Projet NMEA log évènements

Contenu

[Relier l'alternateur à un Arduino. Schéma interface ??? 4](#_Toc473876659)

[Tiersp : Montage pour transmetre l'état de la mer et la gite en NMEA 5](#_Toc473876660)

[SparkFun Triple Axis Accelerometer Breakout - ADXL335 SEN-09269 8](#_Toc473876661)

[Tuto shield sparkfun Triple Axis Magnetometer - HMC5883L Breakout Quickstart Guide 10](#_Toc473876662)

[Overview: 10](#_Toc473876663)

[Requirements: 11](#_Toc473876664)

[How it Works: 11](#_Toc473876665)

[How to Use it: 11](#_Toc473876666)

[Hardware: 11](#_Toc473876667)

[Software: 12](#_Toc473876668)

[Firmware: 12](#_Toc473876669)

[Resources: 14](#_Toc473876670)

[Conclusion: 14](#_Toc473876671)

[Comments 14](#_Toc473876672)

[Guide gyro de Sparkfun 16](#_Toc473876673)

[Accelerometer, Gyro and IMU Buying Guide 16](#_Toc473876674)

[LCD Button Shield V2 DEV-13293 17](#_Toc473876675)

[How to Make Arduino Board: The Easy Way 19](#_Toc473876676)

[MPU-9250 and Arduino (9-Axis IMU) 24](#_Toc473876677)

[Wiring 25](#_Toc473876678)

[Arduino mega 25](#_Toc473876679)

[Source code 25](#_Toc473876680)

[SparkFun IMU Breakout - MPU-9250 SEN-13762 29](#_Toc473876681)

[MPU-9250 Hookup Guide 31](#_Toc473876682)

[Introduction 31](#_Toc473876683)

[SparkFun IMU Breakout - MPU-9250 31](#_Toc473876684)

[Suggested Reading 31](#_Toc473876685)

[Board Overview 32](#_Toc473876686)

[PTH Connections 32](#_Toc473876687)

[Jumpers 33](#_Toc473876688)

[Reducing size 33](#_Toc473876689)

[Hardware Connections 34](#_Toc473876690)

[Library and Example Code 37](#_Toc473876691)

[Magnetic declination 43](#_Toc473876692)

[Resources and Going Further 46](#_Toc473876693)

[Arduino MPU 6050 – Best IMU Sensor Tutorial 47](#_Toc473876694)

[MMA8452 3 axes 54](#_Toc473876695)

[MMA8452Q Accelerometer Breakout Hookup Guide Sparkfun 56](#_Toc473876696)

[Introduction 56](#_Toc473876697)

[Covered In This Tutorial 57](#_Toc473876698)

[Required Materials 57](#_Toc473876699)

[Suggested Reading 57](#_Toc473876700)

[Hardware Overview 57](#_Toc473876701)

[Voltage Supply Requirements 59](#_Toc473876702)

[Address Select Jumper 59](#_Toc473876703)

[Example Hookup 60](#_Toc473876704)

[Solder Something 60](#_Toc473876705)

[SparkFun Triple Axis Accelerometer Breakout - MMA8452Q (with Headers) 61](#_Toc473876706)

[Simple Hookup 61](#_Toc473876707)

[Level Shifting 62](#_Toc473876708)

[Example Code 63](#_Toc473876709)

[Load Up the Example Sketch 63](#_Toc473876710)

[Using the SFE\_MMA8452Q Library 65](#_Toc473876711)

[Resources and Going Further 67](#_Toc473876712)

[Going Further …/… Des liens vers des montages 67](#_Toc473876713)

[MMA8451 9 axes 68](#_Toc473876714)

[Adafruit MMA8451 accelerometer 69](#_Toc473876715)

[Wiring & Test 69](#_Toc473876716)

[Download Adafruit\_MMA8451 70](#_Toc473876717)

[Download Adafruit\_Sensor 70](#_Toc473876718)

[Load Demo 70](#_Toc473876719)

[Library Reference 72](#_Toc473876720)

[Set & Get Range 72](#_Toc473876721)

[Read Raw Count Data 73](#_Toc473876722)

[Reading Normalized Adafruit\_Sensor data 73](#_Toc473876723)

[Read Orientation 73](#_Toc473876724)

<http://www.hisse-et-oh.com/forums/equipements/messages/2083683-relier-l-alternateur-a-un-arduino-schema-interface>

<http://patricktiercelin.wixsite.com/logiciels-tiercelin1/capteur-etat-de-la-mer-et-gite>

<https://www.sparkfun.com/products/9269>

<https://www.sparkfun.com/tutorials/301>

<https://www.sparkfun.com/pages/accel_gyro_guide>

<https://www.sparkfun.com/products/13293>

<http://diyhacking.com/make-arduino-board-and-bootload/>

<http://www.lucidarme.me/?p=5057>

<https://www.sparkfun.com/products/13762>

<https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide>

<https://diyhacking.com/arduino-mpu-6050-imu-sensor-tutorial/>

MMA8452 3C E-commerce store

<https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide>

MMA8451 ModuleFans <https://fr.aliexpress.com/store/612195>

<https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test>

# Relier l'alternateur à un Arduino. Schéma interface ???

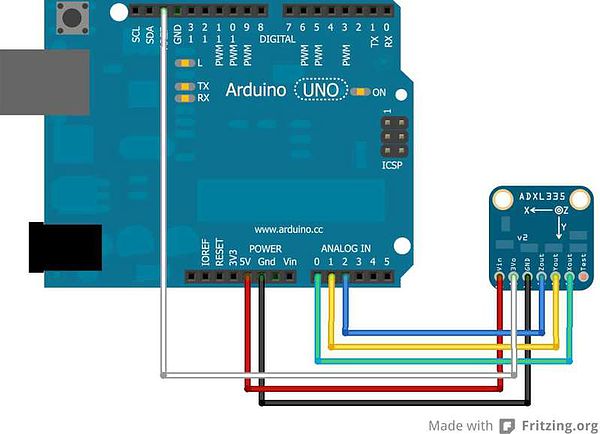
<http://www.hisse-et-oh.com/forums/equipements/messages/2083683-relier-l-alternateur-a-un-arduino-schema-interface>

# Tiersp : Montage pour transmetre l'état de la mer et la gite en NMEA

<http://patricktiercelin.wixsite.com/logiciels-tiercelin1/capteur-etat-de-la-mer-et-gite>

Montage Arduino pour le capteur trois axes.

Branchement de la carte Arduino UNO :



Remarques:

Certains capteurs n'ont pas de reférence ( fil blanc ) cela fonctionne quand même.

Il faut étalonner le capteur ( déterminer les mini et maxi de l'axe z ) ici : 270 et 405 le capteur étant à plat, puis retounéde 180°.

On peut utiliser ce petit programme  et en bougeant le capteur de haut en bas et de bas en haut rapidement

const int pin\_x = A0;

const int pin\_y = A1;

const int pin\_z = A2;

const float aref = 3.3;

int minz;

int maxz;

void setup()

{

  Serial.begin(9600);

  minz=500;

  maxz=0;

}

void loop()

{

  int x;

  int y;

  int z;

  x = analogRead(0);

  y = analogRead(1);

  z=analogRead(2);

  if (z<minz) minz=z;

  if (z>maxz) maxz=z;

  Serial.print("x=");

  Serial.println(x);

  Serial.print("y=");

  Serial.println(y);

  Serial.print("minz=");

  Serial.println(minz);

  Serial.print("maxz=");

  Serial.println(maxz);

  delay(1000);

}

Le Programme  etat de la mer et angle de gite :

String laphrasemer;

String merini;

String etatmer;

String laphrasegite;

String giteini;

String gite;

int check;

byte  etatdelamer = 0;

byte lagite=0;

const int xPin = 0;

const int yPin = 1;

const int zPin = 2;

//il faut ajuster ces quatre valeurs

int minVal = 270;

int maxVal = 405;

int minValac = 270;

int maxValac = 410;

double x;

double y;

double z;

byte xx;

byte nb=0;

double zz=0;

byte debut;

void setup()

{

  Serial.begin(9600);

  nb=0;

  zz=0;

}

void loop(){

  zz=0;

   int xRead = analogRead(xPin);

  int zRead = analogRead(zPin);

  int yRead = analogRead(yPin);

  //convertion en degrees -90 to 90 - pour atan2

  int xAng = map(xRead, minVal, maxVal, -90, 90);

  int yAng = map(yRead, minVal, maxVal, -90, 90);

  int zAng = map(zRead, minVal, maxVal, -90, 90);

  //faire dix mesures pour l'état de la mer avec l'accélération verticale

  nb=0;

  do

  {

    delay(50);

    z=analogRead(zPin);

    if (z>zz)     zz=z;

    nb++;

  }

  while (nb<10);

  //calcule de l'angle de gite

  x = RAD\_TO\_DEG \* (atan2(-yAng, -zAng) + PI);

  if (x>180)  x=abs(x-360);

  xx=byte(x);

  lagite= xx;

  //fabriquer la trame nmea

  merini = "$ERRMT";

  etatmer = String(calculetatmer(zz));

  gite = String(lagite);

  laphrasegite = merini + ",,,,,,," + etatmer + ","+gite+",R" ;

  check = getCheckSum(laphrasegite);

  laphrasegite = laphrasegite+"\*";

  Serial.print(laphrasegite);

  Serial.println(check,HEX);

 }

byte calculetatmer(float zz){

  byte etat;

  float z;

  z=abs(414-zz);

  if (z<40) etat=0;

  if ((z>=40)&& (z<80)) etat=1;

  if ((z>=80)&& (z<120)) etat=2;

  if ((z>=120)&& (z<160)) etat=3;

  if ((z>=160) && (z<200)) etat=4;

  if ((z>=200) && (z<240)) etat=5;

  if (z>=240) etat=6;

  return etat;

}

int getCheckSum(String phrase) {

  int checksum = 0;

  int longueur = 0;

  longueur = phrase.length();

  for(int i = 1; i < longueur; i++)

  {

    checksum = checksum ^ int(phrase[i]);

  }

  return checksum;

}

# SparkFun Triple Axis Accelerometer Breakout - ADXL335 SEN-09269

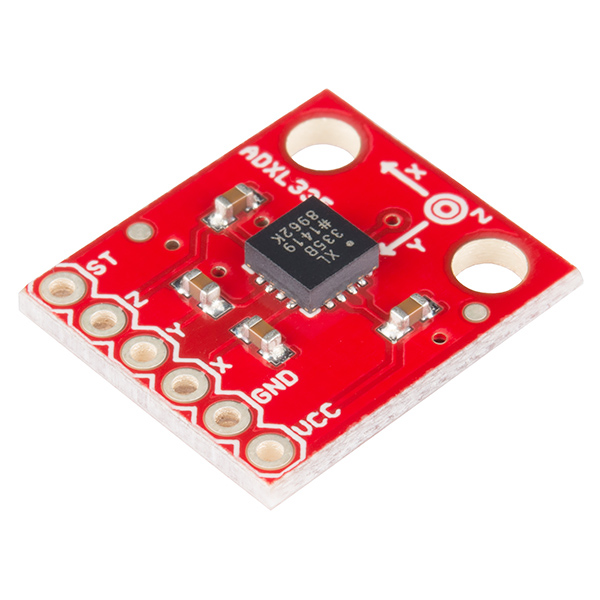
<https://www.sparkfun.com/products/9269>

**Description:** Breakout board for the 3 axis ADXL335 from Analog Devices. This is the latest in a long, proven line of analog sensors - the holy grail of accelerometers. The ADXL335 is a triple axis MEMS accelerometer with extremely low noise and power consumption - only 320uA! The sensor has a full sensing range of +/-3g.

There is no on-board regulation, provided power should be between 1.8 and 3.6VDC.

Board comes fully assembled and tested with external components installed. The included 0.1uF capacitors set the bandwidth of each axis to 50Hz.

Not sure which accelerometer is right for you? Our [Accelerometer and Gyro Buying Guide](https://www.sparkfun.com/pages/accel_gyro_guide) might help!



**Dimensions:** 0.7x0.7"

**Documents:**

* [Schematic](https://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/ADXL335_Breakout.pdf)
* [Eagle Files](https://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/ADXL335_Breakout.zip)
* [Datasheet](http://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf) (ADXL335)
* [Wiring Example](http://wiring.org.co/learning/basics/accelerometer.html)
* [Bildr Example](http://bildr.org/2011/04/sensing-orientation-with-the-adxl335-arduino/)
* [3D cube project](http://www.pyrofersprojects.com/3dcube.php) using a PIC
* [GitHub](https://github.com/sparkfun/ADXL335_Breakout)

14$95

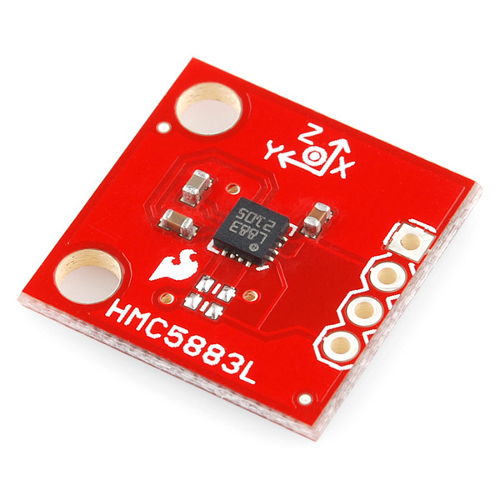
# Tuto shield sparkfun [Triple Axis Magnetometer - HMC5883L Breakout Quickstart Guide](https://www.sparkfun.com/tutorials/301)

<https://www.sparkfun.com/tutorials/301>

by [JordanDee](https://www.sparkfun.com/users/206753) | June 30, 2011 | [2 comments](https://www.sparkfun.com/tutorials/301#comments)

## Overview:

This is a [breakout board for Honeywell's HMC5883L](http://www.sparkfun.com/products/10530), a 3-axis magnetometer. [Magnetometers](http://en.wikipedia.org/wiki/Magnetometer) have a wide range of uses. The most common include using the chip as a digital compass to sense direction or using them to detect ferrous (magnetic) metals.

*[](http://dlnmh9ip6v2uc.cloudfront.net/tutorialimages/HMC5883_Landing/5883L_BoB.jpg)*

**Triple Axis Magnetometer Breakout - HMC5883L**

## Requirements:

This breakout board can be hooked up to a number of microcontrollers, as long as they have an I2C interface . However, for this guide you will find the following helpful:

* [Arduino Uno](http://www.sparkfun.com/products/9950)
* [Breadboard](http://www.sparkfun.com/products/9567)
* [soldering iron](http://www.sparkfun.com/products/9507)
* [solder](http://www.sparkfun.com/products/9163)
* [male headers](http://www.sparkfun.com/products/116) or other connection method
* (optional) [needle nose pliers](http://www.sparkfun.com/products/8793)
* (optional) [third hand](http://www.sparkfun.com/products/9317)

## How it Works:

[Magnetic fields](http://en.wikipedia.org/wiki/Magnetic_field) and current go hand-in-hand. When current flows through a wire, a magnetic field is created. This is the basic principle behind electromagnets. This is also the principle used to measure magnetic fields with a magnetometer. The direction of Earth's magnetic fields affects the flow of electrons in the sensor, and those changes in current can be measured and calculated to derive a compass heading or other useful information.

## How to Use it:

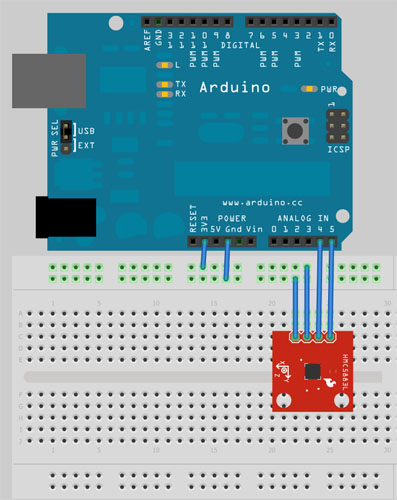
So you're ready to start using this chip with your [Arduino](http://www.sparkfun.com/products/9950) are you? Well lucky for you, we have everything you need to get your project rolling.

### Hardware:

First we'll need to solder on some headers to the breakout board so it will fit into a breadboard. For soldering suggestions, this [tutorial](http://www.sparkfun.com/commerce/tutorial_info.php?tutorials_id=106) is helpful.

The breakout board includes the HMC5883 sensor and all filtering capacitors necessary. There are two unpopulated pads in case you need pull up resistors (no need for them if you're using the Arduino/ATmega328).

Communication with the HMC5883 is simple and all done through an I2C interface. We will go into this in further detail momentarily. All you need to know for now is how to wire it up. Attach the SDA line to A4 and the SCL line to A5. Also attach Vcc to 3.3V and GND to GND.

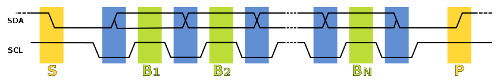
[](http://dlnmh9ip6v2uc.cloudfront.net/tutorialimages/HMC5883_Landing/hmc5883Fritz.jpg)

### Software:

If you haven't done so already, please [download](http://arduino.cc/en/Main/Software) and install the Arduino IDE.

### Firmware:

The HMC5883L speaks a kind of funky serial language called **I2C**. I2C requires just two wires for communication – one for a clock (**SCL**), and one for data (**SDA**). Be aware that you probably should avoid using those two pins for anything else (except for maybe other I2C devices, if you want to get tricky). Don't try using them as analog inputs!

[](http://dlnmh9ip6v2uc.cloudfront.net/tutorialimages/HMC5883_Landing/I2C_data_transfer.png)

The Arduino code example we have, [HMC5883.pde](http://sfecdn.s3.amazonaws.com/datasheets/Sensors/Magneto/HMC5883.pde), will continuously reads data from the magnetometer for each of the magnetometer's axes using I2C. If you need help uploading the sketch to your Arduino, [their site](http://www.arduino.cc/en/Guide/Environment) has lots of helpful information.

Let's take a look at the first part of the code:

#include <Wire.h> //I2C Arduino Library

#define address 0x1E //0011110b, I2C 7bit address of HMC5883

void **setup**(){

  //Initialize Serial and I2C communications

**Serial**.begin(9600);

  Wire.begin();

  //Put the HMC5883 IC into the correct operating mode

  Wire.beginTransmission(address); //open communication with HMC5883

  Wire.send(0x02); //select mode register

  Wire.send(0x00); //continuous measurement mode

  Wire.endTransmission();

}

The code that resides within the *setup()* function is ran once at the beginning. It initializes the serial communication at 9600 baud. We use the serial communication to send data for each axis back to the computer. The I2C is also initialized. We then do a 'write' operation to the HMC5883L. The purpose of this 'write' operation is to adjust the value in the configuration register of the HMC5883L to tell it to be in continuous operation mode. This lets us make continuous reads of the axis data. By default the chip is in single read mode meaning after reading from it once, it will go idle to save power. Once idle, we have write to it to turn it on before we can read from it again. The datasheet has all the information about the registers in addition to other useful information. Of course, we always encourage you to RTFM... just look at the datasheet...

Moving on, here's where we actually request and receive the data:

void **loop**(){

  int x,y,z; //triple axis data

  //Tell the HMC5883L where to begin reading data

  Wire.beginTransmission(address);

  Wire.send(0x03); //select register 3, X MSB register

  Wire.endTransmission();

 //Read data from each axis, 2 registers per axis

  Wire.requestFrom(address, 6);

  if(6<=Wire.available()){

    x = Wire.receive()<<8; //X msb

    x |= Wire.receive(); //X lsb

    z = Wire.receive()<<8; //Z msb

    z |= Wire.receive(); //Z lsb

    y = Wire.receive()<<8; //Y msb

    y |= Wire.receive(); //Y lsb

  }

  //Print out values of each axis

**Serial**.print("x: ");

**Serial**.print(x);

**Serial**.print(" y: ");

**Serial**.print(y);

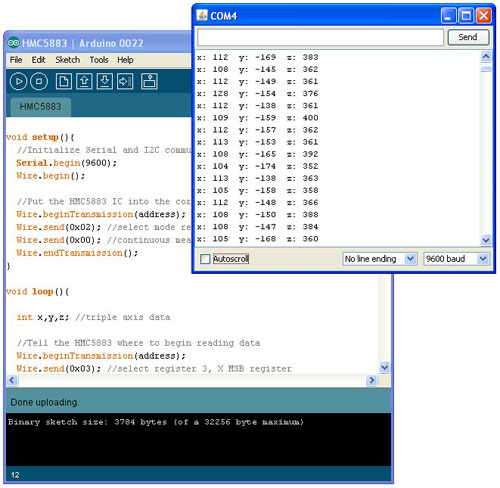
**Serial**.print(" z: ");

**Serial**.println(z);

  delay(250);

}

The *loop()* function will run over and over again as long as the board has power. Each time through, we do a quick write to tell it where we want to begin reading the data for the axes. Register 3 is selected and happens to be the most significant byte of the X axis. We request to read 6 bytes one by one. The chip auto-increments the number of which register we are reading from so we don't have to select each register manually before we read from it. The x, y, and z axis data is then sent back serially to the computer. You can see these values by using the 'Serial Monitor' of the Arduino development environment as shown below.

[](http://sfecdn.s3.amazonaws.com/tutorialimages/HMC5883_Landing/Arduino_HMC5883_xyz.jpg)

All you need now is a creative idea of how to use this information in your next project.  Always feel free to share it with us, it just might end up on the Sparkfun homepage!

## Resources:

* [Schematic](http://sfecdn.s3.amazonaws.com/datasheets/Sensors/Magneto/HMC5883L_Breakout-v11.pdf)
* [HMC5883L Datasheet](http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Magneto/HMC5883L-FDS.pdf)
* [Arduino Code](http://sfecdn.s3.amazonaws.com/datasheets/Sensors/Magneto/HMC5883.pde)
* [Example C code](http://sparkfun.com/datasheets/Sensors/Magneto/hmc5843.zip) (ATmega328)
* [Digital Compass How-To Video](http://www.youtube.com/watch?v=sBKHdt0OdPg)

## Conclusion:

Enjoy your new magnetometer! If you have any problems, feel free to contact SparkFun Technical Support at [techsupport@sparkfun.com](mailto:techsupport@sparkfun.com)

## Comments

* [Member #462421](https://www.sparkfun.com/users/462421) / [about 3 years ago](https://www.sparkfun.com/tutorials/301#comment-520ed3c9ce395fde168b4567) / 1

Sparkfun should also break out the SRDY pin for this chip (making it a 5 pin ‘project’ versus 4.) The SRDY pin does not need to be used, but if it is not made available it cannot ever be used, and it is the means (an interrupt line) by which to read samples from the device much faster than mere serial operations and polling would permit. Such a simple thing, wish they had done it on the first pass.

# Guide gyro de Sparkfun

## Accelerometer, Gyro and IMU Buying Guide

<https://www.sparkfun.com/pages/accel_gyro_guide>

# LCD Button Shield V2 DEV-13293

<https://www.sparkfun.com/products/13293>

**Description:** Do you need an easy to use interface for your LCD screen? The LCD Button Shield V2 attaches to your Arduino to provide a 16x2, black character, green back light LCD with a keypad consisting of 5 keys — select, up, right, down and left. With this shield you will be able to move through menus and make selections straight from one board attached to your Arduino without requiring a massive tower of shields.

The LCD Button Shield V2 works perfectly in 4-bit mode with the “LiquidCrystal” library found in the Arduino IDE, using this library will allow you to control the LCD with only 6 digital I/O lines. Version 2 of this shield provides you with the capability of pushing multiple buttons at once and combining the results. No longer will you be restrained to only 5 inputs, now you have the ability to make use of 32 different button combinations!

Check the LCD Button Shield V2 Wiki link is the Documents section below for example code, schematics, and additional information.

**Note:** Headers are included with this shield but not soldered on.

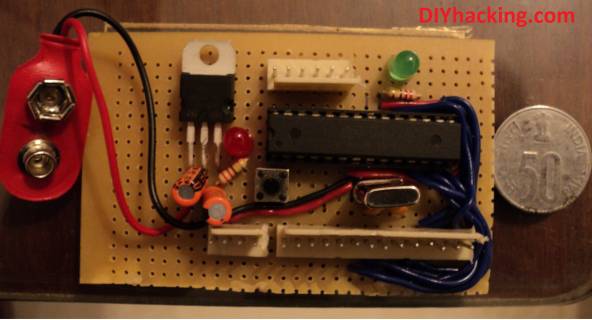
**Documents:**

* [Schematic](https://cdn.sparkfun.com/datasheets/Dev/Arduino/Shields/16X2_LCD_shield.pdf)
* [LCD Button Shield V2 Wiki](http://linksprite.com/wiki/index.php5?title=16_X_2_LCD_Keypad_Shield_for_Arduino_V2)



# How to Make Arduino Board: The Easy Way

<http://diyhacking.com/make-arduino-board-and-bootload/>

[](http://diyhacking.com/wp-content/uploads/2014/05/customardSmall.jpg)

DIY : Make Arduino board and Bootload

Want to save money by making your own arduino clone boards? Or want to make a custom board specifically for your needs, then this DIY project is for you. Make arduino board from cheap electronic components available at your local store. Just follow these simple step by step instructions. DIY : Make arduino board and bootload ATmega chip. Have fun with this DIY hacking tutorial!

**What are the stuff required to do this project?**

Hardware :

1. An arduino bootloader ATmega328/168 or any compatible Atmel IC. Or a blank ATmega328 , ATmega168 , ATmega 8 IC or any other compatible Atmel chips.
2. LM7805 IC.
3. 16 Mhz crystal.
4. Tactile button.
5. Led’s.
6. Capacitors – 22pF (2 Nos), 10uF (2 Nos), 0.1uF (2 Nos).
7. Resistors – 220Ohm (2 Nos), 1k (2 Nos).
8. Pin headers (Male).
9. 28 pin IC socket.
10. Avr pocket programmer (Optional , only if uploading bootloader).

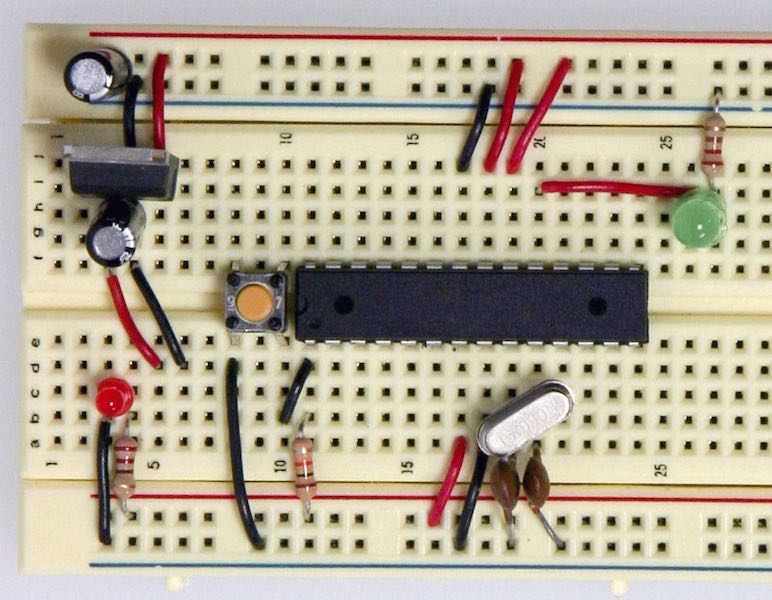
Software :

1. Arduino IDE : [Arduino](http://arduino.cc/en/main/software).
2. Terminal for linux users and Command prompt for windows.
3. Avrdude.

Tools :

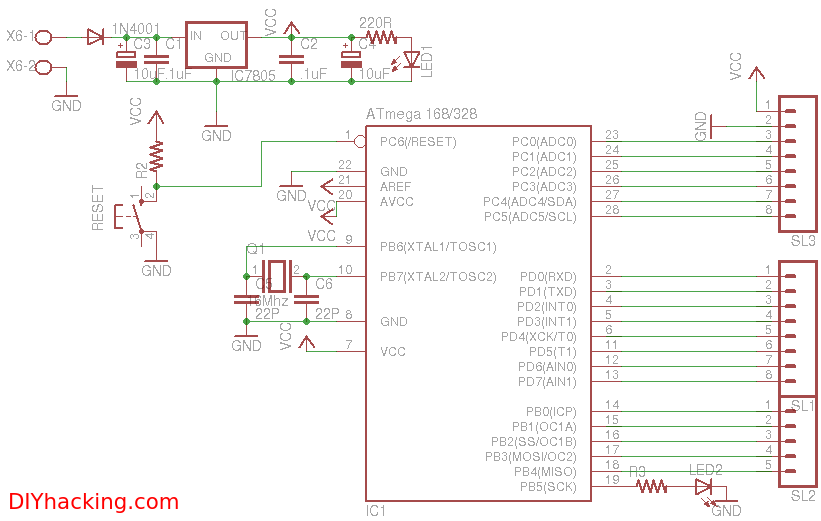
1. Soldering iron.
2. Solder lead.
3. Prototyping solder dot board / breadboard or materials to etch your own board.

#### Step 1 : Connecting the components together to make arduino board

[](http://diyhacking.com/wp-content/uploads/2014/05/arduinobb-min.jpg)

Arduino on breadboard

You can either etch a pcb to make your custom board or use a prototyping dot board to merely solder the components on it. Or even arrange them on a breadboard. Connect the components with each other just like in the diagram below. There are only a very few components required to make your custom arduino board. A voltage regulator to supply 5V to arduino. it’s supporting circuitry to filter the voltage; then a 16Mhz cystal along with it’s parallely connected 22pF capacitors that acts as the clock to the arduino. And an led connected to pin no 19 of the ATmega , for testing purposes.  Use the schematic if you want to etch your own pcb. Modify the schematic to make arduino board suited for your project. Use this eagle schematic file if you want to make a custom pcb : [DIY Hacking arduino schematic](http://diyhacking.com/projects/arduino.sch.rar).

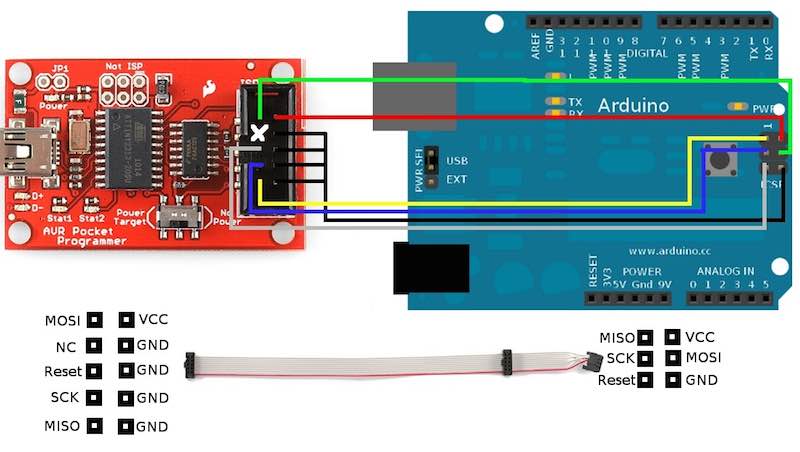
[](http://diyhacking.com/wp-content/uploads/2014/05/arduinosch.png)

Custom arduino schematic: Make arduino board

#### Step 2 : (Optional step) Uploading the bootloader to the blank ATmega chip

This step is for those who have a blank ATmega chip and want to upload the arduino bootloader to it. Others who already have an arduino bootloaded chip can ignore this step and merely insert their chip based on the schematic in the above step. Here you can either use another arduino to bootload the blank chip or use an AVR pocket programmer. I will explain both the methods as follows :

##### Using an AVR pocket programmer :

[](http://diyhacking.com/wp-content/uploads/2014/05/avr-min.jpg)

Arduino and AVR programmer connections

I personally prefer uploading the bootloader using this method as it works most of the time. You just need to buy an AVR pocket programmer. Once you have that, you will be able to burn the bootloader to various types of ATmega chips .Insert the blank chip to the socket of an arduino board.  Connect the programmer to arduino as in the diagram above.

Go to your installed arduino folder -> hardware -> arduino -> boards.txt . Then check the document for your chips name and the bootloader (Duemilanove / Uno) (this step can be ignored if your using  blank ATmega328P chips). Check the values of the following parameters on it : efuse, hfuse and lfuse. Now we will set the fuse bits for the blank chip. Open the command prompt (windows users) or terminal (linux users) and paste the following commands :

NOTE : Substitute the value for efuse, hfuse and lfuse parameters after checking the boards.txt file and edit “m328p” in the below command to your chips name only if it is not an ATmega328 IC.

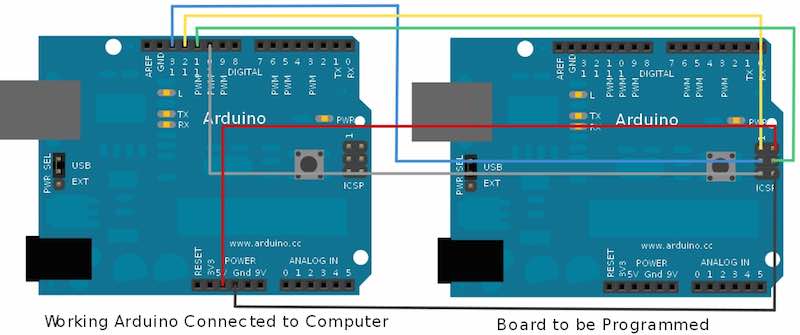
avrdude -b 19200 -c usbtiny -p m328p -v -e -U efuse:w:0x05:m -U hfuse:w:0xD6:m -U lfuse:w:0xFF:m

Next, use the command below to upload the bootloader or your hex program file. Go to the path of your program file or bootloader (/hardware/arduino/bootloaders/atmega) from the command prompt or terminal and instead of “hexfilename.hex” substitute the name of your file.

avrdude -b19200 -c usbtiny -p m328p -v -e -U flash:w:hexfilename.hex -U lock:w:0x0F:m

Now the file will be successfully uploaded to the ATmega chip.

##### Using another arduino as ISP :

[](http://diyhacking.com/wp-content/uploads/2014/05/arduinoboot-min.jpg)

Arduino as ISP

If your using another arduino to bootload the chip then make connections as per the diagram above. You will also need two arduino boards for this : one will be a working arduino board and in the other one the blank ATmega chip needs to be inserted. We will be using the ICSP (In Circuit Serial Programmer) pins of the arduino for this. After making the connections shown connect the working arduino to your PC. Next, open the arduino IDE and go to the tools menu; select burn bootloader and from it’s options click on w/ Arduino as ISP.  Now the chip will be bootloaded with arduino. This method works only if you have the correct chip specified in the boards.txt file and works only rarely. Hence I always prefer the first method.

#### Step 3 : Uploading the code and using the board

Now you have an arduino board with an arduino bootloader ATmega chip. Now use the Rx, Tx and Gnd pins to connect to a USB-UART TTL converter which can be used to connect to your PC. Or just insert a chip that has been uploaded with the program (from an arduino board) into this board’s socket. DIY: Make arduino board and bootload ATmega chip.

# MPU-9250 and Arduino (9-Axis IMU)

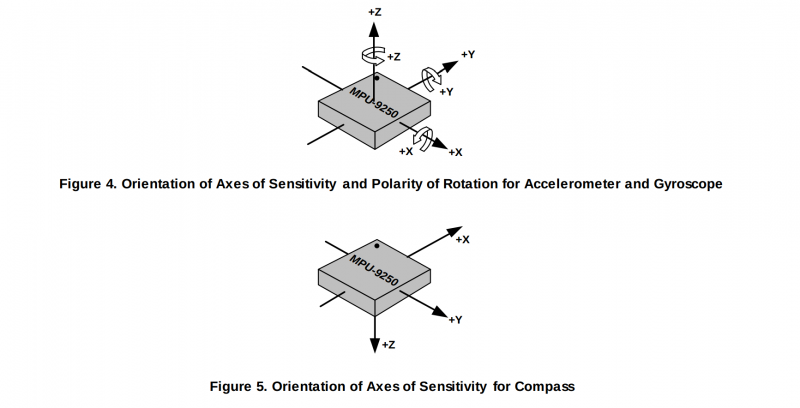
<http://www.lucidarme.me/?p=5057>

[January 26, 2015](http://www.lucidarme.me/?p=5057)

This post presents a simple example of how to interface the MPU-9250 with an Arduino board. The breakout board used here is the [IMU 9DOF - MPU9250 breakout board](http://www.drotek.fr/shop/en/home/421-mpu9250-gyro-accelerometer-magnetometer.html) manufactured by Drotek.

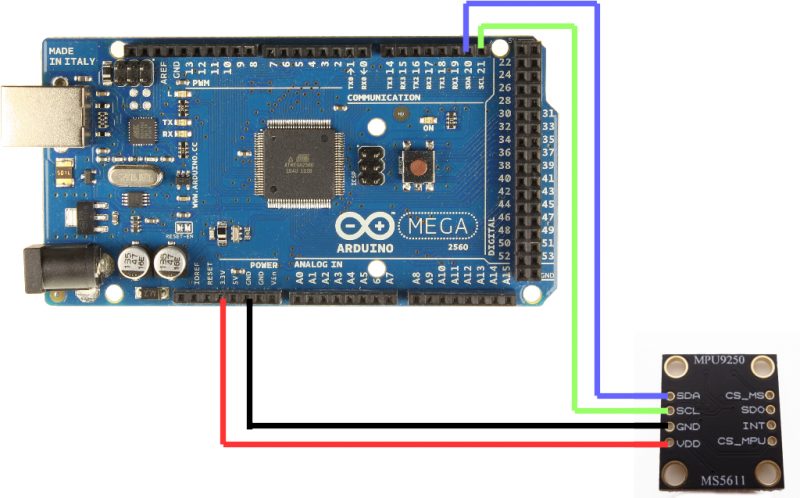
The MPU-9250 is a multi-chip module (MCM) consisting of:

* 3-Axis accelerometer
* 3-Axis gyroscope
* 3-Axis magnetometer

[](http://www.lucidarme.me/wp-content/uploads/2015/01/AxisOrientation.png)

## Wiring

### Arduino mega

[](http://www.lucidarme.me/wp-content/uploads/2015/01/wiring.png)

### Source code

#include

#define MPU9250\_ADDRESS 0x68

#define MAG\_ADDRESS 0x0C

#define GYRO\_FULL\_SCALE\_250\_DPS 0x00

#define GYRO\_FULL\_SCALE\_500\_DPS 0x08

#define GYRO\_FULL\_SCALE\_1000\_DPS 0x10

#define GYRO\_FULL\_SCALE\_2000\_DPS 0x18

#define ACC\_FULL\_SCALE\_2\_G 0x00

#define ACC\_FULL\_SCALE\_4\_G 0x08

#define ACC\_FULL\_SCALE\_8\_G 0x10

#define ACC\_FULL\_SCALE\_16\_G 0x18

// This function read Nbytes bytes from I2C device at address Address.

// Put read bytes starting at register Register in the Data array.

void I2Cread(uint8\_t Address, uint8\_t Register, uint8\_t Nbytes, uint8\_t\* Data)

{

// Set register address

Wire.beginTransmission(Address);

Wire.write(Register);

Wire.endTransmission();

// Read Nbytes

Wire.requestFrom(Address, Nbytes);

uint8\_t index=0;

while (Wire.available())

Data[index++]=Wire.read();

}

// Write a byte (Data) in device (Address) at register (Register)

void I2CwriteByte(uint8\_t Address, uint8\_t Register, uint8\_t Data)

{

// Set register address

Wire.beginTransmission(Address);

Wire.write(Register);

Wire.write(Data);

Wire.endTransmission();

}

// Initializations

void setup()

{

// Arduino initializations

Wire.begin();

Serial.begin(115200);

// Configure gyroscope range

I2CwriteByte(MPU9250\_ADDRESS,27,GYRO\_FULL\_SCALE\_2000\_DPS);

// Configure accelerometers range

I2CwriteByte(MPU9250\_ADDRESS,28,ACC\_FULL\_SCALE\_16\_G);

// Set by pass mode for the magnetometers

I2CwriteByte(MPU9250\_ADDRESS,0x37,0x02);

// Request first magnetometer single measurement

I2CwriteByte(MAG\_ADDRESS,0x0A,0x01);

}

long int cpt=0;

// Main loop, read and display data

void loop()

{

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// ::: Counter :::

// Display data counter

Serial.print (cpt++,DEC);

Serial.print ("\t");

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// ::: accelerometer and gyroscope :::

// Read accelerometer and gyroscope

uint8\_t Buf[14];

I2Cread(MPU9250\_ADDRESS,0x3B,14,Buf);

// Create 16 bits values from 8 bits data

// Accelerometer

int16\_t ax=-(Buf[0]<<8 | Buf[1]);

int16\_t ay=-(Buf[2]<<8 | Buf[3]);

int16\_t az=Buf[4]<<8 | Buf[5];

// Gyroscope

int16\_t gx=-(Buf[8]<<8 | Buf[9]);

int16\_t gy=-(Buf[10]<<8 | Buf[11]);

int16\_t gz=Buf[12]<<8 | Buf[13];

// Display values

// Accelerometer

Serial.print (ax,DEC);

Serial.print ("\t");

Serial.print (ay,DEC);

Serial.print ("\t");

Serial.print (az,DEC);

Serial.print ("\t");

// Gyroscope

Serial.print (gx,DEC);

Serial.print ("\t");

Serial.print (gy,DEC);

Serial.print ("\t");

Serial.print (gz,DEC);

Serial.print ("\t");

// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// ::: Magnetometer :::

// Read register Status 1 and wait for the DRDY: Data Ready

uint8\_t ST1;

do

{

I2Cread(MAG\_ADDRESS,0x02,1,&ST1);

}

while (!(ST1&0x01));

// Read magnetometer data

uint8\_t Mag[7];

I2Cread(MAG\_ADDRESS,0x03,7,Mag);

// Create 16 bits values from 8 bits data

// Magnetometer

int16\_t mx=-(Mag[3]<<8 | Mag[2]);

int16\_t my=-(Mag[1]<<8 | Mag[0]);

int16\_t mz=-(Mag[5]<<8 | Mag[4]);

// Magnetometer

Serial.print (mx+200,DEC);

Serial.print ("\t");

Serial.print (my-70,DEC);

Serial.print ("\t");

Serial.print (mz-700,DEC);

Serial.print ("\t");

// End of line

Serial.println("");

// delay(100);

}

Download

**mpu-9250 - Source code for Arduino** 4.08 KB

[**mpu-9250 - Source code for Arduino**](http://www.lucidarme.me/?p=5057)

**mpu-9250 - OpenGL app** 52.08 KB

[**mpu-9250 - OpenGl app**](http://www.lucidarme.me/?p=5057)

**mpu9250 OpenGL app for Window** 50.61 KB

[**mpu9250\_OpenGL.zip**](http://www.lucidarme.me/?p=5057)

# SparkFun IMU Breakout - MPU-9250 SEN-13762

<https://www.sparkfun.com/products/13762>

**Description:** The SparkFun MPU-9250 IMU Breakout features the latest 9-axis MEMS sensor from InvenSense. Each of these 9DoF breakouts feature an MPU-9250 with a System in Package (SiP) that combines two chips: the MPU-6500, which contains a 3-axis gyroscope as well as a 3-axis accelerometer, and the AK8963, which features a 3-axis magnetometer. This breakout has been designed to be smaller than some of our other offerings to fit in smaller projects. However, if you plan to use a breadboard, or secure the IMU board to a project with something like epoxy, the mounting holes can be easily snapped off.

To achieve its smaller size, the MPU-9250 Breakout features PTH pins that have been wrapped around the border of the PCB in three rows of three or four. The top row (J1) is all one needs to get the most functionality out of the IMU. These include the I2C and power interface. The second most likely to be used set of PTHs are found along the bottom (J3). This includes the address pin, the interrupt pin, and the IO voltage supply for easy interface with a more modern 1.8V processor. The third, a non-breadboard-compatible row (J2), is used for features like running other I2C devices as slaves to this one. For prototyping with these connections, throw your connections on top as you would with an [Arduino Pro Mini](https://www.sparkfun.com/products/11114) or similar product.

The MPU-9250 replaces the popular EOL MPU-9150 and decreases power consumption by 44 percent. According to InvenSense, “Gyro noise performance is 3x better, and compass full-scale range is over 4x better than competitive offerings.” The MPU-9250 uses 16-bit Analog-to-Digital Converters (ADCs) for digitizing all nine axes, making it a very stable 9 Degrees of Freedom board.

<https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide?_ga=1.163362933.855947421.1486021700>

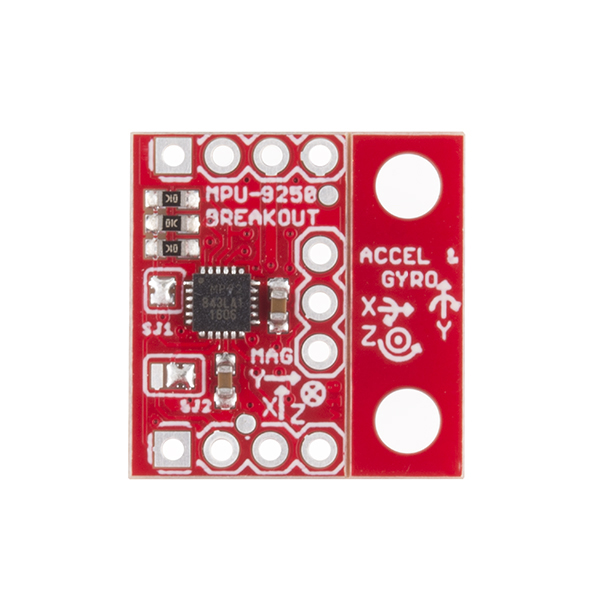
<https://youtu.be/UW9RyHPSPF8>

**Features:**

* Digital-output X-, Y-, and Z-axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1,000 and ±2,000°/sec and integrated 16-bit ADCs
* Digital-output triple-axis accelerometer with a programmable full-scale range of ±2g, ±4g, ±8g and ±16g and integrated 16-bit ADCs
* 3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator
* Digitally programmable low-pass Gyroscope filter
* Gyroscope operating current: 3.2mA
* Accelerometer normal operating current: 450µA
* Magnetometer normal operating current: 280µA at 8Hz repetition rate
* VDD supply voltage range of 2.4 – 3.6V
* Small board design
* Detachable mounting holes

**Documents:**

* [Schematic](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout.pdf)
* [Eagle Files](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout.zip)
* [Hookup Guide](https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide)
* [Datasheet](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU9250REV1.0.pdf) (MPU-9250)
* [Register Map](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU-9250-Register-Map.pdf) (MPU-9250)
* [Arduino Library](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout_Arduino_Library.zip)
* [GitHub](https://github.com/sparkfun/MPU-9250_Breakout)

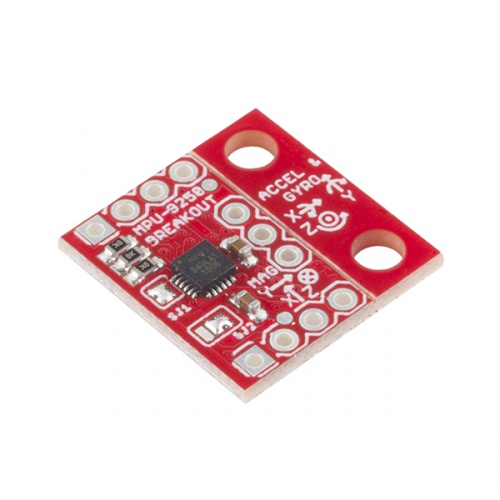


# MPU-9250 Hookup Guide

<https://learn.sparkfun.com/tutorials/mpu-9250-hookup-guide>

## Introduction

The [MPU-9250](https://www.sparkfun.com/products/13762) is the latest 9-axis MEMS sensor from InvenSense®. This replaces the popular EOL’d [MPU-9150](https://www.sparkfun.com/products/retired/11486). InvenSense® lowered power consumption and decreased the size by 44% compared to the MPU-9150. “Gyro noise performance is 3x better, and compass full scale range is over 4x better than competitive offerings.” The MPU-9250 uses 16-bit analog-to-digital converters (ADCs) for digitizing all 9 axes.

[](https://www.sparkfun.com/products/13762)

### [SparkFun IMU Breakout - MPU-9250](https://www.sparkfun.com/products/13762)

[In stock](https://www.sparkfun.com/static/bubbles/) SEN-13762

$14.95

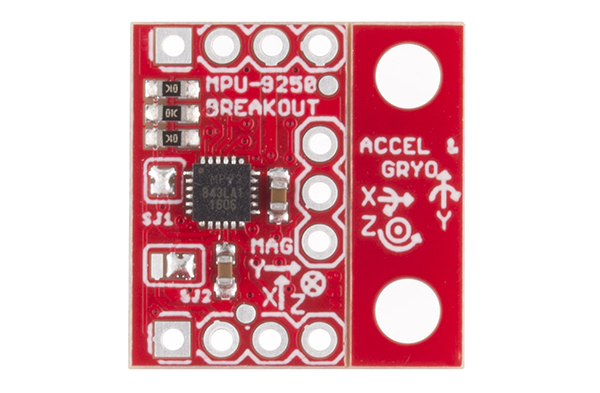
The **S**ystem **i**n **P**ackage (SiP) combines two chips: the MPU-6500, which contains a 3-axis gyroscope, a 3-axis accelerometer, and the AK8963, a 3-axis magnetometer.

### Suggested Reading

Before getting started, you may find the following links useful:

* [I2C Protocol](https://learn.sparkfun.com/tutorials/i2c)
* [Logic Levels](https://learn.sparkfun.com/tutorials/logic-levels)
* [Installing an Arduino Library](https://learn.sparkfun.com/tutorials/installing-an-arduino-library)
* [What are Pull-up Resistors?](https://learn.sparkfun.com/tutorials/pull-up-resistors)
* [How to use a Breadbaord](https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard)

## Board Overview

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/13762-01.jpg)

The MPU-9250 Breakout as you will receive it

The board is designed to be smaller than some of our other offerings to fit in smaller projects. To achieve this, the PTHs are wrapped around the boarder of the PCB in three rows of three or four. The top row ([J1](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout.pdf)) is all one need to get most of the functionality of the IMU. These include the I2C and power interface. If space were really tight, one could take a saw and carefully remove all of the other PTHs.

The second most likely to be used set of PTHs are found along the bottom ([J3](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout.pdf)). This includes the address pin, the interrupt pin, and the IO voltage supply for easy interface with a more modern 1.8V processor.

The third, non-breadboard-compatible row ([J2](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout.pdf)) is used for features like running other I2C devices as slaves to this one. For prototyping with these connections, throw your connections on top like you would with an [Arduino Pro Mini](https://www.sparkfun.com/products/11114) or similar product.

### PTH Connections

The following table summarizes all of the plated through hole (PTH) connections on the breakout board in order found on the board stating in the upper-left corner and wrapping clockwise:

| **Pin Label** | **Pin Function** | **Notes** |
| --- | --- | --- |
| **SCL** | I2C serial clock SPI serial port clock | 100 or 400 kHz I2C Up to 1 MHz SPI (20 MHz in certain cases) |
| **SDA** | I2C serial data | Can also be used for SPI serial data input (SDI) |
| **VDD** | Power supply | +2.4V to +3.6V |
| **GND** | Ground reference | +0V |
| **AUXDA** | Ground reference | I2C master serial data, for connecting to external sensors |
| **FSYNC** | Ground reference | Frame synchronization digital input. Connect to GND if unused. |
| **AUXCL** | Ground reference | I2C Master serial clock, for connecting to external sensors |
| **INT** | Interrupt signal | Interrupt digital output (totem pole or open-drain) |
| **CS** | Chip select | Chip select (SPI mode only) |
| **AD0/ SDO** | Address selection | I2C Slave Address LSB (AD0):    Low: 0b1101000 ➫ 0x68    High: 0b1101001 ➫ 0x69 SPI serial data output (SDO) |
| **VDDIO** | Power supply for I/O pins | +1.71V up to VDD |

### Jumpers

The MPU-9250 Breakout has two solder jumpers, [SJ1](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout.pdf) and [SJ2](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout.pdf).

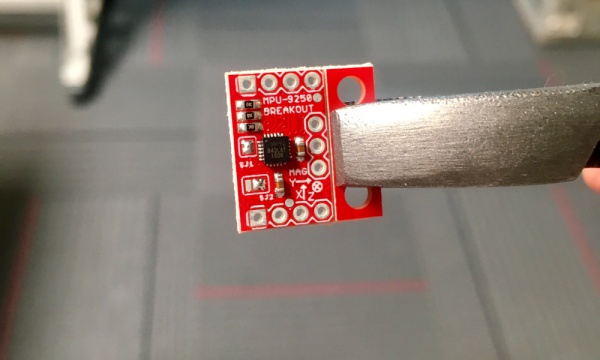
SJ1 comes pre-soldered to short VDD and VDDIO. This reduces the number of power supplies to one with out requiring an external jumper. If the core and IO need to be supplied with different voltages, remove the solder from SJ1.

SJ2 is a two way jumper that comes pre-soldered to connect AD0 to ground. This sets the I2C address to 0x68. It also leaves the PTH for AD0 disconnected and floating. If the solder is moved to connect the center pad with the pad on the left, then the AD0 PTH needs to be connected high or low to chose the I2C address.

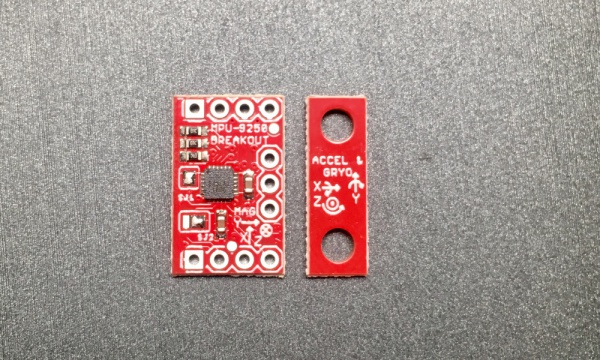
### Reducing size

As stated earlier, one of the design goals for this breakout was to make the board small. Some projects will require mounting holes, so we threw them on the right side of some v-score on this board. Since the board is only ⅔“ wide and there isn’t enough mass to the left of the mounting holes, there isn’t much of a bending moment.

If you plan to use a breadboard, or secure the IMU securely to a project with something like epoxy, the mounting holes can be snapped off. As shown in the following image. The pliers I had on hand made super easy work of this. The edge of a table should work fine too.

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/breakingPCB.jpg)

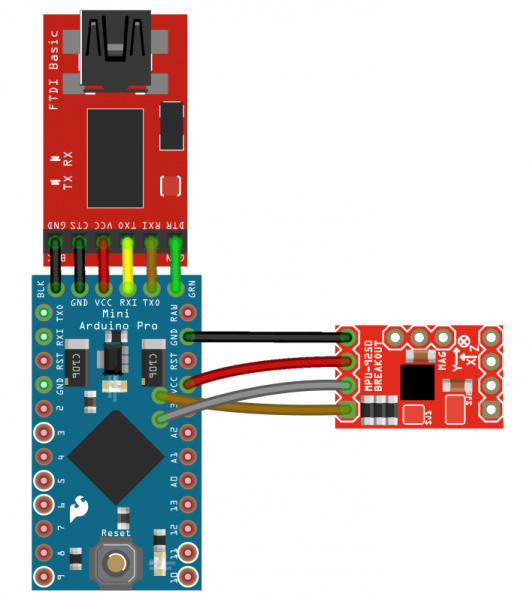
Board was held with pliers and easily broke hand pressing the other side

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/splitPCB.jpg)

PCB snapped in two parts

## Hardware Connections

The MPU-9250 breakout board runs on 3.3 VDC, so a 3.3V USB to UART bridge such as the [SparkFun FTDI Basic Breakout - 3.3V](https://www.sparkfun.com/products/9873) or the [SparkFun Beefy 3 - FTDI Basic Breakout](https://www.sparkfun.com/products/13746) can be used to power and bridge communication with a micro controller. In this case an [Arduino Pro Mini 328 - 3.3V/8MHz](https://www.sparkfun.com/products/11114) was chosen so logic level translation isn’t needed.

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/fritzing_1.jpg)

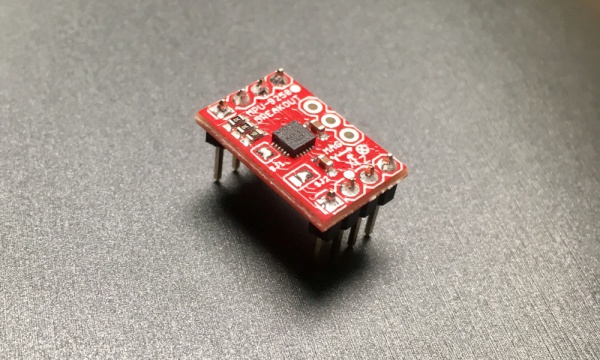
Fritzing diagram of setup

Only 4 connections are needed for I2C communication.

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/PCBPreSolder.jpg)

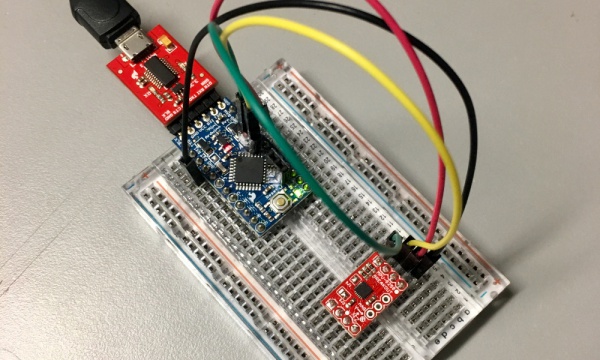
Minimum parts for a breadboard compatible setup

For stability in the breadboard, another four pins were soldered on: VDDIO, AD0/SDO, CS, and INT.

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/solderedPCB.jpg)

PCB with breadboard compatible male headers soldered on

Here is the final setup used for testing.

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/inBreadboard.jpg)

Setup used for testing

## Library and Example Code

The example sketch and library are HEAVILY based on the work of Kris Winer. His original work can be found [here](https://github.com/kriswiner/MPU-9250) on GitHub. Our version is mostly a conversion to make it follow the Arduino library format. [This](http://www.x-io.co.uk/open-source-imu-and-ahrs-algorithms/) is a link to information on the algorithms Kris used for the **a**ttitude and **h**eading **r**eference **s**ystem (AHRS).

To install our library manually grab a copy [here](https://github.com/sparkfun/MPU-9250_Breakout_Arduino_Library), or just use the library manager as detailed in [this tutorial](https://learn.sparkfun.com/tutorials/installing-an-arduino-library).

/\* MPU9250 Basic Example Code

by: Kris Winer

date: April 1, 2014

license: Beerware - Use this code however you'd like. If you

find it useful you can buy me a beer some time.

Modified by Brent Wilkins July 19, 2016

Demonstrate basic MPU-9250 functionality including parameterizing the register

addresses, initializing the sensor, getting properly scaled accelerometer,

gyroscope, and magnetometer data out. Added display functions to allow display

to on breadboard monitor. Addition of 9 DoF sensor fusion using open source

Madgwick and Mahony filter algorithms. Sketch runs on the 3.3 V 8 MHz Pro Mini

and the Teensy 3.1.

SDA and SCL should have external pull-up resistors (to 3.3V).

10k resistors are on the EMSENSR-9250 breakout board.

Hardware setup:

MPU9250 Breakout --------- Arduino

VDD ---------------------- 3.3V

VDDI --------------------- 3.3V

SDA ----------------------- A4

SCL ----------------------- A5

GND ---------------------- GND

\*/

#include "quaternionFilters.h"

#include "MPU9250.h"

#ifdef LCD

#include <Adafruit\_GFX.h>

#include <Adafruit\_PCD8544.h>

// Using NOKIA 5110 monochrome 84 x 48 pixel display

// pin 9 - Serial clock out (SCLK)

// pin 8 - Serial data out (DIN)

// pin 7 - Data/Command select (D/C)

// pin 5 - LCD chip select (CS)

// pin 6 - LCD reset (RST)

Adafruit\_PCD8544 display = Adafruit\_PCD8544(9, 8, 7, 5, 6);

#endif // LCD

#define AHRS true // Set to false for basic data read

#define SerialDebug true // Set to true to get Serial output for debugging

// Pin definitions

int intPin = 12; // These can be changed, 2 and 3 are the Arduinos ext int pins

int myLed = 13; // Set up pin 13 led for toggling

MPU9250 myIMU;

void setup()

{

Wire.begin();

// TWBR = 12; // 400 kbit/sec I2C speed

Serial.begin(38400);

// Set up the interrupt pin, its set as active high, push-pull

pinMode(intPin, INPUT);

digitalWrite(intPin, LOW);

pinMode(myLed, OUTPUT);

digitalWrite(myLed, HIGH);

#ifdef LCD

display.begin(); // Ini8ialize the display

display.setContrast(58); // Set the contrast

// Start device display with ID of sensor

display.clearDisplay();

display.setTextSize(2);

display.setCursor(0,0); display.print("MPU9250");

display.setTextSize(1);

display.setCursor(0, 20); display.print("9-DOF 16-bit");

display.setCursor(0, 30); display.print("motion sensor");

display.setCursor(20,40); display.print("60 ug LSB");

display.display();

delay(1000);

// Set up for data display

display.setTextSize(1); // Set text size to normal, 2 is twice normal etc.

display.setTextColor(BLACK); // Set pixel color; 1 on the monochrome screen

display.clearDisplay(); // clears the screen and buffer

#endif // LCD

// Read the WHO\_AM\_I register, this is a good test of communication

byte c = myIMU.readByte(MPU9250\_ADDRESS, WHO\_AM\_I\_MPU9250);

Serial.print("MPU9250 "); Serial.print("I AM "); Serial.print(c, HEX);

Serial.print(" I should be "); Serial.println(0x71, HEX);

#ifdef LCD

display.setCursor(20,0); display.print("MPU9250");

display.setCursor(0,10); display.print("I AM");

display.setCursor(0,20); display.print(c, HEX);

display.setCursor(0,30); display.print("I Should Be");

display.setCursor(0,40); display.print(0x71, HEX);

display.display();

delay(1000);

#endif // LCD

if (c == 0x71) // WHO\_AM\_I should always be 0x68

{

Serial.println("MPU9250 is online...");

// Start by performing self test and reporting values

myIMU.MPU9250SelfTest(myIMU.SelfTest);

Serial.print("x-axis self test: acceleration trim within : ");

Serial.print(myIMU.SelfTest[0],1); Serial.println("% of factory value");

Serial.print("y-axis self test: acceleration trim within : ");

Serial.print(myIMU.SelfTest[1],1); Serial.println("% of factory value");

Serial.print("z-axis self test: acceleration trim within : ");

Serial.print(myIMU.SelfTest[2],1); Serial.println("% of factory value");

Serial.print("x-axis self test: gyration trim within : ");

Serial.print(myIMU.SelfTest[3],1); Serial.println("% of factory value");

Serial.print("y-axis self test: gyration trim within : ");

Serial.print(myIMU.SelfTest[4],1); Serial.println("% of factory value");

Serial.print("z-axis self test: gyration trim within : ");

Serial.print(myIMU.SelfTest[5],1); Serial.println("% of factory value");

// Calibrate gyro and accelerometers, load biases in bias registers

myIMU.calibrateMPU9250(myIMU.gyroBias, myIMU.accelBias);

#ifdef LCD

display.clearDisplay();

display.setCursor(0, 0); display.print("MPU9250 bias");

display.setCursor(0, 8); display.print(" x y z ");

display.setCursor(0, 16); display.print((int)(1000\*accelBias[0]));

display.setCursor(24, 16); display.print((int)(1000\*accelBias[1]));

display.setCursor(48, 16); display.print((int)(1000\*accelBias[2]));

display.setCursor(72, 16); display.print("mg");

display.setCursor(0, 24); display.print(myIMU.gyroBias[0], 1);

display.setCursor(24, 24); display.print(myIMU.gyroBias[1], 1);

display.setCursor(48, 24); display.print(myIMU.gyroBias[2], 1);

display.setCursor(66, 24); display.print("o/s");

display.display();

delay(1000);

#endif // LCD

myIMU.initMPU9250();

// Initialize device for active mode read of acclerometer, gyroscope, and

// temperature

Serial.println("MPU9250 initialized for active data mode....");

// Read the WHO\_AM\_I register of the magnetometer, this is a good test of

// communication

byte d = myIMU.readByte(AK8963\_ADDRESS, WHO\_AM\_I\_AK8963);

Serial.print("AK8963 "); Serial.print("I AM "); Serial.print(d, HEX);

Serial.print(" I should be "); Serial.println(0x48, HEX);

#ifdef LCD

display.clearDisplay();

display.setCursor(20,0); display.print("AK8963");

display.setCursor(0,10); display.print("I AM");

display.setCursor(0,20); display.print(d, HEX);

display.setCursor(0,30); display.print("I Should Be");

display.setCursor(0,40); display.print(0x48, HEX);

display.display();

delay(1000);

#endif // LCD

// Get magnetometer calibration from AK8963 ROM

myIMU.initAK8963(myIMU.magCalibration);

// Initialize device for active mode read of magnetometer

Serial.println("AK8963 initialized for active data mode....");

if (SerialDebug)

{

// Serial.println("Calibration values: ");

Serial.print("X-Axis sensitivity adjustment value ");

Serial.println(myIMU.magCalibration[0], 2);

Serial.print("Y-Axis sensitivity adjustment value ");

Serial.println(myIMU.magCalibration[1], 2);

Serial.print("Z-Axis sensitivity adjustment value ");

Serial.println(myIMU.magCalibration[2], 2);

}

#ifdef LCD

display.clearDisplay();

display.setCursor(20,0); display.print("AK8963");

display.setCursor(0,10); display.print("ASAX "); display.setCursor(50,10);

display.print(myIMU.magCalibration[0], 2);

display.setCursor(0,20); display.print("ASAY "); display.setCursor(50,20);

display.print(myIMU.magCalibration[1], 2);

display.setCursor(0,30); display.print("ASAZ "); display.setCursor(50,30);

display.print(myIMU.magCalibration[2], 2);

display.display();

delay(1000);

#endif // LCD

} // if (c == 0x71)

else

{

Serial.print("Could not connect to MPU9250: 0x");

Serial.println(c, HEX);

while(1) ; // Loop forever if communication doesn't happen

}

}

void loop()

{

// If intPin goes high, all data registers have new data

// On interrupt, check if data ready interrupt

if (myIMU.readByte(MPU9250\_ADDRESS, INT\_STATUS) & 0x01)

{

myIMU.readAccelData(myIMU.accelCount); // Read the x/y/z adc values

myIMU.getAres();

// Now we'll calculate the accleration value into actual g's

// This depends on scale being set

myIMU.ax = (float)myIMU.accelCount[0]\*myIMU.aRes; // - accelBias[0];

myIMU.ay = (float)myIMU.accelCount[1]\*myIMU.aRes; // - accelBias[1];

myIMU.az = (float)myIMU.accelCount[2]\*myIMU.aRes; // - accelBias[2];

myIMU.readGyroData(myIMU.gyroCount); // Read the x/y/z adc values

myIMU.getGres();

// Calculate the gyro value into actual degrees per second

// This depends on scale being set

myIMU.gx = (float)myIMU.gyroCount[0]\*myIMU.gRes;

myIMU.gy = (float)myIMU.gyroCount[1]\*myIMU.gRes;

myIMU.gz = (float)myIMU.gyroCount[2]\*myIMU.gRes;

myIMU.readMagData(myIMU.magCount); // Read the x/y/z adc values

myIMU.getMres();

// User environmental x-axis correction in milliGauss, should be

// automatically calculated

myIMU.magbias[0] = +470.;

// User environmental x-axis correction in milliGauss TODO axis??

myIMU.magbias[1] = +120.;

// User environmental x-axis correction in milliGauss

myIMU.magbias[2] = +125.;

// Calculate the magnetometer values in milliGauss

// Include factory calibration per data sheet and user environmental

// corrections

// Get actual magnetometer value, this depends on scale being set

myIMU.mx = (float)myIMU.magCount[0]\*myIMU.mRes\*myIMU.magCalibration[0] -

myIMU.magbias[0];

myIMU.my = (float)myIMU.magCount[1]\*myIMU.mRes\*myIMU.magCalibration[1] -

myIMU.magbias[1];

myIMU.mz = (float)myIMU.magCount[2]\*myIMU.mRes\*myIMU.magCalibration[2] -

myIMU.magbias[2];

} // if (readByte(MPU9250\_ADDRESS, INT\_STATUS) & 0x01)

// Must be called before updating quaternions!

myIMU.updateTime();

// Sensors x (y)-axis of the accelerometer is aligned with the y (x)-axis of

// the magnetometer; the magnetometer z-axis (+ down) is opposite to z-axis

// (+ up) of accelerometer and gyro! We have to make some allowance for this

// orientationmismatch in feeding the output to the quaternion filter. For the

// MPU-9250, we have chosen a magnetic rotation that keeps the sensor forward

// along the x-axis just like in the LSM9DS0 sensor. This rotation can be

// modified to allow any convenient orientation convention. This is ok by

// aircraft orientation standards! Pass gyro rate as rad/s

// MadgwickQuaternionUpdate(ax, ay, az, gx\*PI/180.0f, gy\*PI/180.0f, gz\*PI/180.0f, my, mx, mz);

MahonyQuaternionUpdate(myIMU.ax, myIMU.ay, myIMU.az, myIMU.gx\*DEG\_TO\_RAD,

myIMU.gy\*DEG\_TO\_RAD, myIMU.gz\*DEG\_TO\_RAD, myIMU.my,

myIMU.mx, myIMU.mz, myIMU.deltat);

if (!AHRS)

{

myIMU.delt\_t = millis() - myIMU.count;

if (myIMU.delt\_t > 500)

{

if(SerialDebug)

{

// Print acceleration values in milligs!

Serial.print("X-acceleration: "); Serial.print(1000\*myIMU.ax);

Serial.print(" mg ");

Serial.print("Y-acceleration: "); Serial.print(1000\*myIMU.ay);

Serial.print(" mg ");

Serial.print("Z-acceleration: "); Serial.print(1000\*myIMU.az);

Serial.println(" mg ");

// Print gyro values in degree/sec

Serial.print("X-gyro rate: "); Serial.print(myIMU.gx, 3);

Serial.print(" degrees/sec ");

Serial.print("Y-gyro rate: "); Serial.print(myIMU.gy, 3);

Serial.print(" degrees/sec ");

Serial.print("Z-gyro rate: "); Serial.print(myIMU.gz, 3);

Serial.println(" degrees/sec");

// Print mag values in degree/sec

Serial.print("X-mag field: "); Serial.print(myIMU.mx);

Serial.print(" mG ");

Serial.print("Y-mag field: "); Serial.print(myIMU.my);

Serial.print(" mG ");

Serial.print("Z-mag field: "); Serial.print(myIMU.mz);

Serial.println(" mG");

myIMU.tempCount = myIMU.readTempData(); // Read the adc values

// Temperature in degrees Centigrade

myIMU.temperature = ((float) myIMU.tempCount) / 333.87 + 21.0;

// Print temperature in degrees Centigrade

Serial.print("Temperature is "); Serial.print(myIMU.temperature, 1);

Serial.println(" degrees C");

}

#ifdef LCD

display.clearDisplay();

display.setCursor(0, 0); display.print("MPU9250/AK8963");

display.setCursor(0, 8); display.print(" x y z ");

display.setCursor(0, 16); display.print((int)(1000\*myIMU.ax));

display.setCursor(24, 16); display.print((int)(1000\*myIMU.ay));

display.setCursor(48, 16); display.print((int)(1000\*myIMU.az));

display.setCursor(72, 16); display.print("mg");

display.setCursor(0, 24); display.print((int)(myIMU.gx));

display.setCursor(24, 24); display.print((int)(myIMU.gy));

display.setCursor(48, 24); display.print((int)(myIMU.gz));

display.setCursor(66, 24); display.print("o/s");

display.setCursor(0, 32); display.print((int)(myIMU.mx));

display.setCursor(24, 32); display.print((int)(myIMU.my));

display.setCursor(48, 32); display.print((int)(myIMU.mz));

display.setCursor(72, 32); display.print("mG");

display.setCursor(0, 40); display.print("Gyro T ");

display.setCursor(50, 40); display.print(myIMU.temperature, 1);

display.print(" C");

display.display();

#endif // LCD

myIMU.count = millis();

digitalWrite(myLed, !digitalRead(myLed)); // toggle led

} // if (myIMU.delt\_t > 500)

} // if (!AHRS)

else

{

// Serial print and/or display at 0.5 s rate independent of data rates

myIMU.delt\_t = millis() - myIMU.count;

// update LCD once per half-second independent of read rate

if (myIMU.delt\_t > 500)

{

if(SerialDebug)

{

Serial.print("ax = "); Serial.print((int)1000\*myIMU.ax);

Serial.print(" ay = "); Serial.print((int)1000\*myIMU.ay);

Serial.print(" az = "); Serial.print((int)1000\*myIMU.az);

Serial.println(" mg");

Serial.print("gx = "); Serial.print( myIMU.gx, 2);

Serial.print(" gy = "); Serial.print( myIMU.gy, 2);

Serial.print(" gz = "); Serial.print( myIMU.gz, 2);

Serial.println(" deg/s");

Serial.print("mx = "); Serial.print( (int)myIMU.mx );

Serial.print(" my = "); Serial.print( (int)myIMU.my );

Serial.print(" mz = "); Serial.print( (int)myIMU.mz );

Serial.println(" mG");

Serial.print("q0 = "); Serial.print(\*getQ());

Serial.print(" qx = "); Serial.print(\*(getQ() + 1));

Serial.print(" qy = "); Serial.print(\*(getQ() + 2));

Serial.print(" qz = "); Serial.println(\*(getQ() + 3));

}

// Define output variables from updated quaternion---these are Tait-Bryan

// angles, commonly used in aircraft orientation. In this coordinate system,

// the positive z-axis is down toward Earth. Yaw is the angle between Sensor

// x-axis and Earth magnetic North (or true North if corrected for local

// declination, looking down on the sensor positive yaw is counterclockwise.

// Pitch is angle between sensor x-axis and Earth ground plane, toward the

// Earth is positive, up toward the sky is negative. Roll is angle between

// sensor y-axis and Earth ground plane, y-axis up is positive roll. These

// arise from the definition of the homogeneous rotation matrix constructed

// from quaternions. Tait-Bryan angles as well as Euler angles are

// non-commutative; that is, the get the correct orientation the rotations

// must be applied in the correct order which for this configuration is yaw,

// pitch, and then roll.

// For more see

// http://en.wikipedia.org/wiki/Conversion\_between\_quaternions\_and\_Euler\_angles

// which has additional links.

myIMU.yaw = atan2(2.0f \* (\*(getQ()+1) \* \*(getQ()+2) + \*getQ() \*

\*(getQ()+3)), \*getQ() \* \*getQ() + \*(getQ()+1) \* \*(getQ()+1)

- \*(getQ()+2) \* \*(getQ()+2) - \*(getQ()+3) \* \*(getQ()+3));

myIMU.pitch = -asin(2.0f \* (\*(getQ()+1) \* \*(getQ()+3) - \*getQ() \*

\*(getQ()+2)));

myIMU.roll = atan2(2.0f \* (\*getQ() \* \*(getQ()+1) + \*(getQ()+2) \*

\*(getQ()+3)), \*getQ() \* \*getQ() - \*(getQ()+1) \* \*(getQ()+1)

- \*(getQ()+2) \* \*(getQ()+2) + \*(getQ()+3) \* \*(getQ()+3));

myIMU.pitch \*= RAD\_TO\_DEG;

myIMU.yaw \*= RAD\_TO\_DEG;

// Declination of SparkFun Electronics (40°05'26.6"N 105°11'05.9"W) is

// 8° 30' E ± 0° 21' (or 8.5°) on 2016-07-19

// - http://www.ngdc.noaa.gov/geomag-web/#declination

myIMU.yaw -= 8.5;

myIMU.roll \*= RAD\_TO\_DEG;

if(SerialDebug)

{

Serial.print("Yaw, Pitch, Roll: ");

Serial.print(myIMU.yaw, 2);

Serial.print(", ");

Serial.print(myIMU.pitch, 2);

Serial.print(", ");

Serial.println(myIMU.roll, 2);

Serial.print("rate = ");

Serial.print((float)myIMU.sumCount/myIMU.sum, 2);

Serial.println(" Hz");

}

#ifdef LCD

display.clearDisplay();

display.setCursor(0, 0); display.print(" x y z ");

display.setCursor(0, 8); display.print((int)(1000\*myIMU.ax));

display.setCursor(24, 8); display.print((int)(1000\*myIMU.ay));

display.setCursor(48, 8); display.print((int)(1000\*myIMU.az));

display.setCursor(72, 8); display.print("mg");

display.setCursor(0, 16); display.print((int)(myIMU.gx));

display.setCursor(24, 16); display.print((int)(myIMU.gy));

display.setCursor(48, 16); display.print((int)(myIMU.gz));

display.setCursor(66, 16); display.print("o/s");

display.setCursor(0, 24); display.print((int)(myIMU.mx));

display.setCursor(24, 24); display.print((int)(myIMU.my));

display.setCursor(48, 24); display.print((int)(myIMU.mz));

display.setCursor(72, 24); display.print("mG");

display.setCursor(0, 32); display.print((int)(myIMU.yaw));

display.setCursor(24, 32); display.print((int)(myIMU.pitch));

display.setCursor(48, 32); display.print((int)(myIMU.roll));

display.setCursor(66, 32); display.print("ypr");

// With these settings the filter is updating at a ~145 Hz rate using the

// Madgwick scheme and >200 Hz using the Mahony scheme even though the

// display refreshes at only 2 Hz. The filter update rate is determined

// mostly by the mathematical steps in the respective algorithms, the

// processor speed (8 MHz for the 3.3V Pro Mini), and the magnetometer ODR:

// an ODR of 10 Hz for the magnetometer produce the above rates, maximum

// magnetometer ODR of 100 Hz produces filter update rates of 36 - 145 and

// ~38 Hz for the Madgwick and Mahony schemes, respectively. This is

// presumably because the magnetometer read takes longer than the gyro or

// accelerometer reads. This filter update rate should be fast enough to

// maintain accurate platform orientation for stabilization control of a

// fast-moving robot or quadcopter. Compare to the update rate of 200 Hz

// produced by the on-board Digital Motion Processor of Invensense's MPU6050

// 6 DoF and MPU9150 9DoF sensors. The 3.3 V 8 MHz Pro Mini is doing pretty

// well!

display.setCursor(0, 40); display.print("rt: ");

display.print((float) myIMU.sumCount / myIMU.sum, 2);

display.print(" Hz");

display.display();

#endif // LCD

myIMU.count = millis();

myIMU.sumCount = 0;

myIMU.sum = 0;

} // if (myIMU.delt\_t > 500)

} // if (AHRS)

}

Full demo sketch

Some configuration can be found at the beginning of the sketch. Here is where you can turn on or off AHRS calculations, and serial debugging:

#define AHRS true // Set to false for basic data read

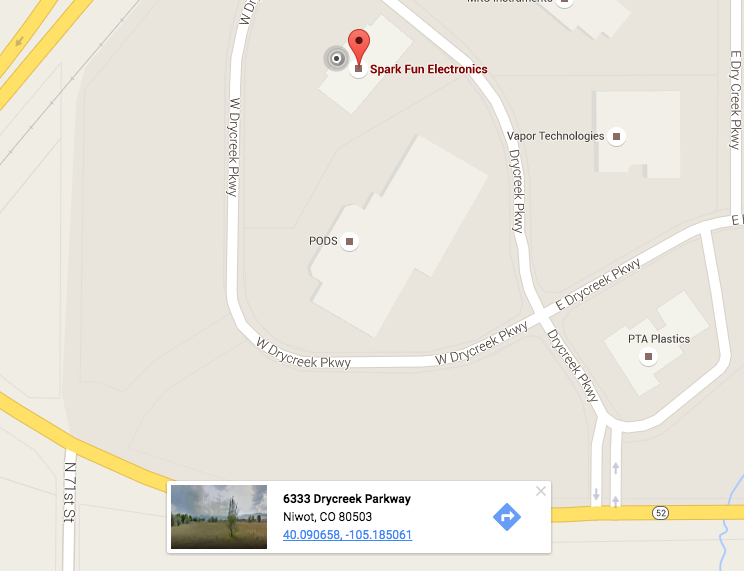
#define SerialDebug true // Set to true to get Serial output for debugging

Some of the settings used by library exposed in the sketch

### Magnetic declination

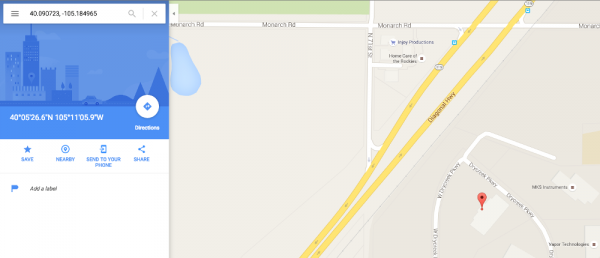
The AHRS needs to know where you are located to convert magnetic north to true north. I used the combination of two services to find the current declination of our offices; Google, and NOAA. This may or may not be ideal in your country.

The first thing I needed were the latitude and longitude of the office. Searching for SparkFun in [Google Maps](https://www.google.com/maps/place/Spark+Fun+Electronics/) shows a red pin on our building. Clicking near the front door, but not on the existing pin dropped a new pin which also made a card appear bottom center. The bottom of this card contains a link to a search of the latitude and longitude marked by the new pin.

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/droppedPin.png)

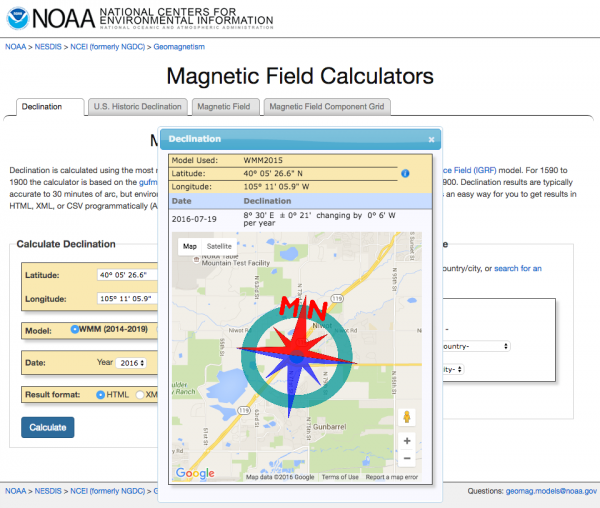
Google Maps showing the latitude and longitude of SparkFun Electronics

Clicking on the link in the card brings up a page showing the latitude and longitude in both the **d**ecimal **d**egrees (DD) and **d**egrees **m**inutes **s**econds (DMS) forms. I find sign errors easier to avoid by using DMS form Google provides with the since I’m not intimately familiar with the WGS 84 **c**oordinate **r**eference **s**ystem (CRS). The included cardinal directions are handy and required by the next tool.

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/Geolocation.png)

Search for SparkFun Electronics' coordinates

The second tool is the default [magnetic field calculator](http://www.ngdc.noaa.gov/geomag-web/#declination) on NOAA’s website. Enter the coordinates found in the previous step into the latitude and longitude inputs. The Calculate button will trigger a dialog box with the results to appear. If the results appear on the Google map where you are expecting them, then you chose the correct directions with the radio inputs!

[](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/Geomagnetism.png)

Magnetic declination of SparkFun Electronics' Colorado office

Note that the declination here is currently 8˚ 30' E. Also note that the image on the map shows magnetic north to be east of true north. The provided DM format needs to be converted to DD format for the code. There are 60 minutes per degree, 30 of them is 30⁄60 of a degree, or 0.5˚. The declination in DD format is thus 8.5˚ E. Update the code in the example sketch around line 391 with the declination for your desired location.

myIMU.yaw -= 8.5;

The library left a lot of the math in the example sketch in part to make things like this easier to access.

Here is an example of what the output of the demo sketch should look like:

MPU9250 I AM 71 I should be 71  
MPU9250 is online...  
x-axis self test: acceleration trim within : 2.4% of factory value  
y-axis self test: acceleration trim within : 0.5% of factory value  
z-axis self test: acceleration trim within : -0.0% of factory value  
x-axis self test: gyration trim within : -0.3% of factory value  
y-axis self test: gyration trim within : -1.0% of factory value  
z-axis self test: gyration trim within : 0.5% of factory value  
MPU9250 initialized for active data mode....  
AK8963 I AM 48 I should be 48  
AK8963 initialized for active data mode....  
X-Axis sensitivity adjustment value 1.21  
Y-Axis sensitivity adjustment value 1.22  
Z-Axis sensitivity adjustment value 1.17  
ax = -70.19 ay = -70.56 az = 931.03 mg  
gx = 0.02 gy = 0.00 gz = 0.02 deg/s  
mx = -189 my = 355 mz = 127 mG  
q0 = 0.97 qx = -0.04 qy = 0.03 qz = 0.22  
Yaw, Pitch, Roll: 16.45, 4.22, -4.14  
rate = 140.15 Hz

## Resources and Going Further

For more information about the MPU-9250 Breakout, check out the links below.

* [MPU-9250 Datasheet](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU9250REV1.0.pdf)
* [MPU-9250 Register Map](https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU-9250-Register-Map.pdf)
* [MPU-9250 Breakout Schematic](https://cdn.sparkfun.com/datasheets/Sensors/IMU/SparkFun_MPU-9250_Breakout.pdf)

Many of the advanced features of the MPU-9250 are only accessable by agreeing to a pages of licensing terms and [logging in as a developer](https://www.invensense.com/developers/software-downloads/) to get access to [Embedded MotionDriver 6.12](http://www.digikey.com/en/pdf/i/invensense/motion-driver-6-1-user-guide). This approach isn’t super Arduino friendly. At power up 3062 bytes of undocumented hex needs to be loaded into the MPU-9250.

#define DMP\_CODE\_SIZE (3062)

static const unsigned char dmp\_memory[DMP\_CODE\_SIZE] = {

/\* bank # 0 \*/

0x00, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00,

0x00, 0x00, 0x65, 0x00, 0x54, 0xff, 0xef, 0x00, 0x00, 0xfa, 0x80, 0x00, 0x0b, 0x12, 0x82,

Code snippet of MPU-9250 Embedded MotionDriver firmware

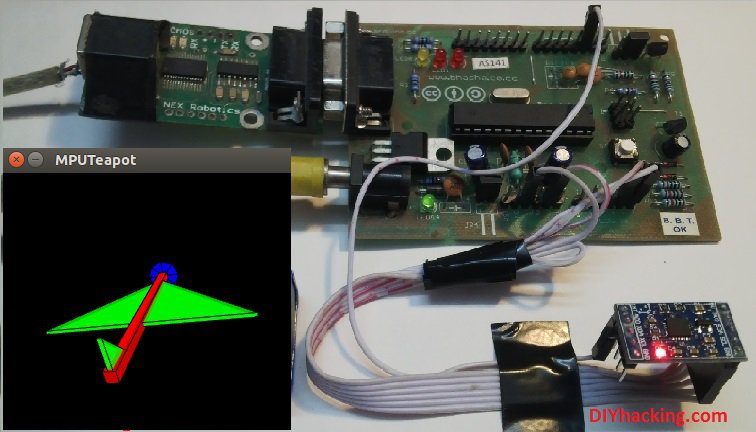
That binary combined with the driver and any code that does something with the sensor data quickly maxes out smaller microcontrollers. Feel free to register and play around if you want to take this product further.

A customer of ours shared his great tutorial on [Affordable 9-DoF Sensor Fusion](https://github.com/kriswiner/MPU-6050/wiki/Affordable-9-DoF-Sensor-Fusion). Check it out for more info on sensor fusion.

# Arduino MPU 6050 – Best IMU Sensor Tutorial

<https://diyhacking.com/arduino-mpu-6050-imu-sensor-tutorial/>

#### From: Arvind Sanjeev, Founder DIY Hacking

[](http://diyhacking.com/wp-content/uploads/2014/11/Newcover.jpg)

Arduino MPU 6050 Tutorial

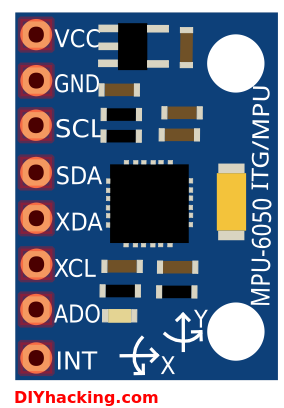
In this post, I will be reviewing a few basic IMU (Inertia Measurement Unit) sensors, compatible with arduino. I shall also give a short tutorial for interfacing arduino with the best IMU sensor available. IMU sensors like the MPU 6050 are used in self balancing robots, UAVs, smartphones, etc.

IMU sensors are one of the most inevitable type of sensors used today in all kinds of electronic gadgets. They are seen in smartphones, wearables, game controllers, etc. IMU sensors help us in getting the attitude of an object, attached to the sensor in three dimensional space. These values usually in angles, thus help us to determine its attitude. Thus, they are used in smartphones to detect its orientation. And also in wearable gadgets like the nike fuel band or fit bit, which use IMU sensors to track movement.

IMU sensors, thus have prolific number of applications. It  is even considered to be an inexorable component in quadrotors. So use this Arduino MPU 6050 tutorial to build a self balancing robot or a hand tracking device . Some of the sensors that I was able to get my hands on were:

1. ADXL 345 accelerometer.
2. ITG 3200 gyroscope.
3. Sparkfun 6 DOF IMU sensor board.
4. MPU 6050.

I was able to work with both accelerometers and gyroscopes separately. However, they are not as accurate as when they are combined.  And among the lot, I found the Invensense MPU 6050 to be the most reliable and accurate IMU sensor. Apart from being significantly cheap from the other sensors, the MPU 6050 performs much better too.

[](http://diyhacking.com/wp-content/uploads/2014/11/mpu6050.png)

Arduino MPU 6050 Pin out

In this tutorial, I shall give you a basic introduction to the MPU 6050, demonstrate how it can be interfaced to an arduino. And show you to make a 3D model using the data from Arduino MPU 6050.

#### What are the stuff required to do this project?

**Hardware:**

1. [Arduino](http://robokits.co.in/shop/index.php?main_page=product_info&cPath=6_72&products_id=388) or an arduino clone board ([freeduino](http://robokits.co.in/shop/index.php?main_page=product_info&cPath=6_72&products_id=388)). Or make your own custom arduino board with this [tutorial.](http://diyhacking.com/make-arduino-board-and-bootload/)
2. [MPU 6050](http://www.ebay.in/itm/like/261646571375) sensor.
3. Interconnecting wires.

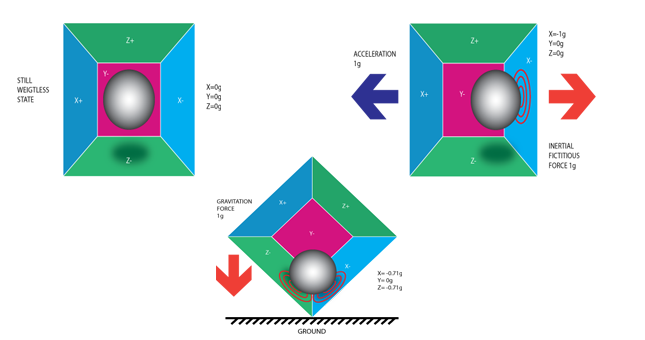
**Software:**

1. Arduino IDE: [Arduino](http://arduino.cc/en/main/software)
2. Processing IDE : [Processing](https://processing.org/download/?processing) (optional)

#### How does it work?

IMU sensors usually consists of two or more parts. Listing them by priority, they are : accelerometer, gyroscope, magnetometer and altimeter. The MPU 6050 is a 6 DOF (Degrees of Freedom) or a six axis IMU sensor, which means that it gives six values as output. Three values from the accelerometer and three from the gyroscope. The MPU 6050 is a sensor based on MEMS (Micro Electro Mechanical Systems) technology. Both the accelerometer and the gyroscope is embedded inside a single chip. This chip uses I2C (Inter Integrated Circuit) protocol for communication.

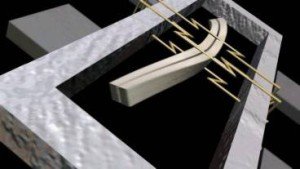
How does an accelerometer work?

[](http://diyhacking.com/wp-content/uploads/2014/11/acc.png)

Piezo Electric Accelerometer

An accelerometer works on the principle of piezo electric effect. Here, imagine a cuboidal box, having a small ball inside it, like in the picture above. The walls of this box are made with piezo electric crystals. Whenever you tilt the box, the ball is forced to move in the direction of the inclination, due to gravity. The wall with which the ball collides, creates tiny piezo electric currents. There are totally, three pairs of opposite walls in a cuboid. Each pair corresponds to an axis in 3D space: X, Y and Z axes. Depending on the current produced from the piezo electric walls, we can determine the direction of inclination and its magnitude. For more information check [this](http://en.wikipedia.org/wiki/Accelerometer).

How does a gyroscope work?

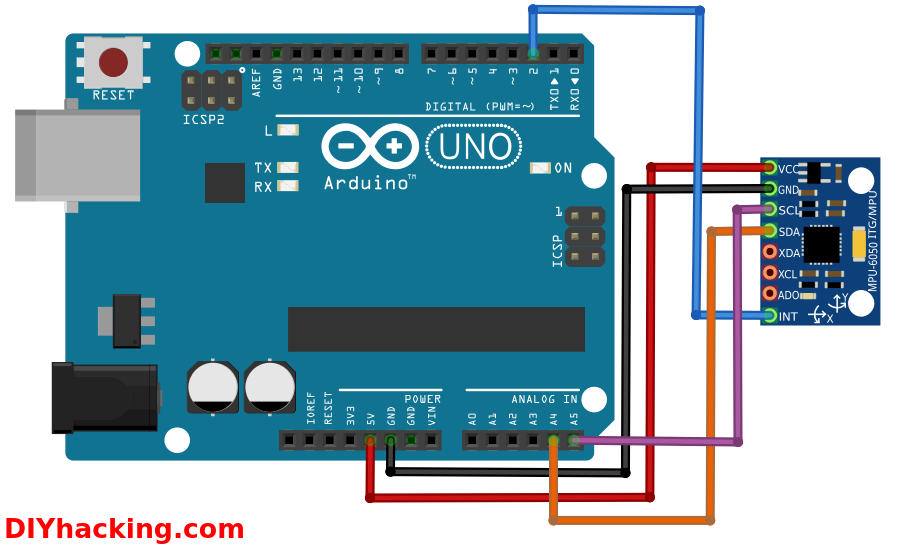
[](http://diyhacking.com/wp-content/uploads/2014/10/gyro-1024x576.jpg)

Piezo Electric Gyroscope

Gyroscopes work on the principle of Coriolis acceleration. Imagine that there is a fork like structure, that is in constant back and forth motion. It is held in place using piezo electric crystals. Whenever, you try to tilt this arrangement, the crystals experience a force in the direction of inclination. This is caused as a result of the inertia of the moving fork. The crystals thus produce a current in consensus with the piezo electric effect, and this current is amplified. The values are then refined by the host microcontroller. Now check this short [video](https://www.youtube.com/watch?v=zwe6LEYF0j8) that explains, how a MEMS gyroscope works.

#### Step 1: Interfacing the Arduino MPU 6050

The MPU 6050 communicates with the Arduino through the I2C protocol. The MPU 6050 is connected to Arduino as shown in the following diagram. Here, if your MPU 6050 module has a 5V pin, then you can connect it to your arduino’s 5V pin. Else, you will have to connect it to the 3.3V pin. Next, the GND of the arduino is connected to the GND of the MPU 6050.

[](http://diyhacking.com/wp-content/uploads/2014/11/conn.png)

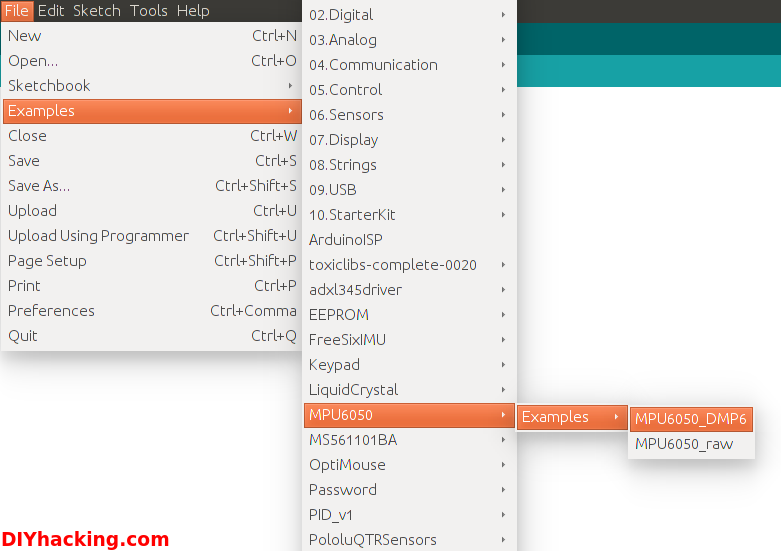
Arduino MPU 6050 connections

The program we will be running here, also takes advantage of the arduino’s interrupt pin. Therefore, connect arduino’s digital pin 2 (interrupt pin 0) to the pin labelled as INT on the MPU 6050. Next, we need to set up the I2C lines. For this connect the pin labelled as SDA on the MPU 6050 to the arduino’s analog pin 4 (SDA). And the pin labelled as SCL on the MPU 6050 to the arduino’s analog pin 5 (SCL). And that’s it, you have finished wiring up the Arduino MPU 6050.

#### Step 2: Uploading the code and testing the Arduino MPU 6050

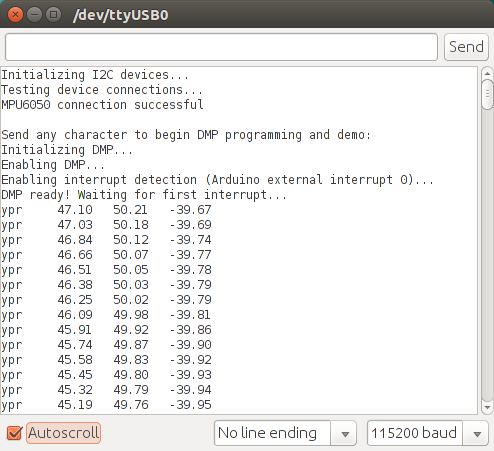
To test the Arduino MPU 6050, first download the arduino library for MPU 6050, developed by Jeff Rowberg. You can find the library [here](http://diyhacking.com/projects/MPU6050.zip). Next you have to unzip/extract this library. And then take the folder named “MPU6050” and paste it inside the arduino’s “library” folder. That is, go to the location where you have installed arduino (Arduino –> libraries) and paste it inside the libraries folder. You might also have to do the same thing to install the I2Cdev library as well, if you don’t already have it for your arduino. Do the same procedure as above for installing it, you can find the file here: [I2Cdev](http://diyhacking.com/projects/I2Cdev.zip) library.

If you have done this correctly, then when you open the arduino IDE, you can see “MPU6050” in File –> Examples. Next, open the example program from: File –> Examples –> MPU6050 –> Examples –> MPU6050\_DMP6.

[](http://diyhacking.com/wp-content/uploads/2014/11/arduinoCode.png)

Arduino MPU 6050 DMP code

Now, you have to upload this code to arduino. After uploading the code, open up the serial monitor and set the baud rate as 115200. Next, check if you see stuff like “Initializing I2C devices…” on the serial monitor. If you don’t, just press the reset button. Now, you’ll see a line saying  “Send any character to begin DMP programming and demo: “. So just type in any character on the serial monitor and send it. And you will start seeing the yaw, pitch and roll values coming in from the MPU 6050. Like so:

[](http://diyhacking.com/wp-content/uploads/2014/11/mpuSerial.png)

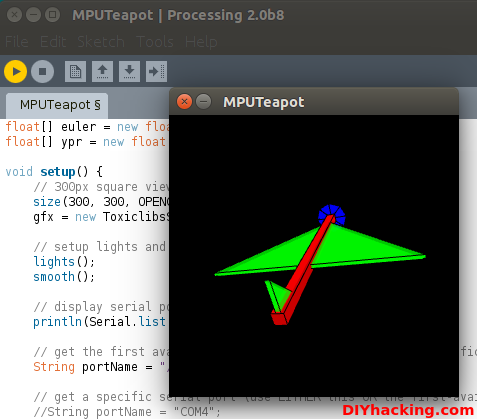
Arduino MPU 6050 Serial Monitor

DMP stands for Digital Motion Processing. Here, the Invense’s MPU 6050 has an inbuilt motion processor. It processes the values from the accelerometer and gyroscope to give us accurate 3D values.

Also, you will need to wait about 10 secs before you get accurate values from the Arduino MPU 6050. After which, the values will begin to stabilize. Just check out the video, if you have any doubts:

#### Step 3: Modelling the values from Arduino MPU 6050 in 3D using processing (Optional)

If you want to see the 3D model of the sensor, continue reading. For viewing the 3D representation of the data from the MPU 6050, you need to install the “processing” software, you can do that from here: [processing](https://processing.org/download/?processing) IDE. Processing is almost similar to arduino, except for a couple of functions. Processing is mainly used for visualizing data and rendering it in 2D/3D models.

[](http://diyhacking.com/wp-content/uploads/2014/11/process.png)

Arduino MPU 6050 with Processing

After installing the processing IDE, next you will need to download a library called “Toxi”. This library is necessary for our Arduino MPU 6050 processing example. Download the latest zip file for the library from [here](https://bitbucket.org/postspectacular/toxiclibs/downloads/). Next, you need to extract this file and paste the folder named “toxiclibs-complete-0020” in the libraries directory of processing. You can find the “libraries” folder inside the Sketchbook folder of processing. If you don’t, then you will have to create a new folder called “libraries” there, and paste the toxiclibs inside it.

To visualize the 3D model in processing, first you have to upload the  arduino code for MPU 6050 (MPU6050\_DMP6). Before doing that, you need to comment the line in the arduino MPU6050\_DMP6 code which says:

#define OUTPUT\_READABLE\_YAWPITCHROLL  by  //#define OUTPUT\_READABLE\_YAWPITCHROLL.

And uncomment the line which says:

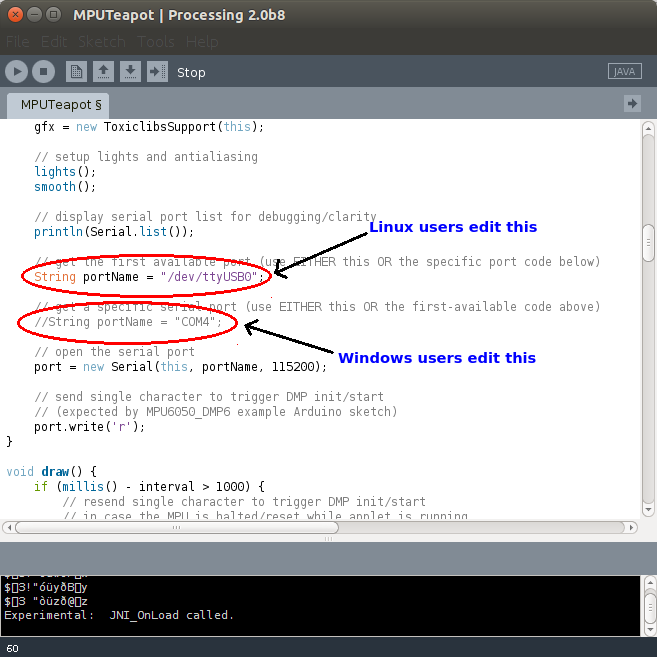
//#define OUTPUT\_TEAPOT  by  #define OUTPUT\_TEAPOT.

Next, you have to open the processing example for the MPU 6050. Open processing, then: File –> Open. And then navigate to the folder where you had installed the MPU6050 library for arduino. You can find the processing example in: MPU6050 –> Examples –> MPU6050\_DMP6 –> Processing –> MPUTeapot.

In this code, you have to then check the serial port which is defined in it. By default the line that defines it for linux/mac users is like:

String portName = “/dev/ttyUSB1”;.

You need to change ttyUSB1 to the port on which your arduino is connected.

[](http://diyhacking.com/wp-content/uploads/2014/11/processCode.png)

Editing the Processing file for MPU 6050

And for windows users, you need to comment the line which says:

String portName = “/dev/ttyUSB1”;  by  //String portName = “/dev/ttyUSB1”;.

And uncomment the line which says:

//String portName = “COM4”;  by  String portName = “COM4”;.

And replace “COM4” with the COM port on which your arduino is connected (check this by going into arduino and Tools –> Serial Port).

Now we can test the whole setup. First upload the arduino code (MPU6050\_DMP6) through arduino. And please remember NOT to open the serial monitor. Next run the processing code (MPUTeapot) by pressing the button with the “play” symbol. You will see a small plane like object. Wait for about 10 secs for the MPU 6050 values to get stabilized. After, which you can see the 3D model of your MPU 6050, which moves accordingly with the sensor. Now, check out the demo for the Arduino MPU 6050 3D model:

<https://youtu.be/ZzOAuscDIiU>

# MMA8452 3 axes

Découvert sur Aliexpress

|  |  |
| --- | --- |
| <https://fr.aliexpress.com/item/MMA8452-Three-3-Axis-Accelerator-Accelerometer-Sensor-Module-Shield-Board-For-Arduino/32740794357.html?spm=2114.06010108.3.1.dVD6NW&ws_ab_test=searchweb0_0,searchweb201602_3_10065_10068_10501_10503_10000032_119_10000025_10000029_430_10000028_10060_10062_10056_10055_10000062_10054_10059_10099_10000022_10000012_10103_10000015_10102_10096_10000018_10000019_10000056_10000059_10052_10053_10107_10050_10106_10051_10000053_10000007_10000050_10084_10118_10083_10000047_10080_10082_10081_10110_10111_10112_10113_10114_10115_10000041_10000044_10078_10079_10000038_429_10073_10000035_10121-10503_10501,searchweb201603_10,afswitch_1,single_sort_1_default&btsid=187ce7cd-39ba-49f1-b0cc-8bafa21b6330> |  |

**Description:**

* le MMA8452Q est une puce de faible puissance, trois-axe, capacitif micro-usiné accéléromètre avec 12 bits de résolution.
* cette accéléromètre est emballé avec fonctions intégrées avec flexible utilisateur programmable options, configurable à deux interrompre pins.
* intégré des fonctions d'interruption permettent pour puissance globale économies soulager le processeur hôte de données de vote en continu.
* le MMA8452Q a sélectionnable par l'utilisateur complet échelles de ± 2g/± 4g/± 8g avec haute filtrée passe données ainsi que non filtrée données disponibles en temps réel.
* le dispositif peut être configuré pour générer inertielle wake-up interrompre les signaux de n'importe quel combinaison de la configurable intégré fonctions permettant la MMA8452Q à suivre la situation et rester dans un mode de faible puissance pendant les périodes d'inactivité.

**caractéristiques:**

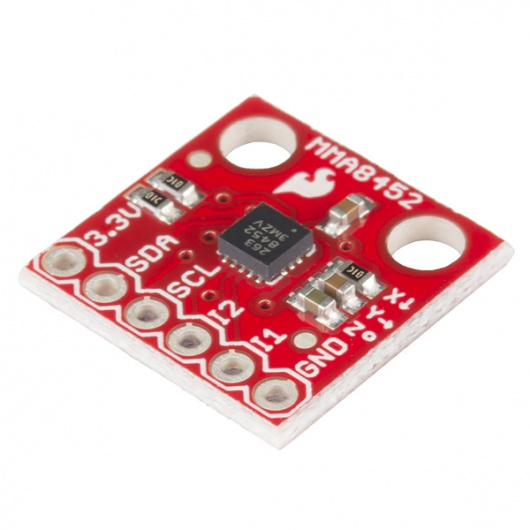
* 1.95 V à 3.6 V tension d'alimentation
* 1.6 V à 3.6 V interface tension
* ± 2g/± 4g/± 8g dynamiquement sélectionnable à grande échelle
* sortie Données Taux (ODR) de 1.56Hz à 800 Hz12-bit et 8-bit numérique sortie
* I2C numérique sortie interface (fonctionne à 2.25 MHz avec 4.7 kΩ pullup)
* deux programmable interrupt pins pour six sources d'interruption
* trois intégré canaux de détection de mouvement
* Orientation (Portrait/Paysage) détection avec ensemble hystérésis
* Filtre Passe-haut Données disponibles en temps réel
* Consommation de courant: 6 μA-165 μA

# MMA8452Q Accelerometer Breakout Hookup Guide Sparkfun

<https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide>

## Introduction

Freescale’s [MMA8452Q](https://www.sparkfun.com/products/10953) is a smart, low-power, three-axis, capacitive micro-machined accelerometer with 12 bits of resolution. It’s perfect for any project that needs to sense orientation or motion. We’ve taken that accelerometer and stuck it on a [breakout board](https://www.sparkfun.com/products/12756), in order to make interfacing with the tiny, QFN package a bit easier.

[](https://www.sparkfun.com/products/12756)

The MMA8452Q is just a solid 3-axis accelerometer. It supports three, selectable sensing ranges: ± 2g, 4g, or 8g. It sports features like orientation detection, single and double-tap sensing, and low power modes. It’s a digital sensor – communicating over an [I2C interface](https://learn.sparkfun.com/tutorials/i2c) – so you’ll get reliable, noise-free data using as few as two microcontroller pins to interact with the accelerometer.

### Covered In This Tutorial

This tutorial aims to get you started up with the MMA8452Q as quickly as possible. It’s split into a handful of pages to explain how the board works and how to use it:

* [Hardware Overview](https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide/hardware-overview) – An overview of the MMA8452Q IC itself, and the breakout board we’ve stuffed it onto.
* [Example Hookup](https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide/example-hookup) – How to connect the MMA8452Q to the ubiquitous Arduino. This hardware hookup is used in the next section…
* [Example Code](https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide/example-code) – We’ve written an **Arduino library** to help make your Arduino sketch cleaner and easier to write. On this page we’ll show off an example sketch using the library and demonstrate how to use the library on your own to control the accelerometer.

### Required Materials

To follow along with this tutorial, the most unique components you’ll need are:

* [MMA8452Q Breakout Board](https://www.sparkfun.com/products/12756) – This is a good place to start (unless you plan on dead-bugging the tiny [IC itself](https://www.sparkfun.com/products/10953).
* **Arduino Board** – You should be able to use any Arduino board you have handy – [Arduino Uno](https://www.sparkfun.com/products/11021), [RedBoard](https://www.sparkfun.com/products/11575), [Pro](https://www.sparkfun.com/products/10914), [Mega](https://www.sparkfun.com/products/11061), …anything.
* [330Ω Resistors](https://www.sparkfun.com/products/11507) – Assuming you’re using a 5V-based microcontroller, these will help protect the MMA8452Q from out-of-spec voltages hitting its pins.

You’ll also need some sort of interconnect between the breakout and Arduino. A [breadboard](https://www.sparkfun.com/products/12002) and [jumper wires](https://www.sparkfun.com/products/11026) are always an easy combo.

Finally, you’ll need to solder a connector onto the MMA8452Q breakout. If you’re using a breadboard to hook it up, we recommend [straight male headers](https://www.sparkfun.com/products/116).

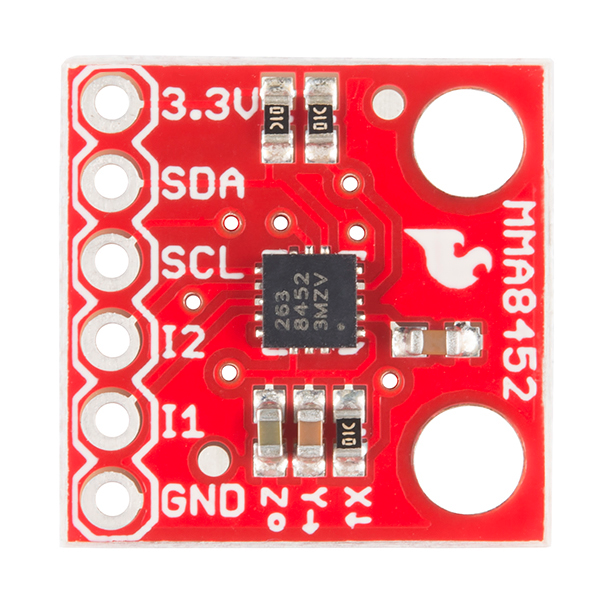
### Suggested Reading

Before continuing on with this tutorial, we recommend you be somewhat familiar with the concepts in these tutorials:

* [Accelerometer Basics](https://learn.sparkfun.com/tutorials/accelerometer-basics) – This is a great primer on accelerometers – how they work, and why they’re used.
* [I2C](https://learn.sparkfun.com/tutorials/i2c) – An introduction to the communication protocol we’ll use to get our Arduino to talk to the MMA8452Q.
* [Accelerometer Buying Guide](https://www.sparkfun.com/pages/accel_gyro_guide) – If you’re not sure which accelerometer is best for you, check out this guide.
* [Logic Levels](https://learn.sparkfun.com/tutorials/logic-levels) – The MMA8452Q is a 3.3V device, so if you’re using a 5V microcontroller (like the Arduino) you’ll have to pay attention to the logic levels!

## Hardware Overview

The MMA8452Q Breakout Board breaks out a select few of the most important pins on the accelerometer.

[](https://cdn.sparkfun.com/assets/learn_tutorials/2/4/9/mma8452q-top.png)

A little bit about each pin:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Label** | **Pin Function** | **Input/Output** | **Notes** |
| 3.3V | Power Supply | Input | Should be between 1.95 - 3.6V |
| SDA | I2C Data Signal | Bi-directional | Bi-directional data line. Voltage should not exceed power supply (e.g. 3.3V). |
| SCL | I2C Clock Signal | Input | Master-controlled clock signal. Voltage should not exceed power supply (e.g. 3.3V). |
| I2 | Interrupt 2 | Output | Programmable interrupt — can indicate data ready, orientation change, tap, and more. |
| I1 | Interrupt 1 | Output | Programmable interrupt — can indicate data ready, orientation change, tap, and more. |
| GND | Ground | Input | 0V/common voltage. |

### Voltage Supply Requirements

The big alert here is that the MMA8452Q has a **maximum voltage of 3.6V** – that range applies to both the power supply and the I2C pins. If you’re using the sensor in a 3.3V system you can plug it right in, but if you’re using it with a 5V device (like an [Arduino Uno](https://www.sparkfun.com/products/11021)), some **level-shifting** is required between devices.

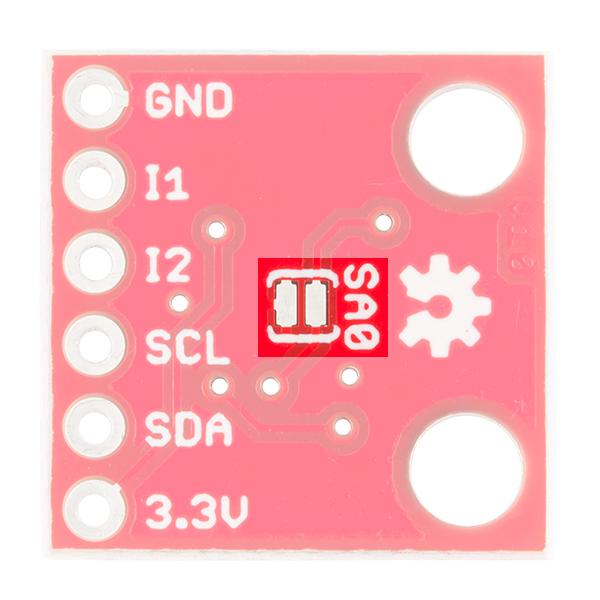
Fortunately, you don’t need a lot of power to make the MMA8452Q work. In normal operating mode it can require anywhere between 7 and 165 µA.

### Address Select Jumper

The MMA8452Q features a **selectable I2C address** – just in case you’re running multiple MMA8452Qs on the same bus (or maybe you have an address conflict). To select the address, a pin on the accelerometer – “SA0” – can be tied to either power or ground.

|  |  |
| --- | --- |
| **SA0 Voltage** | **MMA8452Q I2C Address** |
| 0V | 0x1C |
| 3.3V (VCC) | 0x1D |

The breakout board includes a jumper, on the back side, to help tie this pin high or low.

[](https://cdn.sparkfun.com/assets/learn_tutorials/2/4/9/mma8452q-back-jumper.png)

By default the jumper is open, which will pull the SA0 pin high (there’s a resistor on the top side of the board to help accomplish that task). If you close the jumper, by applying a small solder blob to connect both pads together, SA0 will be pulled to ground.

For most use cases, where you’re only using one MMA8452Q, you can leave this jumper untouched. In that case the I2C address will be 0x1D.

## Example Hookup

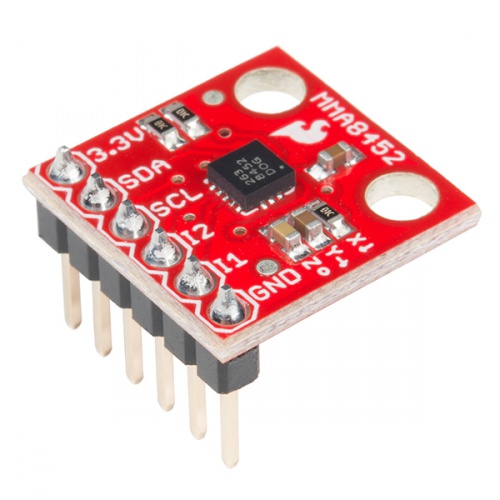
### Solder Something

Before you can plug the breakout board into a breadboard, or connect it to anything, you’ll need to solder connectors or wires to the break-out pins. What, exactly, you solder into the board depends on how you’re going to use it.

If you’re going to use the breakout board in a [breadboard](https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard) or similar 0.1"-spaced perfboard, we recommend soldering [straight male headers](https://www.sparkfun.com/products/116) into the pins (there are also [long headers](https://www.sparkfun.com/products/10158) if you need ‘em).

If you’re going to mount the breakout into a tight enclosure, you may want to solder wires ([stranded](https://www.sparkfun.com/products/11375) or [solid-core](https://www.sparkfun.com/products/11367)) directly into the pins.

You can also purchase the MMA8452Q with headers already soldered to the breakout.

[](https://www.sparkfun.com/products/13926)

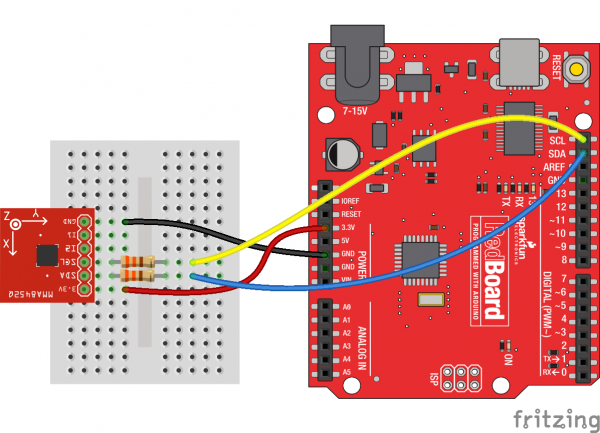
### [SparkFun Triple Axis Accelerometer Breakout - MMA8452Q (with Headers)](https://www.sparkfun.com/products/13926)

[In stock](https://www.sparkfun.com/static/bubbles/) BOB-13926

$10.49

### Simple Hookup

We’ll use Arduino to communicate with the MMA8452Q and interpret the data from the sensor. Since we’re using I2C, all we need is two wires between the Arduino and accelerometer (aside from power and ground). Here’s the hookup:

[](https://cdn.sparkfun.com/assets/learn_tutorials/2/4/9/mma8452q-hookup_bb.png)

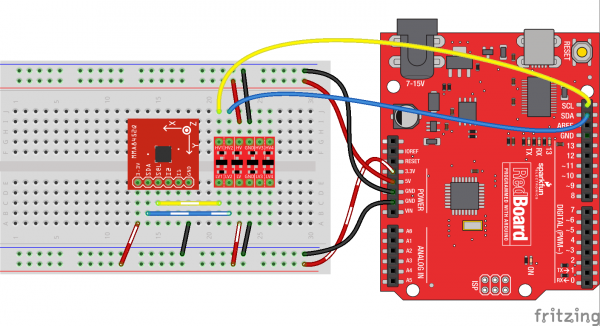
We simply have to supply the accelerometer with power (3.3V and GND), then hookup the SCL and SDA lines between devices. A couple of [330Ω resistors](https://www.sparkfun.com/products/11507) in series on each I2C line will help to perform some simple level shifting. If you want more advanced level shifting…

### Level Shifting

Since the MMA8452Q’s maximum voltage is 3.6V, you’ll need to do some level shifting between your Arduino and accelerometer. Powering the accelerometer off the Arduino’s 3.3V rail is a good start, but you’ll also need to add some protection on the SDA and SCL lines.

In the example hookup above, we used a pair of series resistors on the SDA and SCL lines. This version of “level shifting” works in a pinch, but, if you want a more reliable level-shifting setup, we recommend using a more robust level shifter between the boards.

There are a handful of logic-level shifting boards available. For example, here’s a hookup using a [Bi-Directional Logic Level Shifter](https://www.sparkfun.com/products/12009) between Arduino and accelerometer:

[](https://cdn.sparkfun.com/assets/learn_tutorials/2/4/9/mma8452q-levelshift-hookup_bb.png)

Or, if you just want to skip level shifting entirely, you can use a [3.3V Arduino Pro](https://www.sparkfun.com/products/10914) (or [Pro Mini](https://www.sparkfun.com/products/11114)), and run the whole system at 3.3V. As anyone who’s built electronics knows: there’s more than one way to skin the cat.

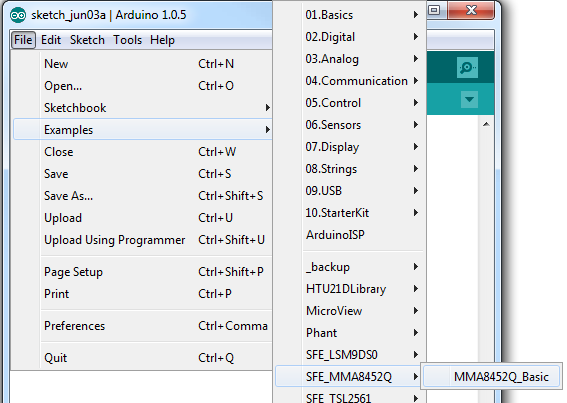
## Example Code

We’ve written an Arduino library to make interfacing with the MMA8452Q as easy as can be. [Click here to download the library](https://cdn.sparkfun.com/assets/learn_tutorials/2/4/9/SFE_MMA8452Q-library.zip). Or you can grab the latest, greatest version over on the [GitHub repository](https://github.com/sparkfun/MMA8452_Accelerometer/tree/master).

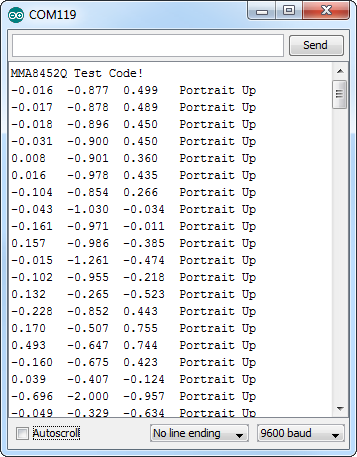
To install the library, extract the SFE\_MMA8452Q folder into the libraries folder within your Arduino sketchbook. For help with Arduino library installations, we recommend checking out our [How to Install an Arduino Library tutorial](https://learn.sparkfun.com/tutorials/installing-an-arduino-library).

### Load Up the Example Sketch

Once you’ve installed the SFE\_MMA8452Q library, restart Arduino. Then go to **File** > **Examples** > **SFE\_MMA8452Q** > **MMA8452Q\_Basic** to open the example sketch.

[](https://cdn.sparkfun.com/assets/learn_tutorials/2/4/9/opening-example.png)

Once you’ve set your Board and Serial Port, upload the sketch to your Arduino. Then **open the serial monitor**. You’ll begin to see acceleration values stream by, in addition to some information about the sensor’s orientation.

[](https://cdn.sparkfun.com/assets/learn_tutorials/2/4/9/serial-monitor-example.png)

Try moving the sensor around to change those values. If it is motionless, flat on the desk, then an acceleration of 1g should be felt on the z-axis, while the others feel around 0. Test the other axes by rotating the board and making them feel the pull of gravity.

### Using the SFE\_MMA8452Q Library

Here are some tips on using the MMA8452Q Arduino library so you can embed it into an Arduino sketch of your own.

#### Include the Library (Global)

To begin, you need to “include” the library in your sketch:

#include <Wire.h> // Must include Wire library for I2C

#include <SFE\_MMA8452Q.h> // Includes the SFE\_MMA8452Q library

The library also requires that you include Wire.h in your sketch. Make sure you include that before you include the SFE\_MMA8452Q.h file.

#### Create an MMA8452Q Object (Global)

Once the library is included, you can create an MMA8452Q object. This line of code will do it for you:

MMA8452Q accel; // Default MMA8452Q object create. (Address = 0x1D)

Optionally, you can define the 7-bit **I2C address** of your MMA8452Q in this parameter, using one of these lines of code:

MMA8452Q accel(0x1C); // Initialize the MMA8452Q with an I2C address of 0x1C (SA0=0)

MMA8452Q accel(0x1D); // Initialize the MMA8452Q with an I2C address of 0x1D (SA0=1)

But if you’ve left the [address jumper](https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide/hardware-overview#address_jumper) untouched (meaning the “SA0” pin is connected to VCC), you can call the default (no parameter) constructor shown earlier.

#### Initialize the MMA8452Q (Setup)

Finally, in the setup() function of your sketch, you can initialize the accelerometer using the init() function. The init() function verifies communication with the accelerometer, and sets up the **full-scale range** and **output data rate**.

Again, you have a few options here. You can use a simple declaration like below. This will initialize the accelerometer with range of **±2g** and an output data rate of **800 Hz** (turns the accelerometer up to the max!):

accel.init(); // Default init: +/-2g and 800Hz ODR

If you want to specify the acceleration and output data rate, you can instead use an init() function like this:

accel.init([scale], [odr]); // Init and customize the FSR and ODR

Scale can be either SCALE\_2G, SCALE\_4G, or SCALE\_8G. The “odr” variable can be either ODR\_800, ODR\_400, ODR\_200, ODR\_100, ODR\_50, ODR\_12, ODR\_6, or ODR\_1, respectively setting the data rate to 800, 400, 200, 100, 50, 12.5, 6.25, or 1.56 Hz.

#### Reading and Using Values

Once you’ve set the accelerometer up, you can immediately start reading the data coming out of the chip. Reading and using the values is a two-step process. First, call the read() function to pull in the values.

accel.read(); // Update acceleromter data

After you’ve called the read() function, you can use either of two sets of values to use the data. Reading from the x, y, and z class variables will return a signed 12-bit integer read straight out of the accelerometer.

xAcceleration = accel.x; // Read in raw x-axis acceleration data

Serial.print("Acceleration on the x-axis is ");

Serial.println(xAcceleration);

Or, if you want a value with physical units, you can use the cx, cy, and cz class variables. These are the **calculated** acceleration values read out of the accelerometer; they’ll be in units of g’s.

zAcceleration = accel.cz; // Read in calculated z-axis acceleration

Serial.print("Acceleration on the z-axis is: ");

Serial.print(zAcceleration);

Serial.println(" g's");

Remember! Those variables are only updated after the read() function is called. Make sure that happens before you start using acceleration values.

#### Reading Portrait/Landscape

The MMA8452Q has all sorts of nifty, extra features, one of which is **orientation detection** – it can estimate if it’s being held in landscape mode, portrait mode, or flat.

To read the portrait/landscape data from the accelerometer, use the readPL() function. This function returns a byte, which will either be equal to PORTRAIT\_U, PORTRAIT\_D, LANDSCAPE\_R, LANDSCAPE\_L, or LOCKOUT.

byte pl = accel.readPL();

switch (pl)

{

case PORTRAIT\_U:

Serial.print("Portrait Up");

break;

case PORTRAIT\_D:

Serial.print("Portrait Down");

break;

case LANDSCAPE\_R:

Serial.print("Landscape Right");

break;

case LANDSCAPE\_L:

Serial.print("Landscape Left");

break;

case LOCKOUT:

Serial.print("Flat");

break;

}

As in the example above, you can use if or switch statements to check which orientation your accelerometer is in.

## Resources and Going Further

Thanks for reading! We’re excited to see what you build with the MMA8452Q. If you’re left needing more MMA8452Q-related documentation, check out some of these resources:

* [MMA8452Q Datasheet](http://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/MMA8452Q.pdf) – Loads of information about the MMA8452Q’s electrical characteristics, registers, communication specifications, and more.
* [MMA8452Q Breakout Schematic](https://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/MMA8452Q_Breakout-v10.pdf) – PDF schematic of the MMA8452Q Breakout board.
* [MMA8452Q Breakout Eagle Files](https://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/MMA8452Q_Breakout-v10.zip) – PCB design files for the MMA8452Q Breakout.
* [MMA8452Q Breakout GitHub Repo](https://github.com/sparkfun/MMA8452_Accelerometer) – Design files and example code all related to the MMA8452Q.

### Going Further …/… Des liens vers des montages

…/… Des liens vers des montages

# MMA8451 9 axes

Découvert sur Aliexpress chez ModuleFans <https://fr.aliexpress.com/store/612195>

|  |  |
| --- | --- |
| <https://fr.aliexpress.com/item/MMA8451-Three-Axis-Accelerator-Accelerometer-Sensor-Module-Shield-For-for-Ar/1859122447.html?spm=2114.06010108.3.1.lfgSXe&ws_ab_test=searchweb0_0,searchweb201602_3_10065_10068_10501_10503_10000032_119_10000025_10000029_430_10000028_10060_10062_10056_10055_10000062_10054_10059_10099_10000022_10000012_10103_10000015_10102_10096_10000018_10000019_10000056_10000059_10052_10053_10107_10050_10106_10051_10000053_10000007_10000050_10084_10118_10083_10000047_10080_10082_10081_10110_10111_10112_10113_10114_10115_10000041_10000044_10078_10079_10000038_429_10073_10000035_10121,searchweb201603_10,afswitch_1,single_sort_1_default&btsid=ae1f46ad-7178-463a-85f4-38aee8da9a58> | aeProduct.getSubject() |

# Adafruit MMA8451 accelerometer

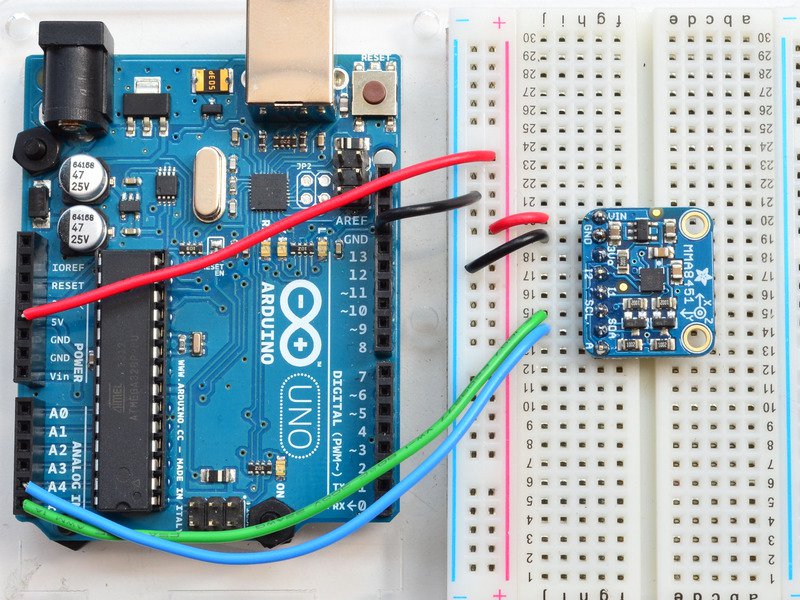
<https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/>

<https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test>

## Wiring & Test

by [lady ada](https://learn.adafruit.com/users/adafruit2)

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, just make sure it has I2C **with repeated-start support**, then port the code - its pretty simple stuff!

[](https://learn.adafruit.com/assets/18508)

* Connect **Vin** to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
* Connect **GND** to common power/data ground
* Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**
* Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**

The MMA8451 has a default I2C address of **0x1D** and can be changed to 0x1C by tying the **A** pin to GND

## Download Adafruit\_MMA8451

To begin reading sensor data, you will need to [download the Adafruit\_MMA8451\_Library from our github repository](https://github.com/adafruit/Adafruit_MMA8451_Library). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

Rename the uncompressed folder **Adafruit\_MMA8451** and check that the **Adafruit\_MMA8451** folder contains **Adafruit\_MMA8451.cpp** and **Adafruit\_MMA8451.h**  
  
Place the **Adafruit\_MMA8451** library folder your **arduinosketchfolder/libraries/** folder.   
You may need to create the **libraries** subfolder if its your first library. Restart the IDE.  
  
We also have a great tutorial on Arduino library installation at:  
<http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use>

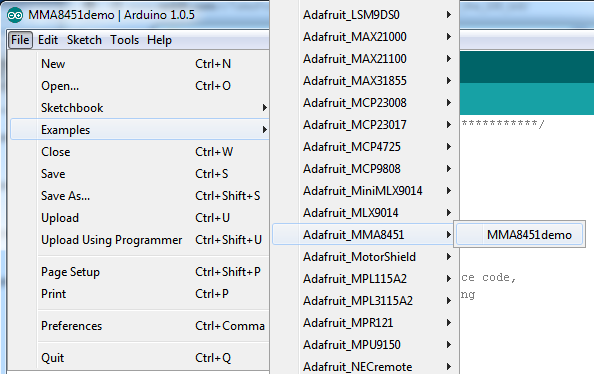
## Download Adafruit\_Sensor

The MMA8451 library uses the Adafruit\_Sensor support backend so that readings can be normalized between sensors. [You can grab Adafruit\_Sensor from the github repo](https://github.com/adafruit/Adafruit_Sensor) or just click the button below.

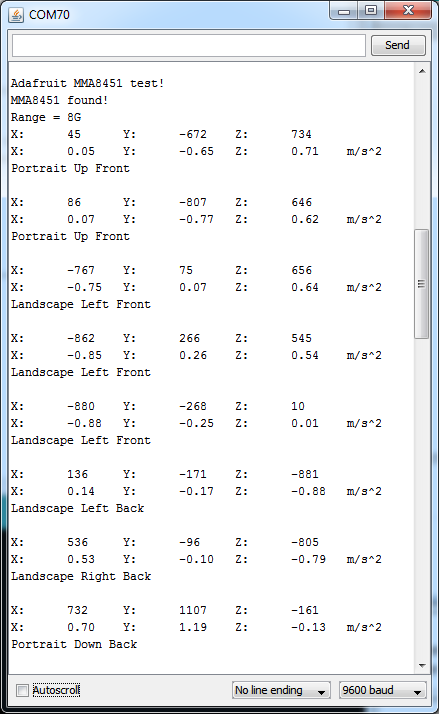
Install like you did with Adafruit\_MMA8451

## Load Demo

Open up **File->Examples->Adafruit\_MMA8451->MMA8451demo** and upload to your Arduino wired up to the sensor

[](https://learn.adafruit.com/assets/18509)

Thats it! Now open up the serial terminal window at 9600 speed to begin the test.

[](https://learn.adafruit.com/assets/18510)

There's three lines of output from the sensor.  
  
Example for line 1:

X: 45 Y: -672 Z: 734

This is the "raw count" data from the sensor, its a number from -8192 to 8191 (14 bits) that measures over the set range. The range can be set to 2G, 4G or 8G  
  
Example for line 2:

X: -0.07 Y: 0.09 Z: 0.98 m/s^2

This is the Adafruit\_Sensor'ified nice output which is in m/s\*s, the SI units for measuring acceleration. No matter what the range is set to, it will give you the same units, so its nice to use this instead of mucking with the raw counts  
  
Example for line 3:

Portrait Up Front

This is the output of the orientaiton detection inside the chip. Since inexpensive accelerometers are often used to detect orientation and tilt, this sensor has it built in. The orientation can be Portrait or Landscape, then Up/Down or Left/Right and finally tilted forward or tilted back. Note that if the sensor is tilted less than 30 degrees it cannot determine the forward/back orientation. If you play with twisting the board around you'll get the hang of it.

## Library Reference

The library we have is simple and easy to use  
  
You can create the **Adafruit\_MMA8451** object with:

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. Adafruit\_MMA8451 mma = Adafruit\_MMA8451();

There are no pins to set since you must use the I2C bus!  
  
Then initialize the sensor with:

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. mma.begin()

this function returns **True** if the sensor was found and responded correctly and **False** if it was not found. We suggest something like this:

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. if (! mma.begin()) {
2. Serial.println("Couldnt start")
3. while (1);
4. }
5. Serial.println("MMA8451 found!");

## Set & Get Range

You can set the accelerometer max range to ±2g, ±4g or ±8g with

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. mma.setRange(MMA8451\_RANGE\_2\_G);
2. mma.setRange(MMA8451\_RANGE\_4\_G);
3. mma.setRange(MMA8451\_RANGE\_8\_G);

And read what the current range is with

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. mma.getRange()

Which returns 1 for ±2g, 2 for ±4g and 3 for ±8g

## Read Raw Count Data

You can read the raw counts data with

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. mma.read();

The x, y and z data is then available in **mma.x**, **mma.y** and **mma.z**  
All three are read in one transaction.

## Reading Normalized Adafruit\_Sensor data

We recommend using the Adafruit\_Sensor interface which allows reading into an event structure. First create a new event structure

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. sensors\_event\_t event;

Then read the event whenever you want

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. mma.getEvent(&event);

The normalized SI unit data is available in **event.acceleration.x,** **event.acceleration.y** and **event.acceleration.z**

## Read Orientation

The sensor has built in tilt/orientation detection. You can read the current orientation with

[Copy Code](https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test)

1. mma.getOrientation();

The return value ranges from 0 to 7

* 0: Portrait Up Front
* 1: Portrait Up Back
* 2: Portrait Down Front
* 3: Portrait Down Back
* 4: Landscape Right Front
* 5: Landscape Right Back
* 6: Landscape Left Front
* 7: Landscape Left Back