# MMA8452Q Accelerometer Breakout Hookup Guide

CONTRIBUTORS: *JIMB0*

Introduction

Freescale's MMA8452Q 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, in order to make interfacing with the tiny, QFN package a bit easier.
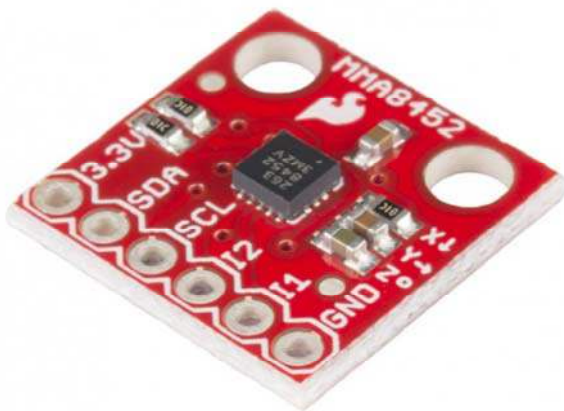
The MMA8452Q is just a solid 3-axis accelerometer. It supports three, selectable sensing ranges: ± 2g, 4g, or 8g. It sports features like orientation detection, single and double-tap sensing, and low power modes. It's a

digital sensor – communicating over an I²C interface – 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 – An overview of the MMA8452Q IC itself, and the breakout board we've stuffed it onto.
- Example Hookup – How to connect the MMA8452Q to the ubiquitous Arduino. This hardware hookup is used in the next section…
- 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 – This is a good place to start (unless you plan on dead-bugging the tiny IC itself.
- **Arduino Board** – You should be able to use any Arduino board you have handy – Arduino Uno, RedBoard, Pro, Mega, …anything.
- 330Ω Resistors – 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 and jumper wires 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.

Suggested Reading

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

- Accelerometer Basics – This is a great primer on accelerometers – how they work, and why they're used.
- I²C – An introduction to the communication protocol we'll use to get our Arduino to talk to the MMA8452Q.

- Accelerometer Buying Guide – If you're not sure which accelerometer is best for you, check out this guide.
- 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.
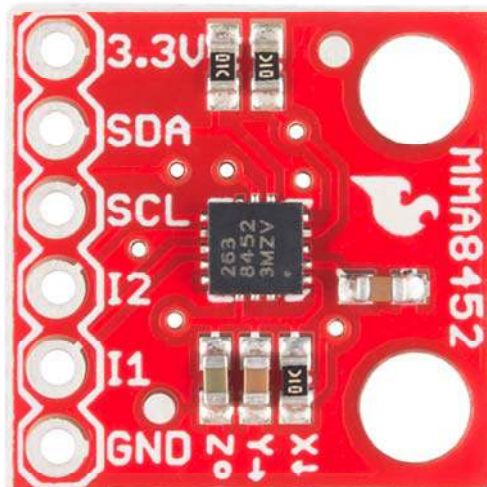
A little bit about each pin:

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 I^{2}C 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), 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 I^{2}C 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 I^{2}C 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.
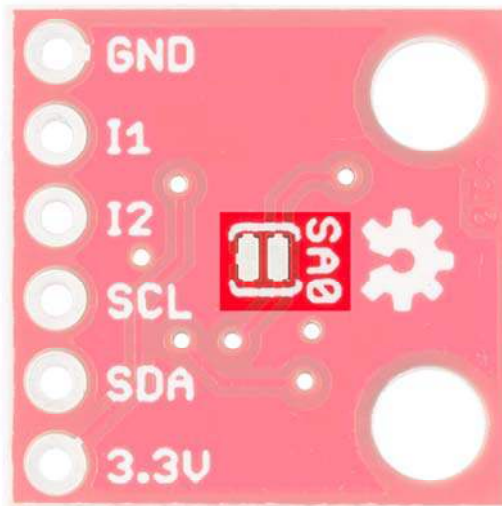
By default the jumper is open, which will pull the SA0 pin high (there's a resistor on the top side of the board to help accomplish that task). If you close the jumper, by applying a small solder blob to connect both pads together, SA0 will be pulled to ground.

For most use cases, where you're only using one MMA8452Q, you can leave this jumper untouched. In that case the I^{2}C 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 or similar 0.1"-spaced perfboard, we recommend soldering straight male headers into the pins (there are also long headers if you need 'em).

If you're going to mount the breakout into a tight enclosure, you may want to solder wires (stranded or solid-core) directly into the pins.

Simple Hookup

We'll use Arduino to communicate with the MMA8452Q and interpret the data from the sensor. Since we're using I^{2}C, all we need is two wires between the Arduino and accelerometer (aside from power and ground). Here's the hookup:
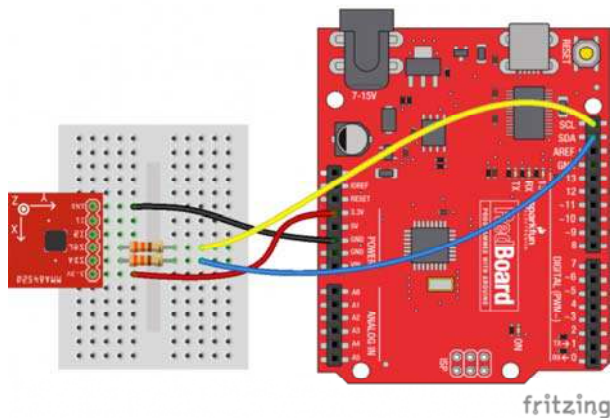

We simply have to supply the accelerometer with power (3.3V and GND), then hookup the SCL and SDA lines between devices. A couple of 330Ω resistors in series on each I^{2}C line will help to perform some simple level shifting. If you want more advanced level shifting…

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 between Arduino and accelerometer:
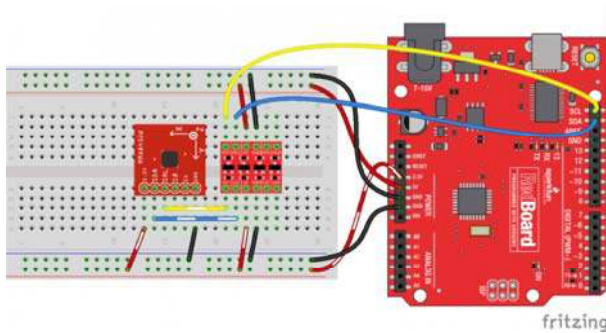
Or, if you just want to skip level shifting entirely, you can use a 3.3V Arduino Pro (or Pro Mini), and run the whole system at 3.3V. As anyone who's built electronics knows: there's more than one way to skin the cat.

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. Or you can grab the latest, greatest version over on the GitHub repository.

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.

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.
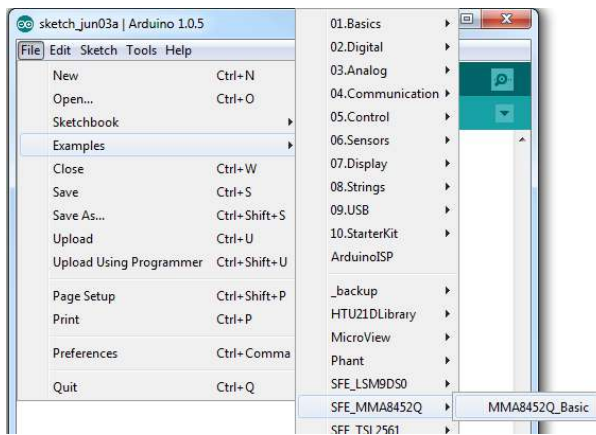
Once you've set your Board and Serial Port, upload the sketch to your Arduino. Then **open the serial monitor**. You'll begin to see acceleration values stream by, in addition to some information about the sensor's orientation.
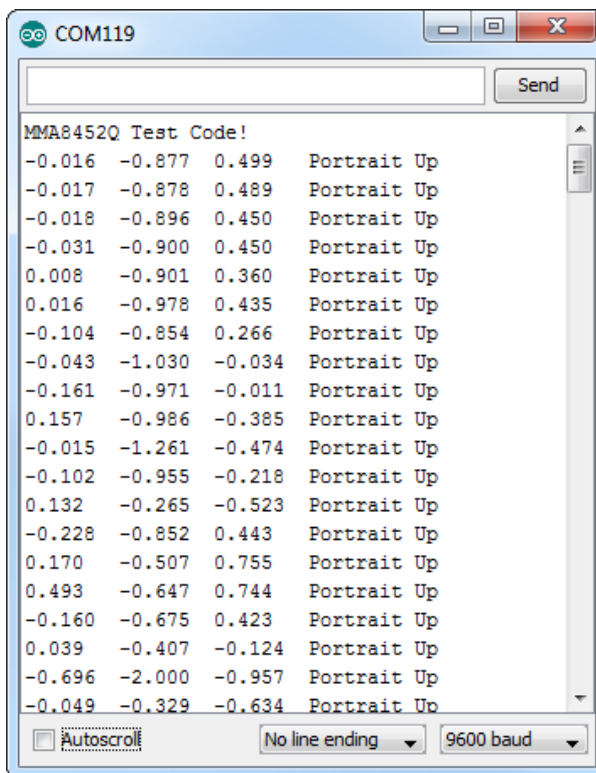
Try moving the sensor around to change those values. If it is motionless, flat on the desk, then an acceleration of 1g should be felt on the z-axis, while the others feel around 0. Test the other axes by rotating the board and making them feel the pull of gravity.

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 a
ddress of 0x1C (SA0=0)
MMA8452Q accel(0x1D); // Initialize the MMA8452Q with an I2C a
ddress of 0x1D (SA0=1)
```

But if you've left the 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 da
ta
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 acceler
ation
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 & 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 – Loads of information about the MMA8452Q's electrical characteristics, registers, communication specifications, and more.
- MMA8452Q Breakout Schematic – PDF schematic of the MMA8452Q Breakout board.
- MMA8452Q Breakout Eagle Files – PCB design files for the MMA8452Q Breakout.
- MMA8452Q Breakout GitHub Repo – Design files and example code all related to the MMA8452Q.

Going Further

If you can't already tell, we love accelerometers! Check out all of the tutorials and projects we've created that feature an accelerometer and more.

- Are You Okay? Widget – Check out how Elecia embedded the MMA8452Q into a stuffed animal to create the Are-You-OK Widget.

Are You Okay? Shake the stuffed animal to send a tweet and let your buddy know.

- Getting Started with the RedBot – The MMA8452Q is used with the RedBot Kit to add bump detection.
- Das Blinken Top Hat – Add an accelerometer to your LED-covered top hat, so you can control the blink pattern based on how your head is tilted.
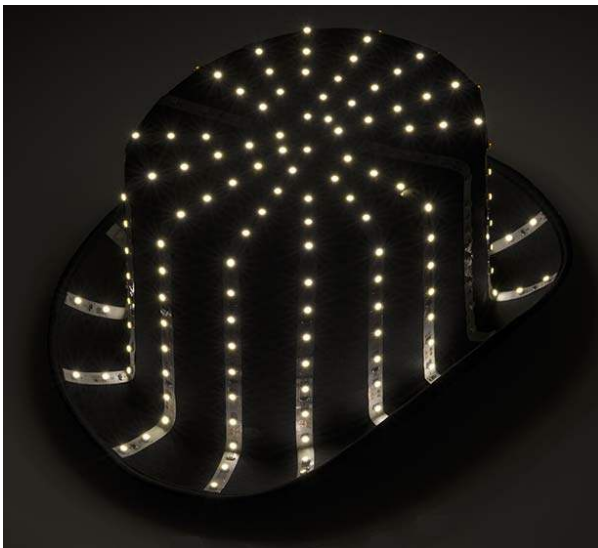
Das Blinken Top Hat! An accelerometer-controlled LED-covered dress hat.

- Simon Tilts – Accelerometers aren't the only sensors capable of sensing tilt. Check out the Simon Tilt game, which uses a really ingenious, optical method of detecting six different orientations.
- Dungeons and Dragons Dice Gauntlet – You can embed an accelerometer into your D&D bracers and roll virtual dice whenever you shake your arm!