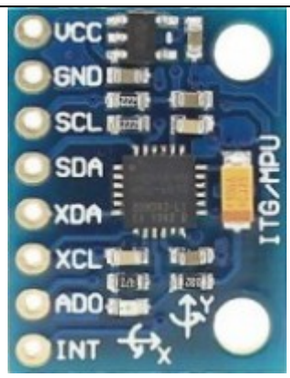
Sensor MPU6050. “*El primer contacto*”

INTRODUCCIÓN.

Cuando iniciamos una nueva tarea que requiere de un nuevo dispositivo al que no habíamos tenido la ocasión de utilizar, lo propio es tener dudas y naturalmente debemos buscamos información que nos aclare los conceptos y conocimientos que se requieren para manejar, y dominar el nuevo «juguete».

El MPU-6050 no es caro, especialmente teniendo en cuenta el hecho de que combina un acelerómetro y un giroscopio en el mismo dispositivo, ademas mide temperatura.

La temperatura que mide es interna, en el interior del chip. Pero como el chip usa muy poca corriente y no se calienta, es casi igual a la temperatura exterior.
Sin embargo, es mejor usar otros sensores para mediciones de temperatura precisas.
Si alguien mide temperatura con este dispositivo es porque seguramente lo esta utilizando en sus funciones principales y no solo para medir temperatura.



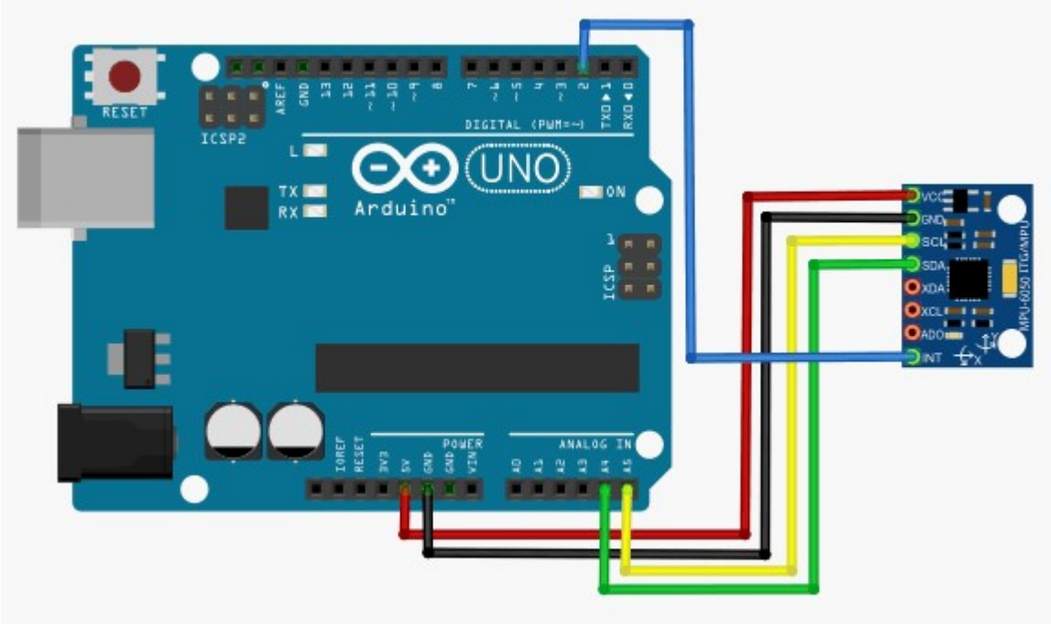
Boceto Corto Ejemplo.

El boceto corto de ejemplo es un esbozo muy breve y muestra todos los valores brutos (de acelerómetro, giroscopio y temperatura). El esquema práctico se muestra en la figura 3. Debería funcionar en Arduino Uno, Nano, Leonardo, y también Due.

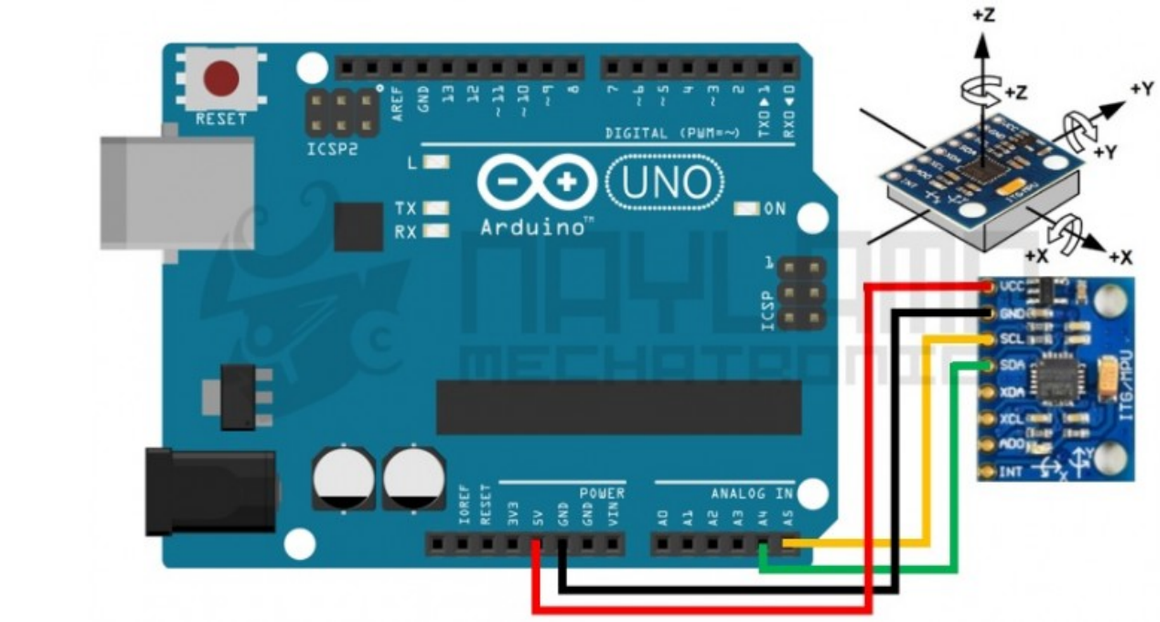
```
/*El boceto corto de ejemplo es un esbozo muy breve y

 * muestra todos los valores brutos (de acelerómetro,

 * giroscopio y temperatura). El esquema práctico se

 * muestra en la figura 3. Debería funcionar en Arduino Uno,

 * Nano, Leonardo, y también Due.

 *

 */



// MPU-6050 Short Example Sketch

// By Arduino User JohnChi

// August 17, 2014

// Public Domain

#include<Wire.h>

const int MPU=0x68;  // I2C address of the MPU-6050

int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;

void setup(){

  Wire.begin();
```

```
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);  // PWR_MGMT_1 register
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  Serial.begin(9600);
}
void loop(){
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,14,true);  // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

  Serial.println("MPU-6050");
  Serial.print("Datos del Acelerometro.");
  Serial.print("AcX = "); Serial.print(AcX);
  Serial.print(" | AcY = "); Serial.print(AcY);
  Serial.print(" | AcZ = "); Serial.print(AcZ);
  Serial.print(" | Tmp = "); Serial.println(Tmp/340.00+36.53);  //equation for temperature in grados C a partir de
hoja de datos

  Serial.println("Datos del Giroscopo.");
  Serial.print("GyX = "); Serial.print(GyX);
  Serial.print(" | GyY = "); Serial.print(GyY);
  Serial.print(" | GyZ = "); Serial.println(GyZ);
  delay(333);
}
```

Conexionado para probar el programa anterior. LEA CON ATENCION



PERO UD PUEDE PROBAR CON EL SIGUIENTE (ACONSEJABLE PARA LOS PRINCIPIANTES)



ACLARACION de opiniones de foros de Arduino:

El INTes la señal de interrupción, por lo general la salida de drenaje abierto y se puede conectar a muchos de ellos a la única línea de interrupción (pero hay que comprobar todos los dispositivos cuáles están listos). El LOW nivel del dispositivo indica que los datos están listos (la conversión se completó). Por lo tanto, no tiene que leer periódicamente el registro de estado para averiguarlo (no es necesario realizar una encuesta).

En resumen, el uso INT es opcional.

En el primer nivel son opcionales, y al menos la mitad del código, si no más, nunca los usa. Si los conectas, debes encuestarlos a todos para ver cuál está listo.

Basta con conectar los pines de alimentación y montar los pines necesarios para el control del bus I2C. Cuando el IMU dispone de alguna medida se lo informa a nuestro Arduino mediante una interrupción y por eso conectamos el pin INT a nuestra primera interrupción en el pin 2. (PERO EL PROGRAMA USADO LO DEBE CONTEMPLAR).

*Si vas a usar un **Arduino Mega** la conexión es ligeramente diferente como se muestra aquí*