.print("Inclinacion en X: ");
  Serial.print(accel_ang_x);
  Serial.print("tInclinacion en Y:");
  Serial.println(accel_ang_y);
  delay(10);
}
```

A continuación mostramos los resultados al tener inclinado el MPU6050 aproximadamente 45° con respecto a X:



Esto funciona solo si la única aceleración presente es la gravedad, pero si movemos rápidamente el MPU y sin realizar ninguna inclinación el ángulo que obtenemos con el programa anterior varía, generándonos errores para estos casos.

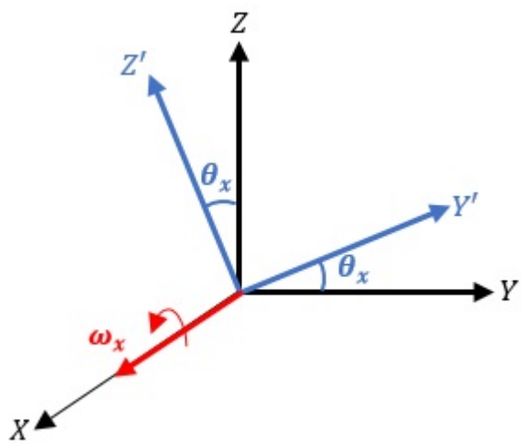
Ej.5: Calculando el ángulo de rotación usando el giroscopio del MPU5060

Como se explicó al inicio el giroscopio nos entrega la velocidad angular, y para calcular el ángulo actual necesitamos integrar la velocidad y conocer el ángulo incial. Esto lo hacemos usando la siguiente formula:

$$\theta_x = \theta_{x_0} + \omega_x \, \Delta t$$

$$\theta_y = \theta_{y_0} + \omega_y \, \Delta t$$

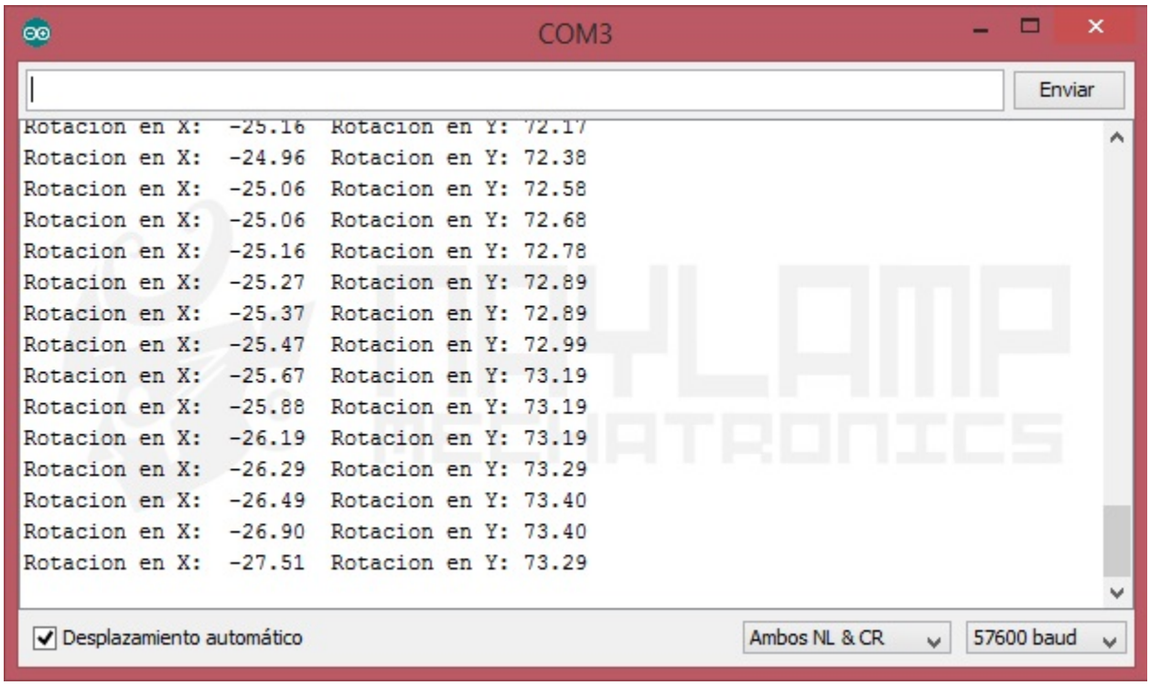
Tener en cuenta que cuando nos referimos a θx nos referimos al ángulo que gira el eje X sobre su propio eje. En la siguiente imagen se observa que la velocidad angular es perpendicular al plano de rotación.



Para calcular los ángulos de rotación tanto en el eje X como en Y usamos el siguiente sketch:

```
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int gx, gy, gz;

long tiempo_prev, dt;
float girosc_ang_x, girosc_ang_y;
float girosc_ang_x_prev, girosc_ang_y_prev;

void setup() {
  Serial.begin(57600);      //Iniciando puerto serial
  Wire.begin();             //Iniciando I2C
  sensor.initialize();      //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
  else Serial.println("Error al iniciar el sensor");
  tiempo_prev=millis();
}



void loop() {
  // Leer las velocidades angulares
  sensor.getRotation(&gx, &gy, &gz);

  //Calcular los angulos rotacion:

  dt = millis()-tiempo_prev;
  tiempo_prev=millis();

  girosc_ang_x = (gx/131)*dt/1000.0 + girosc_ang_x_prev;
  girosc_ang_y = (gy/131)*dt/1000.0 + girosc_ang_y_prev;

  girosc_ang_x_prev=girosc_ang_x;
  girosc_ang_y_prev=girosc_ang_y;

  //Mostrar los angulos separadas por un [tab]
  Serial.print("Rotacion en X: ");
  Serial.print(girosc_ang_x);
  Serial.print("tRotacion en Y: ");
  Serial.println(girosc_ang_y);

  delay(100);
}
```

Con este programa al girar aumentará o disminuirá el ángulo en función del sentido de giro del MPU



Tomar nota que la medida no es exacta incluso cuando no se mueve, el ángulo varía, o si se gira cierto ángulo y luego se regresa a la posición original el ángulo que medimos no es el inicial, esto se debe a que al integrar la velocidad angular y sumar el ángulo inicial hay un error producto de la mala medición del tiempo o del ruido en la lectura del MPU, el error por más pequeño que sea, se va acumulando en cada iteración y creciendo, este error es conocido como DRIFT.

Para disminuir el drift existen varios métodos, la mayoria aplica filtros para eliminar el ruido de las lecturas del sensor. También se pueden usar otros sensores como magnetómetros o acelerómetros y con los ángulos calculados con estos mejorar el cálculo del giroscopio.

Uno de los mejores filtros para eliminar el drift es el filtro Kalman, pero se necesita una buena capacidad de procesamiento computacional, haciéndolo difícil implementar en Arduino.

Otro filtro muy usado es el filtro de complemento, que mostramos a continuación:

Ej.6: Implementando un filtro de Complemento: acelerómetro + giroscopio

El filtro de complemento o en inglés "Complementary Filter" es uno de los más usados por su fácil implementación, combina el ángulo calculado por el giroscopio y el ángulo calculado por el acelerómetro.

La necesidad de combinar ambas lecturas es que si solo trabajámos con el acelerómetro, este es susceptible a las aceleraciones producto del movimiento del MPU o a fuerzas externas, pero en tiempos largos el ángulo no acumula errores. A diferencia que si trabajamos solo con el giroscopio si bien este no es susceptible a fuerzas externas, con el tiempo el drift es muy grande y nos sirve solo para mediciones de tiempos cortos.

La ecuación para calcular el ángulo usando el filtro de complemento es:

$$ángulo = 0.98\left(ángulo + \boldsymbol{\omega}_{\boldsymbol{giroscopio}}\, dt\right) + 0.02(\boldsymbol{ang}_{\boldsymbol{acelerómetro}})$$

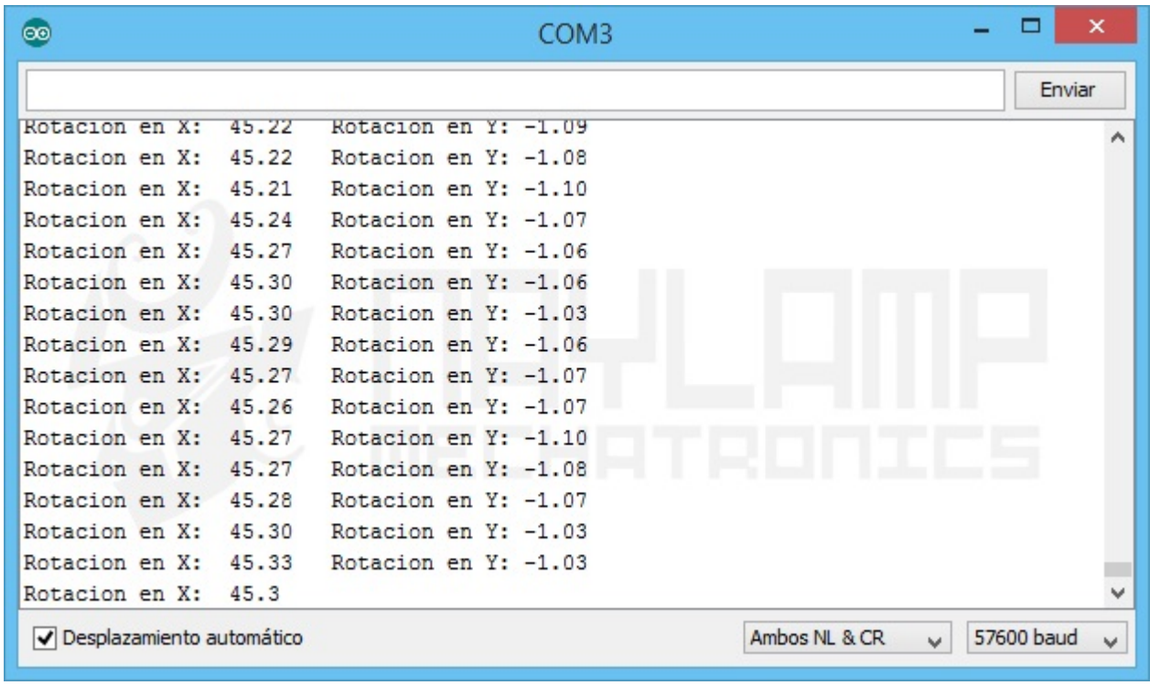
De esta forma el ángulo del acelerómetro está pasando por un filtro pasa bajos, amortiguando las variaciones bruscas de aceleración; y el ángulo calculado por el giroscopio tiene un filtro pasa

altos teniendo gran influencia cuando hay rotaciones rápidas. Podemos probar también con otros valores diferentes a 0.98 y 0.02 pero siempre deben de sumar 1.

a continuación mostramos el sketch para realizar esta tarea:

```
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;

// Valores RAW (sin procesar) del acelerometro y giroscopio en los ejes x,y,z
int ax, ay, az;
int gx, gy, gz;

long tiempo_prev;
float dt;
float ang_x, ang_y;
float ang_x_prev, ang_y_prev;

void setup() {
  Serial.begin(57600);      //Iniciando puerto serial
  Wire.begin();               //Iniciando I2C
  sensor.initialize();      //Iniciando el sensor

  if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
  else Serial.println("Error al iniciar el sensor");
}

void loop() {
  // Leer las aceleraciones y velocidades angulares
  sensor.getAcceleration(&ax, &ay, &az);
  sensor.getRotation(&gx, &gy, &gz);

  dt = (millis()-tiempo_prev)/1000.0;
  tiempo_prev=millis();

  //Calcular los ángulos con acelerometro
  float accel_ang_x=atan(ay/sqrt(pow(ax,2) + pow(az,2)))*(180.0/3.14);
  float accel_ang_y=atan(-ax/sqrt(pow(ay,2) + pow(az,2)))*(180.0/3.14);

  //Calcular angulo de rotación con giroscopio y filtro complemento
  ang_x = 0.98*(ang_x_prev+(gx/131)*dt) + 0.02*accel_ang_x;
  ang_y = 0.98*(ang_y_prev+(gy/131)*dt) + 0.02*accel_ang_y;


  ang_x_prev=ang_x;
  ang_y_prev=ang_y;

  //Mostrar los angulos separadas por un [tab]

  Serial.print("Rotacion en X:  ");
  Serial.print(ang_x);
  Serial.print("tRotacion en Y: ");
  Serial.println(ang_y);

  delay(10);
}
```

Ahora si movemos el MPU6050 rápidamente sin rotar, la variación del ángulo será mínima, además el drift se elimina y solo se nota en tiempos cortos.



Pueden adquirir los materiales usados en este tutorial en nuestra tienda:

- Arduino Uno R3
- Módulo MPU6050, Acelerómetro, Giroscopio I2C

https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html