Determinar la orientación con Arduino y el IMU MPU-6050

luisllamas.es/arduino-orientacion-imu-mpu-6050

¿Qué es un IMU MPU-6050?

El MPU-6050 es una unidad de medición inercial (IMU) de seis grados de libertad (6DOF) fabricado por Invensense, que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes.

La comunicación puede realizarse tanto por SPI como por bus I2C, por lo que es sencillo obtener los datos medidos. La tensión de alimentación es de bajo voltaje entre 2.4 a 3.6V.

Frecuentemente se encuentran integrados en módulos como el GY-521 que **incorporan la electrónica necesaria para conectarla de forma sencilla a un Arduino**. En la mayoría de los módulos, esto incluye un regulador de voltaje que permite alimentar directamente a 5V.

Dispone de conversores analógicos digitales (ADC) de 16bits. El rango del acelerómetro puede ser ajustado a ±2g, ±4g, ±8g, y ±16g, el del giroscopio a ±250, ±500, ±1000, and ±2000°/sec.

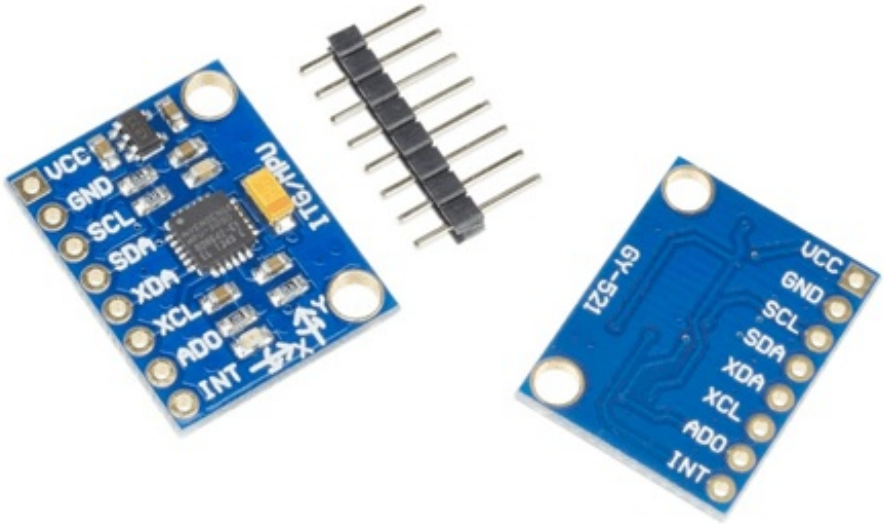
Es un sensor consume 3.5mA, con todos los sensores y el DMP activados. Dispone de un sensor de temperatura embebido, un reloj de alta precisión e interrupciones programables. También puede conectarse a otros dispositivos I2C como master.

El MPU-6050 incorpora un procesador interno (DMP Digital Motion Processor) que ejecuta complejos algortimos de MotionFusion para combinar las mediciones de los sensores internos, evitando tener que realizar los filtros de forma exterior.

El MPU-6050 es uno de los IMUs más empleados por su gran calidad y precio. Será uno de los componentes que con mayor frecuencia incorporaremos a nuestros proyectos de electrónica y robótica.

Precio

El MPU-6050 es un sensor excelente en realidad precio debido, entre otros factores, a que es ampliamente utilizado y esto ha permitido reducir su precio.

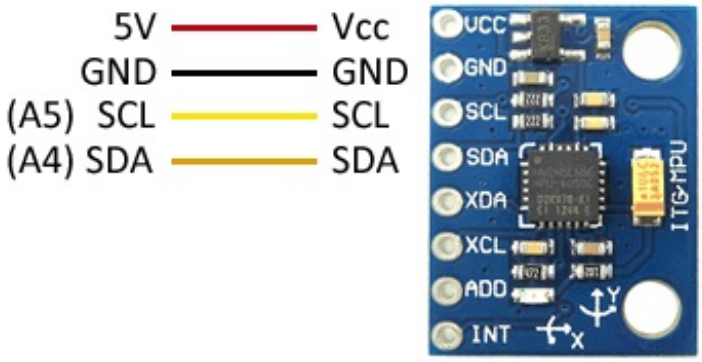


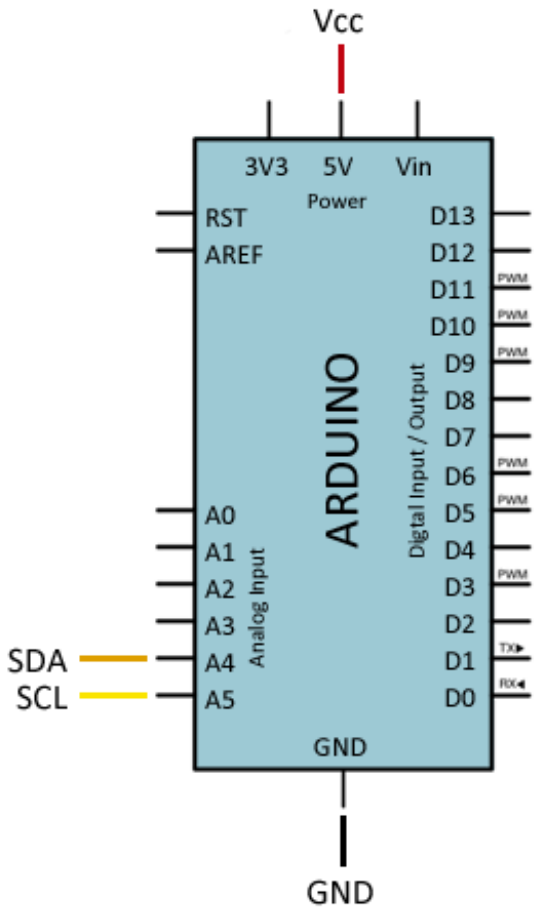
Podemos encontrarlo en placas de montaje como la GY-521, que incorpora la electrónica necesaria para conectarlo de forma sencilla a Arduino, por 1.10 € en vendedores internacionales de eBay o AliExpress.

Esquema montaje

La conexión es sencilla, simplemente alimentamos el módulo desde Arduino mediante GND y 5V y conectamos el pin SDA y SCL de Arduino con los pines correspondientes del sensor.

Mientras que la conexión vista desde el lado de Arduino quedaría así.





En Arduino Uno, Nano y Mini Pro, SDA es el pin A4 y el SCK el pin A5. Para otros modelos de Arduino consultar el **esquema patillaje** correspondiente.
Verificar que vuestra placa es compatible con 5V antes de conectarla a Arduino. Si no, tendréis que usar un adaptador de nivel lógico.

Ejemplos de código

Para realizar la lectura del MPU-6050 usaremos la librería desarrollada por Jeff Rowberg disponible en este enlace. También emplearemos la librería I2Cdev desarrollada por el mismo autor, que mejora la comunicación I2C.

La librería proporciona ejemplos de código, que resulta aconsejable revisar. Los siguientes ejemplos son modificaciones a partir de los disponibles en la librería.

Leer valores RAW

En el primer ejemplo, aprendemos a leer los valores directamente proporcionados por el MPU-6050 (valores RAW) a través del bus I2C. Los valores RAW tienen un rango de medición entre -32768 y +32767.

```
//GND - GND

//VCC - VCC

//SDA - Pin A4

//SCL - Pin A5

#include "I2Cdev.h"
```

```
#include "MPU6050.h"
#include "Wire.h"
constintmpuAddress=0x68;//Puede ser 0x68 o 0x69
MPU6050 mpu(mpuAddress);
intax,ay,az;
intgx,gy,gz;
voidprintTab()
{
Serial.print(F("\t"));
}
voidprintRAW()
{
Serial.print(F("a[x y z] g[x y z]:t" ));
Serial.print(ax);printTab();
Serial.print(ay);printTab();
Serial.print(az);printTab();
Serial.print(gx);printTab();
Serial.print(gy);printTab();
Serial.println(gz);
}
voidsetup()
{
Serial.begin(115200);
Wire.begin();
mpu.initialize();
Serial.println(mpu.testConnection()?F("IMU iniciado correctamente"):F("Error al iniciar IMU"));
}
voidloop()
{
// Leer las aceleraciones y velocidades angulares
mpu.getAcceleration(&ax,&ay,&az);
mpu.getRotation(&gx,&gy,&gz);
printRAW();
delay(100);
}
```

Leer valores en Sistema Internacional

En el siguiente ejemplo aplicamos una escala a los valores RAW para obtener mediciones con significado físico. En el ejemplo, emplearemos valores G para la aceleración, y º/S para la velocidad angular. Con facilidad podréis modificar el código para que proporcione los valores en unidades del Sistema Internacional.

El escalado dependerá del rango de medición que seleccionemos en el MPU-6050, que recordamos puede ser 2g/4g/8g/16g para el acelerómetro y 250/500/1000/2000 (°/s) para el giroscopio.

```
//GND - GND
//VCC - VCC
//SDA - Pin A4
//SCL - Pin A5
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
constintmpuAddress=0x68;// Puede ser 0x68 o 0x69
MPU6050 mpu(mpuAddress);
intax,ay,az;
intgx,gy,gz;
// Factores de conversion
constfloataccScale=2.0*9.81/32768.0;
constfloatgyroScale=250.0/32768.0;
voidprintTab()
{
Serial.print(F("\t"));
}
// Mostrar medidas en Sistema Internacional
voidprintRAW()
{
Serial.print(F("a[x y z](m/s2) g[x y z](deg/s):t" ));
Serial.print(ax*accScale);printTab();
Serial.print(ay*accScale);printTab();
Serial.print(az*accScale);printTab();
Serial.print(gx*gyroScale);printTab();
Serial.print(gy*gyroScale);printTab();
Serial.println(gz*accGyroScale);
```

```
}
voidsetup()
{
Serial.begin(115200);
Wire.begin();
mpu.initialize();
Serial.println(mpu.testConnection()?F("IMU iniciado correctamente"):F("Error al iniciar IMU"));
}
voidloop()
{
// Leer las aceleraciones y velocidades angulares
mpu.getAcceleration(&ax,&ay,&az);
mpu.getRotation(&gx,&gy,&gz);
printRAW();
delay(100);
}
```

Leer inclinación con acelerómetro

En el siguiente ejemplo, calculamos la inclinación del MPU-6050 mediante la proyección de la medición de la gravedad y las relaciones trigonométricas que vimos en la entrada Cómo usar un acelerómetro con Arduino.

```
//GND - GND
//VCC - VCC
//SDA - Pin A4
//SCL - Pin A5
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
constintmpuAddress=0x68;// Puede ser 0x68 o 0x69
MPU6050 mpu(mpuAddress);
intax,ay,az;
intgx,gy,gz;
voidsetup()
{
Serial.begin(115200);
Wire.begin();
mpu.initialize();
Serial.println(mpu.testConnection()?F("IMU iniciado correctamente"):F("Error al iniciar IMU"));
}
voidloop()
{
// Leer las aceleraciones
mpu.getAcceleration(&ax,&ay,&az);
//Calcular los angulos de inclinacion
floataccel_ang_x=atan(ax/sqrt(pow(ay,2)+pow(az,2)))*(180.0/3.14);
floataccel_ang_y=atan(ay/sqrt(pow(ax,2)+pow(az,2)))*(180.0/3.14);
// Mostrar resultados
Serial.print(F("Inclinacion en X: "));
Serial.print(accel_ang_x);
Serial.print(F("\tInclinacion en Y:"));
Serial.println(accel_ang_y);
delay(10);
}
```

Obtener orientación con giroscopio

En el siguiente ejemplo, realizamos la integración de la señal de la velocidad del giroscopio

para obtener la orientación del MPU-6050, como vimos en la entrada Cómo usar un giroscopio con Arduino.

```
//GND - GND
//VCC - VCC
//SDA - Pin A4
//SCL - Pin A5
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
constintmpuAddress=0x68;// Puede ser 0x68 o 0x69
MPU6050 mpu(mpuAddress);
intax,ay,az;
intgx,gy,gz;
longtiempo_prev,dt;
floatgirosc_ang_x,girosc_ang_y;
floatgirosc_ang_x_prev,girosc_ang_y_prev;
voidupdateGiro()
{
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;
}
voidsetup()
{
Serial.begin(115200);
Wire.begin();
mpu.initialize();
Serial.println(mpu.testConnection()?F("IMU iniciado correctamente"):F("Error al iniciar IMU"));
}
voidloop()
{
// Leer las velocidades angulares
```

```
mpu.getRotation(&gx,&gy,&gz);
updateGiro();
// Mostrar resultados
Serial.print(F("Rotacion en X:  "));
Serial.print(girosc_ang_x);
Serial.print(F("\tRotacion en Y: "));
Serial.println(girosc_ang_y);
delay(10);
}
```

Obtener la orientación con filtro complementario

Este ejemplo empleamos un filtro complementario para combinar la señal del acelerómetro y giroscopio para obtener una mejor medición de la orientación del MPU-6050, como vimos en la entrada Medir la inclinación de un IMU con Arduino y filtro complementario.

```
//GND - GND
//VCC - VCC
//SDA - Pin A4
//SCL - Pin A5
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
constintmpuAddress=0x68;// Puede ser 0x68 o 0x69
MPU6050 mpu(mpuAddress);
intax,ay,az;
intgx,gy,gz;
longtiempo_prev;
floatdt;
floatang_x,ang_y;
floatang_x_prev,ang_y_prev;
voidupdateFiltered()
{
dt=(millis()-tiempo_prev)/1000.0;
tiempo_prev=millis();
//Calcular los ángulos con acelerometro
floataccel_ang_x=atan(ay/sqrt(pow(ax,2)+pow(az,2)))*(180.0/3.14);
```

```
floataccel_ang_y=atan(-ax/sqrt(pow(ay,2)+pow(az,2)))*(180.0/3.14);
//Calcular angulo de rotación con giroscopio y filtro complementario
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;
}
voidsetup()
{
Serial.begin(115200);
Wire.begin();
mpu.initialize();
Serial.println(mpu.testConnection()?F("IMU iniciado correctamente"):F("Error al iniciar IMU"));
}
voidloop()
{
// Leer las aceleraciones y velocidades angulares
mpu.getAcceleration(&ax,&ay,&az);
mpu.getRotation(&gx,&gy,&gz);
updateFiltered();
Serial.print(F("Rotacion en X:  "));
Serial.print(ang_x);
Serial.print(F("\t Rotacion en Y: "));
Serial.println(ang_y);
delay(10);
}
```

Obtener la orientación mediante el DMP

En este último ejemplo empleamos el DMP integrado en el MPU-6050 para realizar la combinación de la medición del acelerómetro y el giroscopio, lo que proporciona mejor resultados que emplear un filtro complementario, y además libera a Arduino del proceso de cálculo.

```
//GND - GND
//VCC - VCC
//SDA - Pin A4
```

```
//SCL - Pin A5
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
constintmpuAddress=0x68;// Puede ser 0x68 o 0x69
MPU6050 mpu(mpuAddress);
booldmpReady=false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];
Quaternionq;// [w, x, y, z]         quaternion
VectorInt16 aa;// [x, y, z]            accel sensor measurements
VectorInt16 aaReal;// [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;// [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;// [x, y, z]            gravity vector
floateuler[3];// [psi, theta, phi]    Euler angle container
floatypr[3];// [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
volatileboolmpuInterrupt=false;
voiddmpDataReady()
{
mpuInterrupt=true;
}
voidsetup()
{
Wire.begin();
TWBR=24;
Serial.begin(115200);
mpu.initialize();
Serial.println(mpu.testConnection()?F("IMU iniciado correctamente"):F("Error al iniciar IMU"));
devStatus=mpu.dmpInitialize();
// Valores de calibración
mpu.setXGyroOffset(220);
mpu.setYGyroOffset(76);
```

```
mpu.setZGyroOffset(-85);
mpu.setZAccelOffset(1788);
if(devStatus==0)
{
Serial.println(F("Activando DMP..."));
mpu.setDMPEnabled(true);
attachInterrupt(0,dmpDataReady,RISING);
mpuIntStatus=mpu.getIntStatus();
dmpReady=true;
packetSize=mpu.dmpGetFIFOPacketSize();
}
else
{
Serial.print(F("Fallo inicialización DMP"));
}
}
voidloop()
{
if(!dmpReady)return;
// Esperar a la interrupción, o mientras queden datos por leer
while(!mpuInterrupt&&fifoCount<packetSize)
{
// AQUI EL RESTO DE ACCIONES DE TU PROGRAMA
}
mpuInterrupt=false;
mpuIntStatus=mpu.getIntStatus();
fifoCount=mpu.getFIFOCount();
// Comprobar overflow
if((mpuIntStatus&0x10)||fifoCount==1024)
{
// Reset FIFO
mpu.resetFIFO();
Serial.println(F("FIFO overflow!"));
}
elseif(mpuIntStatus&0x02)
```

```
{
// Esperar datos
while(fifoCount<packetSize)fifoCount=mpu.getFIFOCount();
// Leer paquete de FIFO
mpu.getFIFOBytes(fifoBuffer,packetSize);
fifoCount-=packetSize;
// Mostrar angulos euler
mpu.dmpGetQuaternion(&q,fifoBuffer);
mpu.dmpGetGravity(&gravity,&q);
mpu.dmpGetYawPitchRoll(ypr,&q,&gravity);
Serial.print("ypr\t");
Serial.print(ypr[0]*180/M_PI);
Serial.print("\t");
Serial.print(ypr[1]*180/M_PI);
Serial.print("\t");
Serial.println(ypr[2]*180/M_PI);
// Mostar las aceleracinones corregidas
mpu.dmpGetQuaternion(&q,fifoBuffer);
mpu.dmpGetAccel(&aa,fifoBuffer);
mpu.dmpGetGravity(&gravity,&q);
mpu.dmpGetLinearAccel(&aaReal,&aa,&gravity);
mpu.dmpGetLinearAccelInWorld(&aaWorld,&aaReal,&q);
Serial.print("aworld\t");
Serial.print(aaWorld.x);
Serial.print("\t");
Serial.print(aaWorld.y);
Serial.print("\t");
Serial.println(aaWorld.z);
}
}
```

Si te ha gustado esta entrada y quieres leer más sobre Arduino puedes consultar la sección **tutoriales de Arduino**