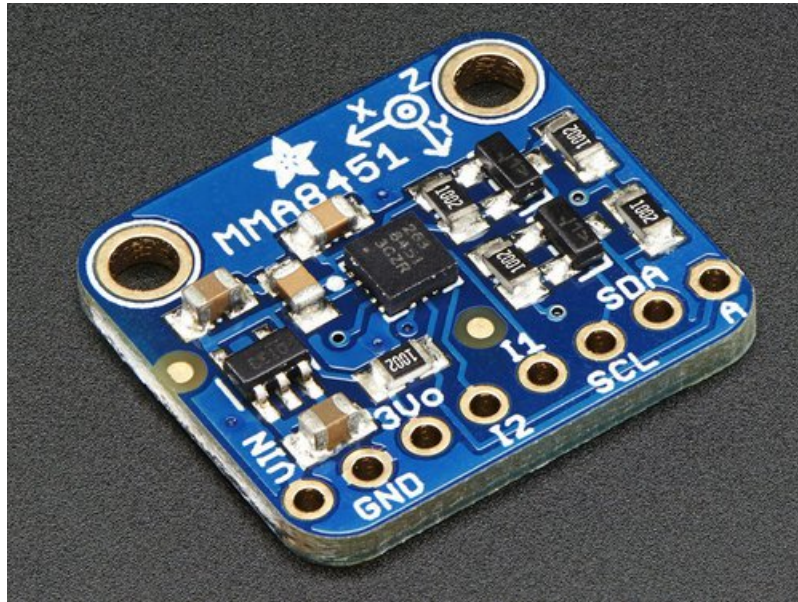




Adafruit MMA8451 Accelerometer Breakout

Created by lady ada

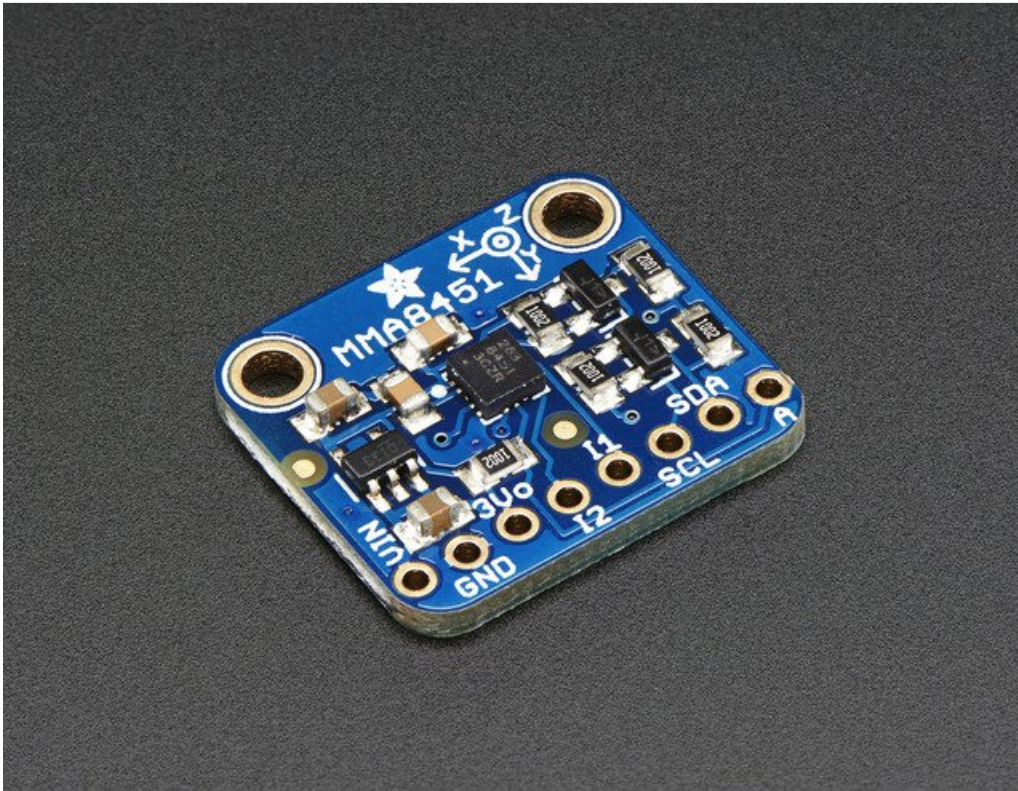


Last updated on 2018-02-06 04:55:03 PM UTC

Guide Contents

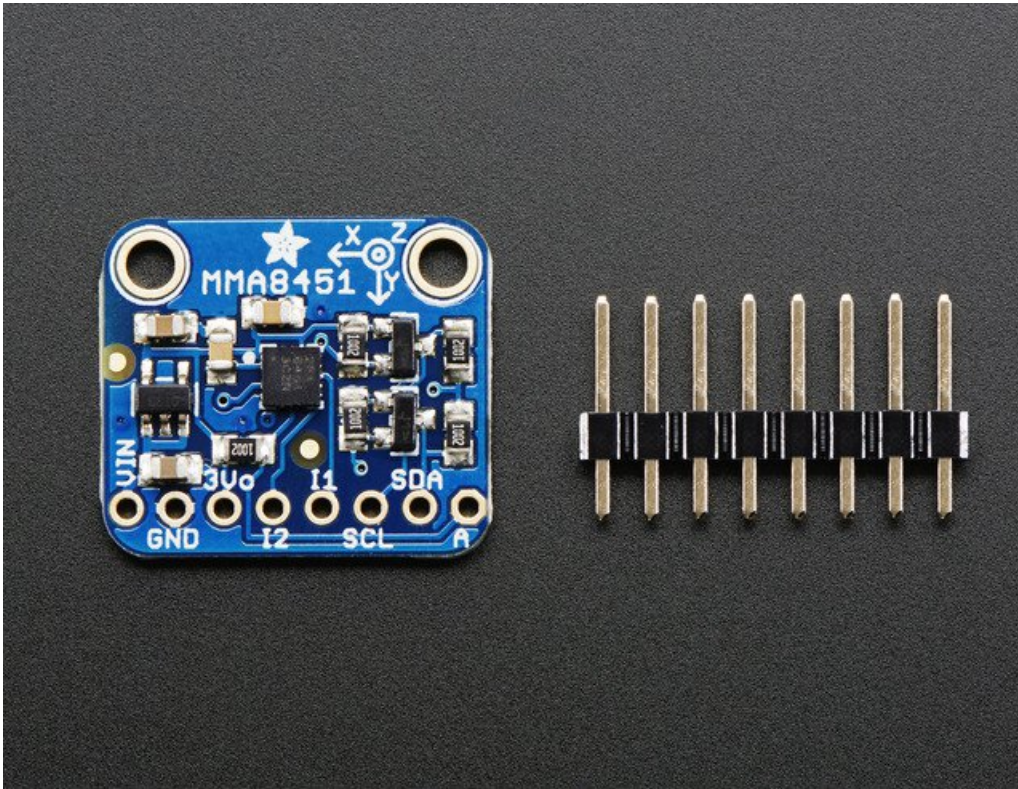
Guide Contents	2
Overview	3
Pinouts	6
Power Pins	6
I2C Pins	6
INT and ADDR Pins	6
Assembly	8
Prepare the header strip:	8
Add the breakout board:	9
And Solder!	9
Arduino Code	11
Download Adafruit_MMA8451	11
Download Adafruit_Sensor	12
Load Demo	12
Library Reference	14
Set & Get Range	14
Read Raw Count Data	14
Reading Normalized Adafruit_Sensor data	15
Read Orientation	15
CircuitPython Code	16
Usage	17
Downloads	20
Datasheet & Files	20
Schematics	20
Fabrication print	20

Overview

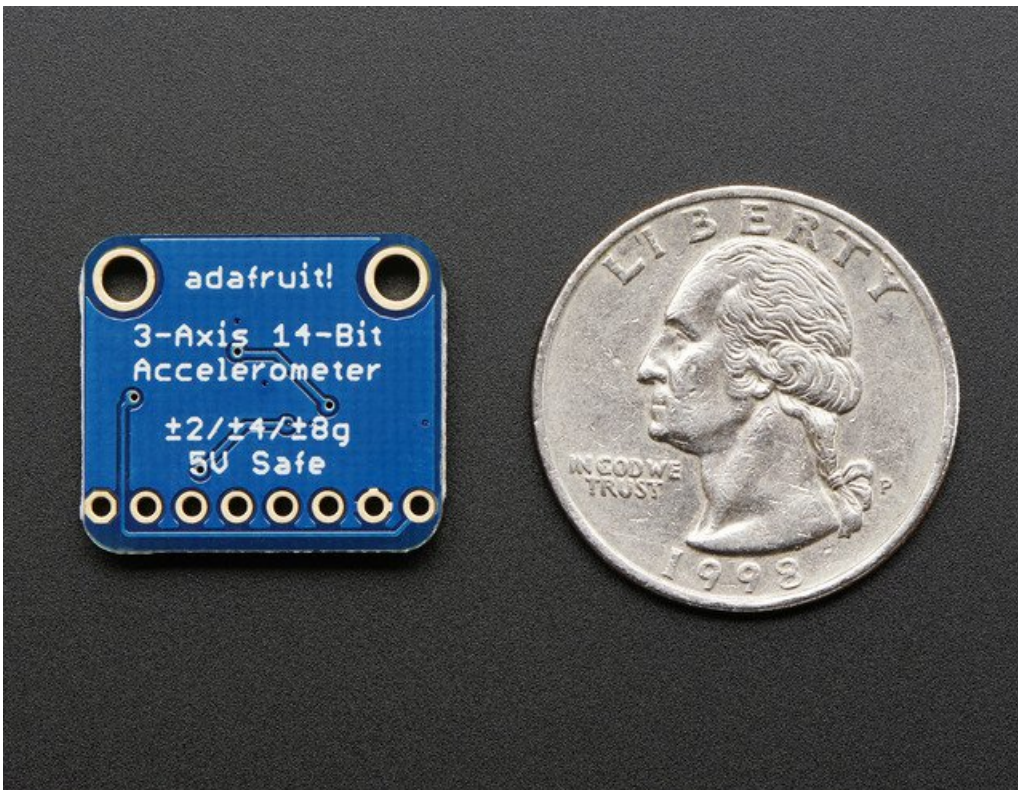


You can detect motion, tilt and basic orientation with a digital accelerometer - and the MMA8451 is a great accelerometer to start with. It's low cost, but high precision with 14-bit ADC. It has a wide usage range, from $\pm 2g$ up to $\pm 8g$ yet is easy to use with Arduino or another microcontroller

The MMA8451 is a miniature little accelerometer from Freescale, who are (by this point) masters at the accelerometer-design game. It's designed for use in phones, tablets, smart watches, and more, but works just as well in your Arduino project. Of the MMA8451/MMA8452/MMA8453 family, the MMA8451 is the most precise with a built in 14-bit ADC. The accelerometer also has built in tilt/orientation detection so i can tell you whether your project is being held in landscape or portrait mode, and whether it is tilted forward or back

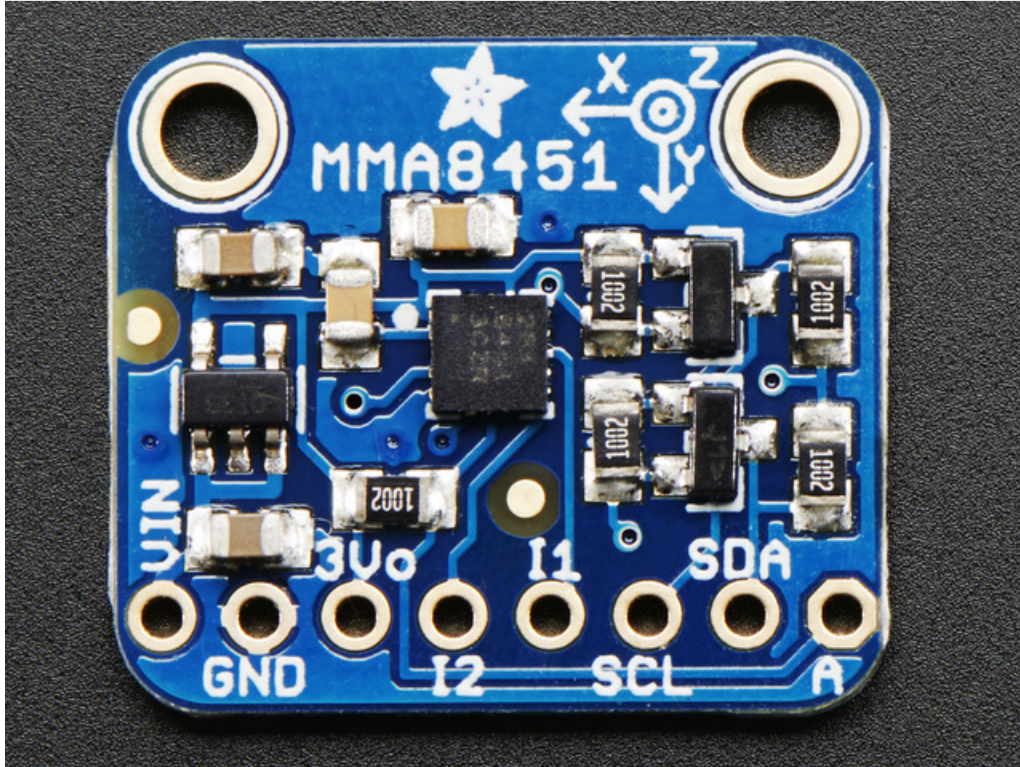


This sensor communicates over I2C so you can share it with a bunch of other sensors on the same two I2C pins. There's an address selection pin so you can have accelerometers share an I2C bus. Please note this chip requires repeated-start I2C support (in case you are looking to port this to another processor)



To get you going fast, we spun up a breakout board for this little guy. Since it's a 3V sensor, we add a low-dropout 3.3V regulator and level shifting circuitry on board. That means its perfectly safe for use with 3V or 5V power and logic.

Pinouts



The little chip in the middle of the PCB is the actual MMA8451 sensor that does all the motion sensing. We add all the extra components you need to get started, and 'break out' all the other pins you may want to connect to onto the PCB. For more details you can check out the schematics in the Downloads page.

Power Pins

The sensor on the breakout requires 3V power. Since many customers have 5V microcontrollers like Arduino, we tossed a 3.3V regulator on the board. Its ultra-low dropout so you can power it from 3.3V-5V just fine.

- **Vin** - this is the power pin. Since the chip uses 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- **3Vo** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like
- **GND** - common ground for power and logic

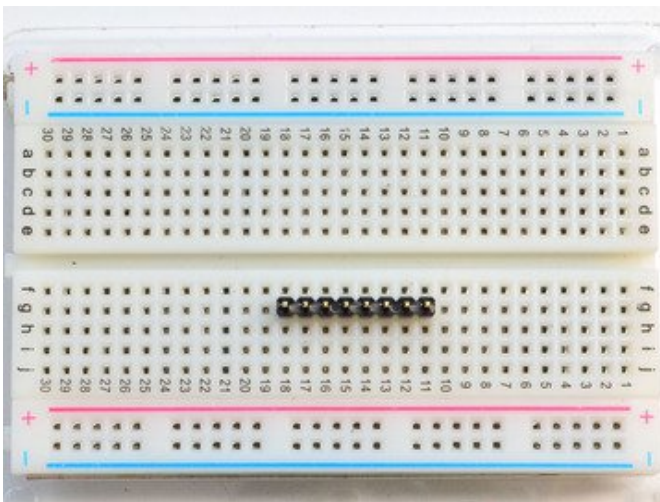
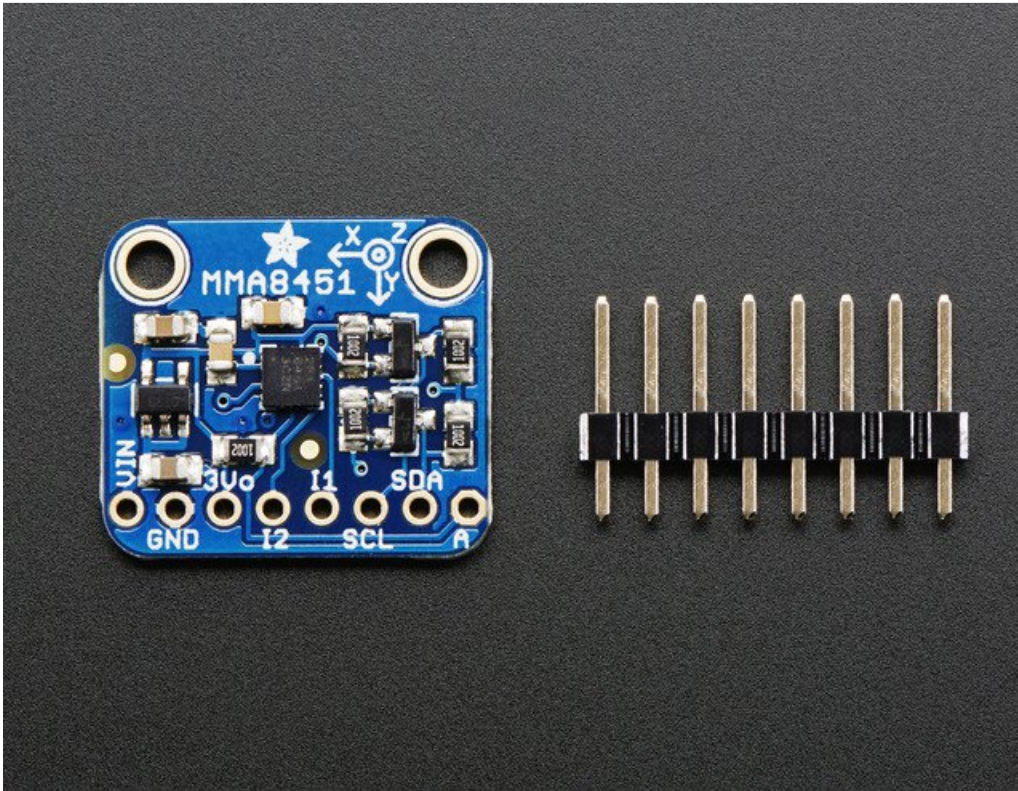
I2C Pins

- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line.
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line.

INT and ADDR Pins

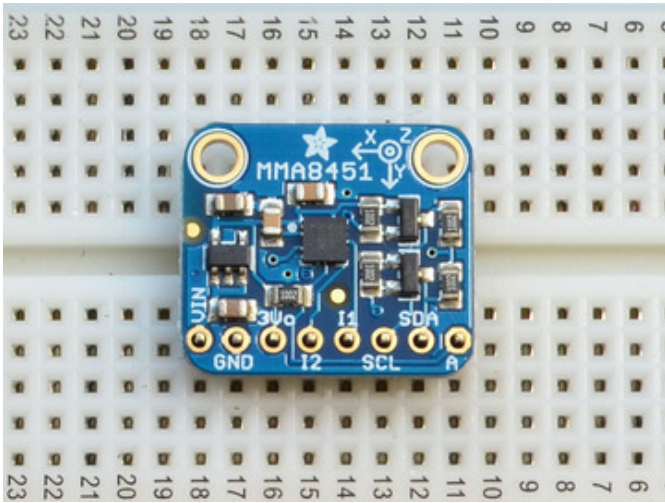
- **A** is the I2C Address select pin. By default this is pulled down to ground with a 10K resistor, for an I2C address of 0x1C. You can also connect it to the 3Vo pin for an address of 0x1D
- **I1** and **I2** are the Interrupt #1 and #2 signal pins. These pins are for more advanced usage, where you want to be alerted by the chip say when data is ready to read, or if it detects a large motion. We don't have direct support in the example Arduino library for these pins, so please check the datasheet for the I2C commands

Assembly



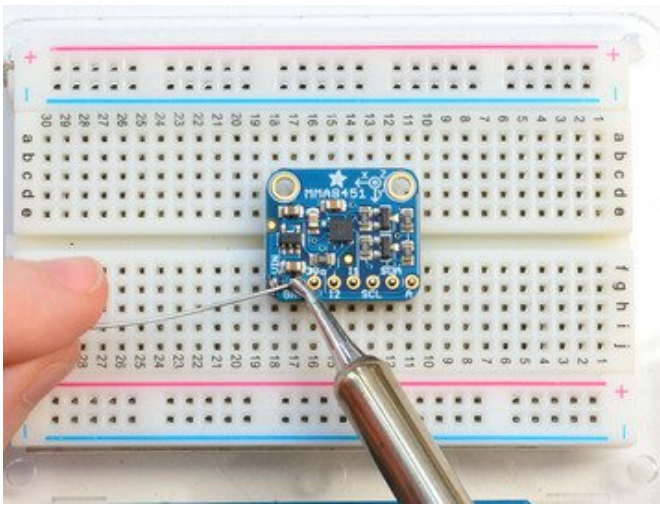
Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



Add the breakout board:

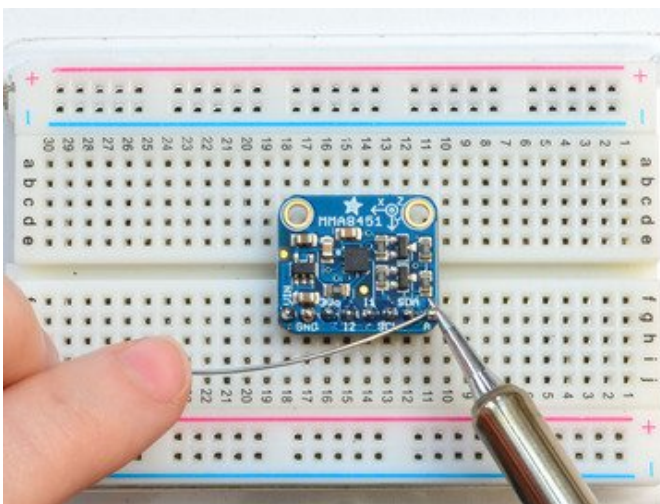
Place the breakout board over the pins so that the short pins poke through the breakout pads

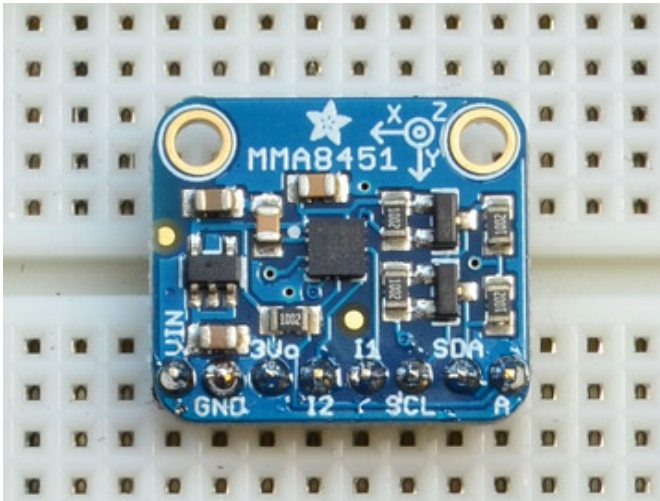


And Solder!

Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](https://adafruit.it/aTk) (<https://adafruit.it/aTk>)).

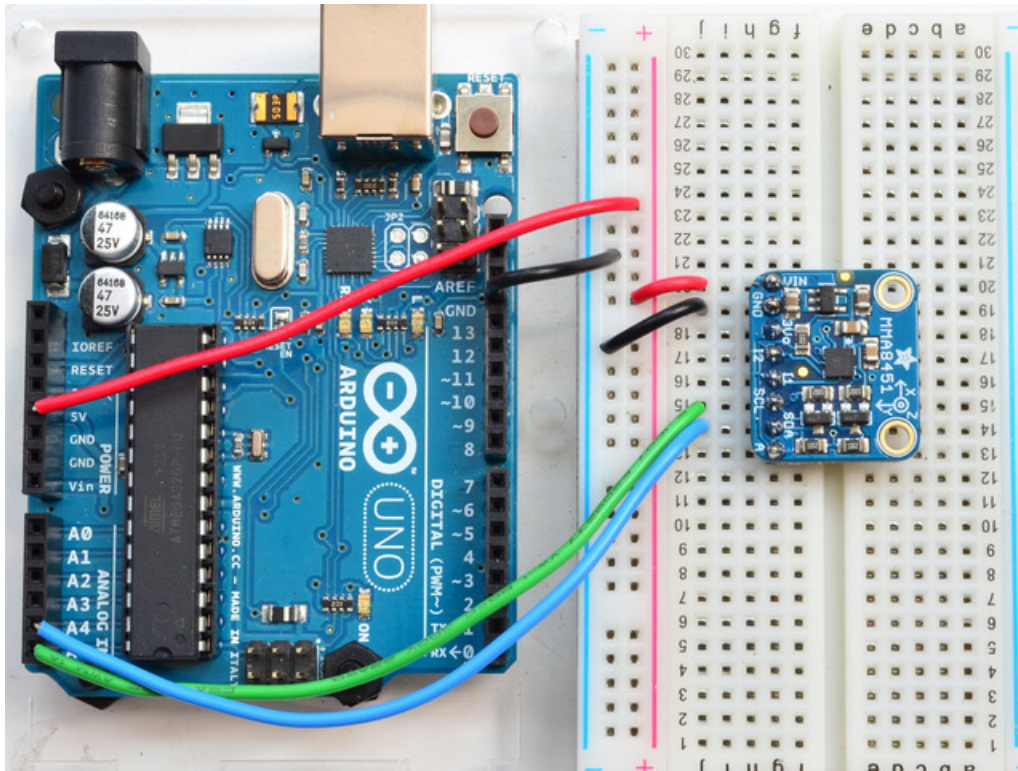




You're done! Check your solder joints visually and continue onto the next steps

Arduino Code

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!



- 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

Download the Adafruit MMA8451 Library

<https://adafru.it/dLK>

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 **arduinofolder/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](#) or just click the button below.

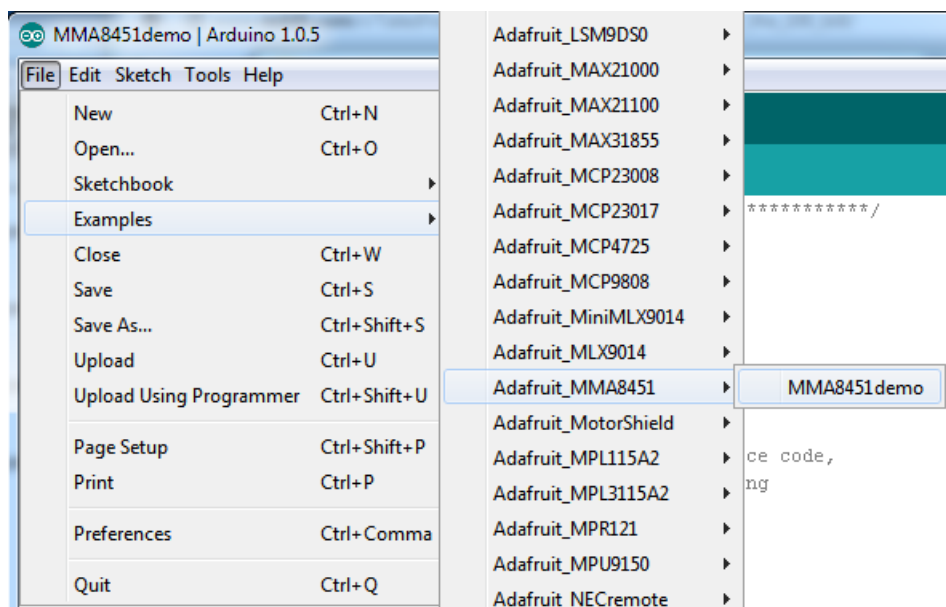
Download Adafruit_Sensor Library

<https://adafru.it/cMO>

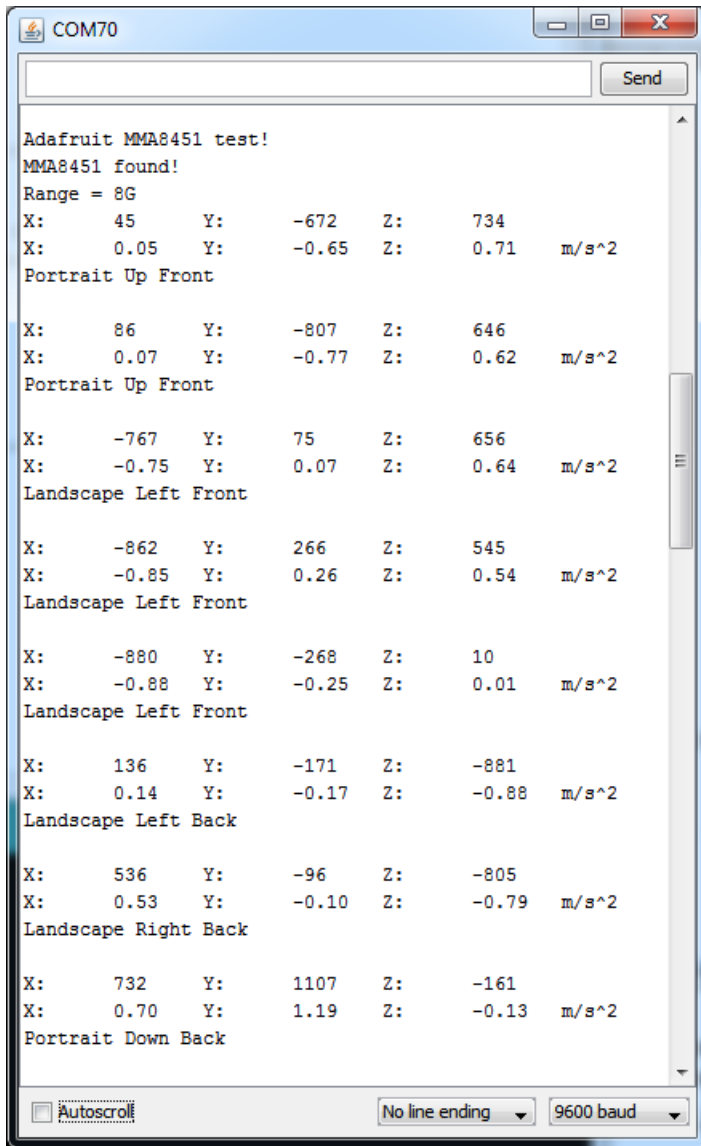
Install like you did with Adafruit_MMA8451

Load Demo

Open up **File->Examples->Adafruit_MMA8451->MMA8451demo** and upload to your Arduino wired up to the sensor



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



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: 9.8 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. (Note that the screenshot above has the m/s^2 divided by 10, you can ignore that typo :)

Example for line 3:

```
Portrait Up Front
```

This is the output of the orientation 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:

```
Adafruit_MMA8451 mma = Adafruit_MMA8451();
```

There are no pins to set since you must use the I2C bus!

Then initialize the sensor with:

```
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:

```
if (! mma.begin()) {  
  Serial.println("Couldnt start")  
  while (1);  
}  
Serial.println("MMA8451 found!");
```

Set & Get Range

You can set the accelerometer max range to $\pm 2g$, $\pm 4g$ or $\pm 8g$ with

```
mma.setRange(MMA8451_RANGE_2_G);  
mma.setRange(MMA8451_RANGE_4_G);  
mma.setRange(MMA8451_RANGE_8_G);
```

And read what the current range is with

```
mma.getRange()
```

Which returns 1 for $\pm 2g$, 2 for $\pm 4g$ and 3 for $\pm 8g$

Read Raw Count Data

You can read the raw counts data with

```
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

```
sensors_event_t event;
```

Then read the event whenever you want

```
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

```
mma.getOrientation();
```

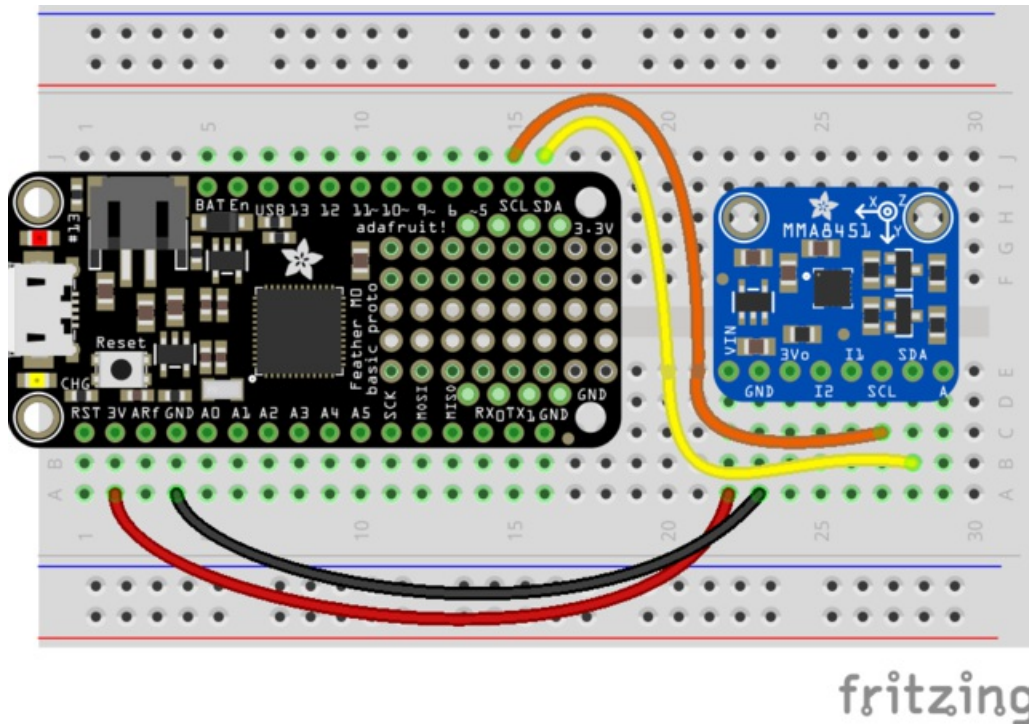
The return value ranges from 0 to 7

- 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

CircuitPython Code

It's easy to use the MMA8451 sensor with CircuitPython and the [Adafruit CircuitPython MMA8451](#) module. This module allows you to easily write Python code that reads the acceleration and more from the sensor.

First wire up a MMA8451 to your board exactly as shown on the previous pages for Arduino using an I2C connection. Here's an example of wiring a Feather M0 to the sensor with I2C:



- Board 3V to sensor VIN
- Board GND to sensor GND
- Board SCL to sensor SCL
- Board SDA to sensor SDA

Next you'll need to install the [Adafruit CircuitPython MMA8451](#) library on your CircuitPython board.

First make sure you are running the [latest version of Adafruit CircuitPython](#) for your board.

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from [Adafruit's CircuitPython library bundle](#). Our introduction guide has [a great page on how to install the library bundle](#) for both express and non-express boards.

Remember for non-express boards like the, you'll need to manually install the necessary libraries from the bundle:

- `adafruit_mma8451.mpy`
- `adafruit_bus_device`

You can also download the `adafruit_mma8451.mpy` from [its releases page on Github](#).

Before continuing make sure your board's lib folder or root filesystem has the `adafruit_mma8451.mpy`, and `adafruit_bus_device` files and folders copied over.

Next [connect to the board's serial REPL](#) so you are at the CircuitPython >>> prompt.

Usage

To demonstrate the usage of the sensor we'll initialize it and read the acceleration from the board's Python REPL. Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import board
import busio
import adafruit_mma8451
i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_mma8451.MMA8451(i2c)
```

Remember if you're using a board that doesn't support hardware I2C (like the ESP8266) you need to use the **bitbangio** module instead:

```
import board
import bitbangio
import adafruit_mma8451
i2c = bitbangio.I2C(board.SCL, board.SDA)
sensor = adafruit_mma8451.MMA8451(i2c)
```

Now you're ready to read values from the sensor using any of these properties:

- **acceleration** - This returns a 3-tuple of X, Y, Z acceleration values in meters per second squared (i.e. 9.8m/s² is the force of gravity on the surface of the earth).
- **orientation** - This is a value the MMA8451 calculates to help you understand what orientation the sensor is in, kind of like how a smartphone detects if its landscape or portrait orientation. This will return one of the following values:
 - **adafruit_mma8451.PL_PUF**: Portrait, up, front
 - **adafruit_mma8451.PL_PUB**: Portrait, up, back
 - **adafruit_mma8451.PL_PDF**: Portrait, down, front
 - **adafruit_mma8451.PL_PDB**: Portrait, down, back
 - **adafruit_mma8451.PL_LRF**: Landscape, right, front
 - **adafruit_mma8451.PL_LRB**: Landscape, right, back
 - **adafruit_mma8451.PL_LLF**: Landscape, left, front
 - **adafruit_mma8451.PL_LLB**: Landscape, left, back

```
x, y, z = sensor.acceleration
print('Acceleration: x={0:0.3f} m/s^2 y={1:0.3f} m/s^2 z={2:0.3f} m/s^2'.format(x, y, z))
orientation = sensor.orientation
print('Orientation: {0}'.format(orientation))
```

```
>>> x, y, z = sensor.acceleration
>>> print('Acceleration: x={0:0.3f}m/s^2 y={1:0.3f}m/s^2 z={2:0.3f}m/s^2'.format(x, y, z))
Acceleration: x=-1.475m/s^2 y=-1.820m/s^2 z=9.357m/s^2
>>> orientation = sensor.orientation
>>> print('Orientation: {0}'.format(orientation))
Orientation: 0
>>>
```

In addition there are a few properties you can read and write to change the behavior of the sensor:

- **range** - The range of the accelerometer measurements. This must be a value of:
 - `adafruit_mma8451.RANGE_2G`: +/- 2G range
 - `adafruit_mma8451.RANGE_4G`: +/- 4G range (the default)
 - `adafruit_mma8451.RANGE_8G`: +/- 8G range
- **data_rate** - The rate at which the sensor measures acceleration data. This must be a value of:
 - `adafruit_mma8451.DATARATE_800HZ`: 800hz
 - `adafruit_mma8451.DATARATE_400HZ`: 400hz
 - `adafruit_mma8451.DATARATE_200HZ`: 200hz
 - `adafruit_mma8451.DATARATE_100HZ`: 100hz
 - `adafruit_mma8451.DATARATE_50HZ`: 50hz
 - `adafruit_mma8451.DATARATE_12_5HZ`: 12.5hz
 - `adafruit_mma8451.DATARATE_6_25HZ`: 6.25hz
 - `adafruit_mma8451.DATARATE_1_56HZ`: 1.56hz

```
sensor.range = adafruit_mma8451.RANGE_8G
sensor.data_rate = adafruit_mma8451.DATARATE_400HZ
```

```
>>> sensor.range = adafruit_mma8451.RANGE_8G
>>> sensor.data_rate = adafruit_mma8451.DATARATE_400HZ
>>> x, y, z = sensor.acceleration
>>> print('Acceleration: x={0:0.3f}m/s^2 y={1:0.3f}m/s^2 z={2:0.3f}m/s^2'.format(x, y, z))
Acceleration: x=-1.456m/s^2 y=-1.810m/s^2 z=9.385m/s^2
>>> orientation = sensor.orientation
>>> print('Orientation: {0}'.format(orientation))
Orientation: 0
>>>
```

That's all there is to using the MMA8451 with CircuitPython!

The following is a complete example that will print the orientation and acceleration of the sensor every second. Save this as **main.py** on your board and open the REPL to see the output. Make sure to change the code to use **bitbangio** if necessary for your board!

```
# Simple demo of reading the MMA8451 orientation every second.
# Author: Tony DiCola
import time

import board
import busio

import adafruit_mma8451

# Initialize I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)

# Initialize MMA8451 module.
sensor = adafruit_mma8451.MMA8451(i2c)
# Optionally change the address if it's not the default:
#sensor = adafruit_mma8451.MMA8451(i2c, address=0x1C)

# Optionally change the range from its default of +/-4G:
#sensor.range = adafruit_mma8451.RANGE_2G # +/- 2G
#sensor.range = adafruit_mma8451.RANGE_4G # +/- 4G (default)
#sensor.range = adafruit_mma8451.RANGE_8G # +/- 8G
```

```

# Optionally change the data rate from its default of 800hz:
#sensor.data_rate = adafruit_mma8451.DATARATE_800HZ # 800Hz (default)
#sensor.data_rate = adafruit_mma8451.DATARATE_400HZ # 400Hz
#sensor.data_rate = adafruit_mma8451.DATARATE_200HZ # 200Hz
#sensor.data_rate = adafruit_mma8451.DATARATE_100HZ # 100Hz
#sensor.data_rate = adafruit_mma8451.DATARATE_50HZ # 50Hz
#sensor.data_rate = adafruit_mma8451.DATARATE_12_5HZ # 12.5Hz
#sensor.data_rate = adafruit_mma8451.DATARATE_6_25HZ # 6.25Hz
#sensor.data_rate = adafruit_mma8451.DATARATE_1_56HZ # 1.56Hz

# Main loop to print the acceleration and orientation every second.
while True:
    x, y, z = sensor.acceleration
    print('Acceleration: x={0:0.3f}m/s^2 y={1:0.3f}m/s^2 z={2:0.3f}m/s^2'.format(x, y, z))
    orientation = sensor.orientation
    # Orientation is one of these values:
    # - PL_PUF: Portrait, up, front
    # - PL_PUB: Portrait, up, back
    # - PL_PDF: Portrait, down, front
    # - PL_PDB: Portrait, down, back
    # - PL_LRF: Landscape, right, front
    # - PL_LRB: Landscape, right, back
    # - PL_LLF: Landscape, left, front
    # - PL_LLБ: Landscape, left, back
    print('Orientation: ', end='')
    if orientation == adafruit_mma8451.PL_PUF:
        print('Portrait, up, front')
    elif orientation == adafruit_mma8451.PL_PUB:
        print('Portrait, up, back')
    elif orientation == adafruit_mma8451.PL_PDF:
        print('Portrait, down, front')
    elif orientation == adafruit_mma8451.PL_PDB:
        print('Portrait, down, back')
    elif orientation == adafruit_mma8451.PL_LRF:
        print('Landscape, right, front')
    elif orientation == adafruit_mma8451.PL_LRB:
        print('Landscape, right, back')
    elif orientation == adafruit_mma8451.PL_LLF:
        print('Landscape, left, front')
    elif orientation == adafruit_mma8451.PL_LLБ:
        print('Landscape, left, back')
    time.sleep(1.0)

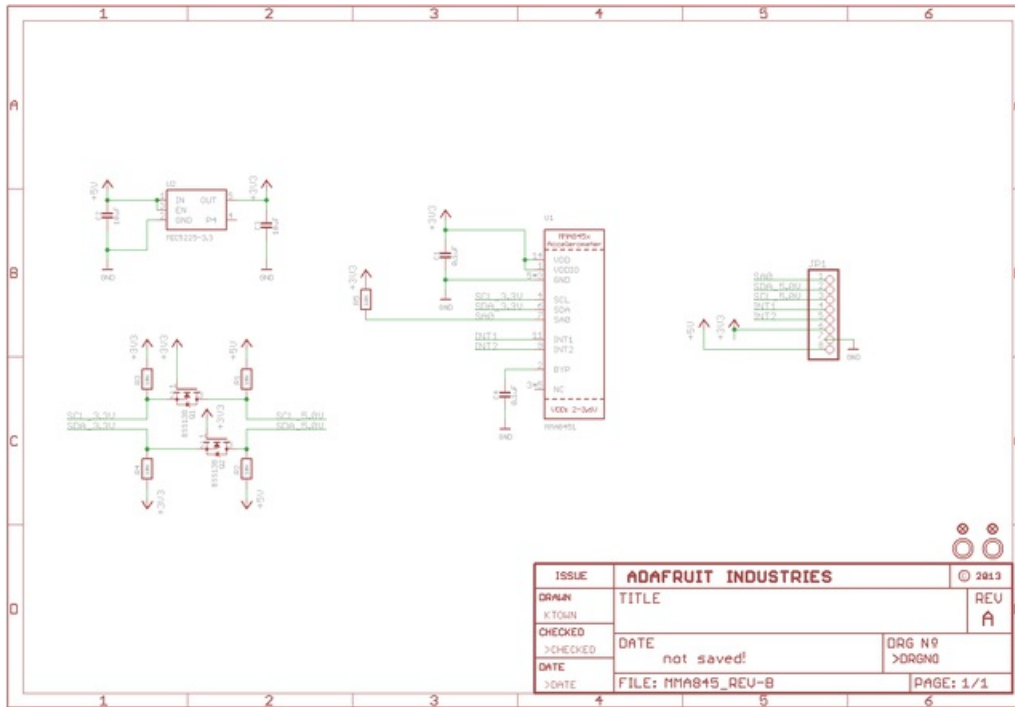
```

Downloads

Datasheet & Files

- [MMA8451-Q Datasheet](#)
- [Fritzing object in Adafruit Fritzing library](#)
- [EagleCAD PCB files on GitHub](#)

Schematics



Fabrication print

Dimensions are in Inches

