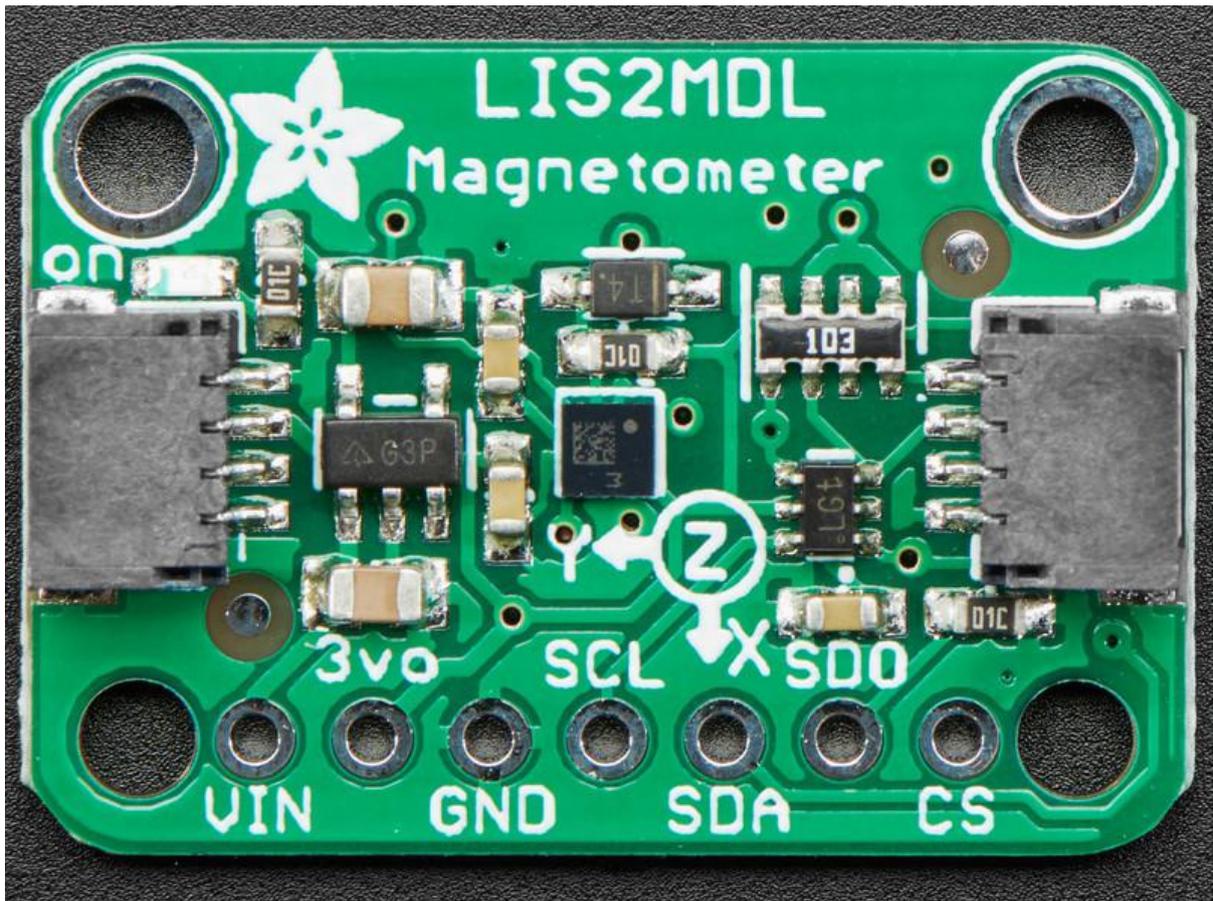




# Adafruit LIS2MDL Triple Axis Magnetometer

Created by Bryan Siepert



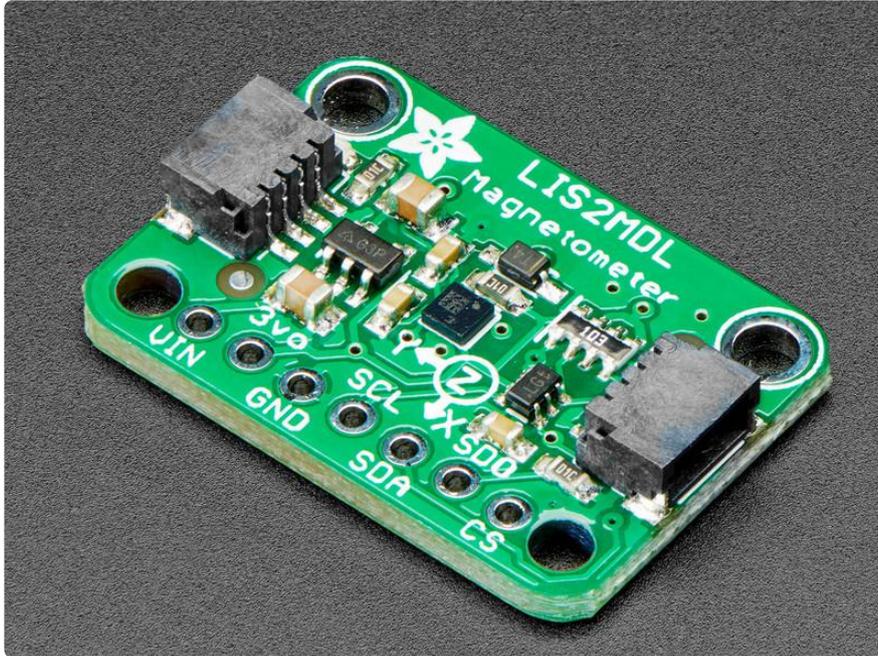
<https://learn.adafruit.com/adafruit-lis2mdl-triple-axis-magnetometer>

Last updated on 2022-12-01 03:50:29 PM EST

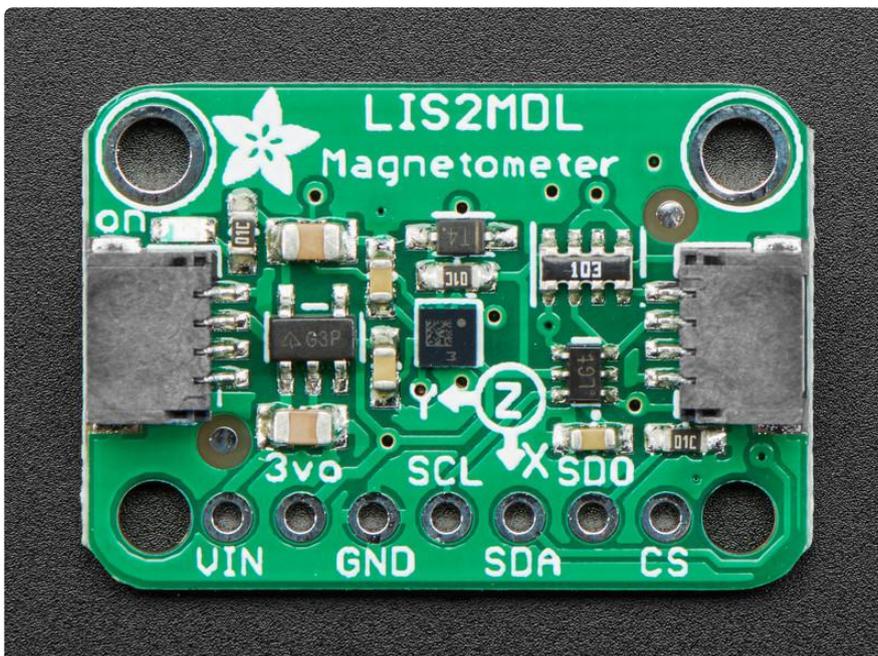
# Table of Contents

Overview	3
Pinouts	5
<ul style="list-style-type: none"><li>• Power Pins</li><li>• I2C Logic Pins</li><li>• SPI Logic pins:</li></ul>	
Arduino	6
<ul style="list-style-type: none"><li>• I2C Wiring</li><li>• SPI Wiring</li><li>• Library Installation</li><li>• Load Example</li><li>• Example Code</li></ul>	
Arduino Docs	10
Python & CircuitPython	10
<ul style="list-style-type: none"><li>• CircuitPython Microcontroller Wiring</li><li>• Python Computer Wiring</li><li>• CircuitPython Installation of LIS2MDL Library</li><li>• Python Installation of LIS2MDL Library</li><li>• CircuitPython &amp; Python Usage</li><li>• Example Code</li></ul>	
Python Docs	14
Downloads	14
<ul style="list-style-type: none"><li>• Files</li><li>• Schematic</li><li>• Fab Print</li></ul>	

# Overview

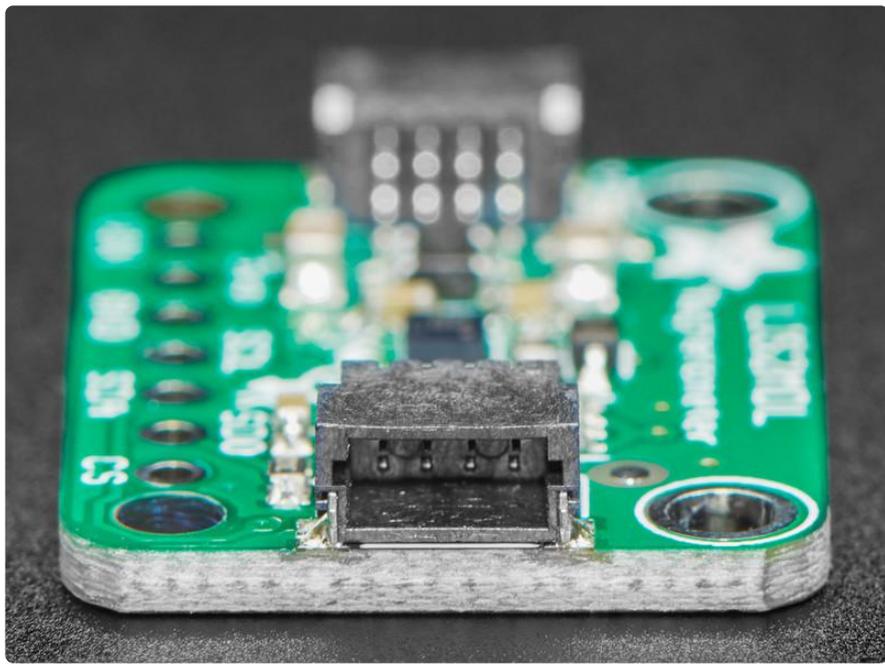


Sense the magnetic fields that surround us with this handy triple-axis magnetometer (compass) module. Magnetometers can sense where the strongest magnetic force is coming from, generally used to detect magnetic north, but can also be used for measuring magnetic fields. This sensor tends to be paired with a 6-DoF (degree of freedom) accelerometer/gyroscope to create a 9-DoF inertial measurement unit that can detect its orientation in real-space, thanks to Earth's stable magnetic field. It's a great match for any of our 6-DoF IMU sensors such as the [LSM6DSOX \(\)](#) or [LSM6DS3 3 \(\)](#).



We based this breakout on ST's LIS2MDL, a great general purpose magnetometer. This compact sensor uses I2C to communicate and its very easy to use. Simply download our library and connect the SCL pin to your I2C clock pin, and SDA pin to your I2C data pin and upload our test program to read out magnetic field data. If you'd like, you can also use SPI to receive data (we just happen to prefer I2C here)

This sensor can measure up to nearly +/-50 gauss (49.152 gauss to be specific, +/-4952 uTesla) which is quite a bit! It also has an adjustable data rate and can take measurements as slowly as 10Hz and as fast as 100Hz.

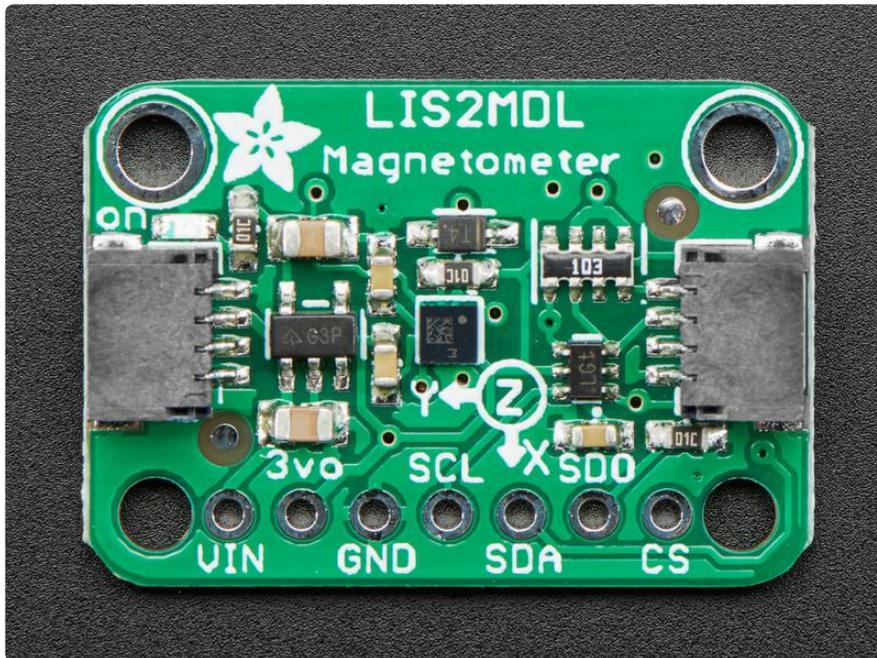


This adorable little magnetometer is quite capable but it is near-microscopic at 2mm square. To make things easier, we've put it on a breakout PCB along with support circuitry to let you use this little wonder with 3.3V (Feather/Raspberry Pi) or 5V (Arduino/ Metro328) logic levels.

Additionally since it speaks I2C, you can easily connect it up with two wires (plus power and ground!). We've even included [SparkFun qwiic \(\)](#) compatible [STEMMA QT \( \)](#) connectors for the I2C bus so you don't even need to solder! Just wire up to your favorite micro and [you can use our CircuitPython / Python \(\)](#) or [Arduino drivers to easily interface with the LIS2MDL \(\)](#) and get magnetic measurements ASAP.

---

# Pinouts



## Power Pins

- Vin - this is the power pin. Since the sensor 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 microcontroller 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 Logic Pins

- SCL - I2C clock pin, connect to your microcontroller's I2C clock line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
- SDA - I2C data pin, connect to your microcontroller's I2C data line. This pin is level shifted so you can use 3-5V logic, and there's a 10K pullup on this pin.
- [STEMMA QT \(\)](#) - These connectors allow you to connect to dev boards with STEMMA QT connectors or to other things with [various associated accessories \(\)](#)

## SPI Logic pins:

All pins going into the breakout have level shifting circuitry to make them 3-5V logic level safe. Use whatever logic level is on Vin!

- SCL - This is also the SPI Clock pin, it's an input to the chip
- SDA - this is also the Serial Data In / Microcontroller Out Sensor In pin, for data sent from your processor to the LIS2MDL
- SDO - this is the Serial Data Out / Microcontroller In Sensor Out pin, for data sent from the LIS2MDL to your processor.
- CS - this is the Chip Select pin, drop it low to start an SPI transaction. Its an input to the chip

If you want to connect multiple LIS2MDLs to one microcontroller, have them share the SDA, SDO and SCL pins. Then assign each one a unique CS pin.

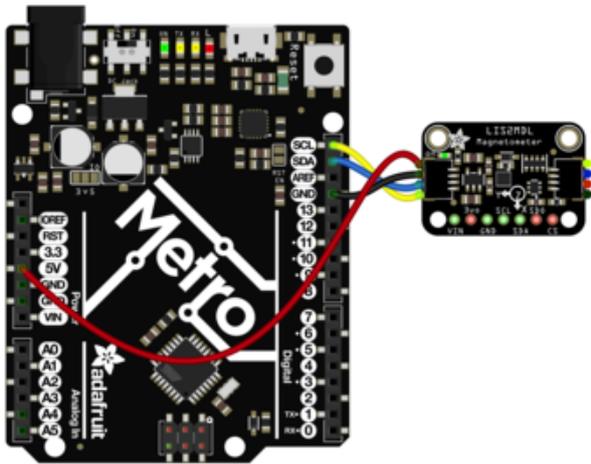
---

## Arduino

### I2C Wiring

Use this wiring if you want to connect via I2C interface:

The I2C address is 0x1E.

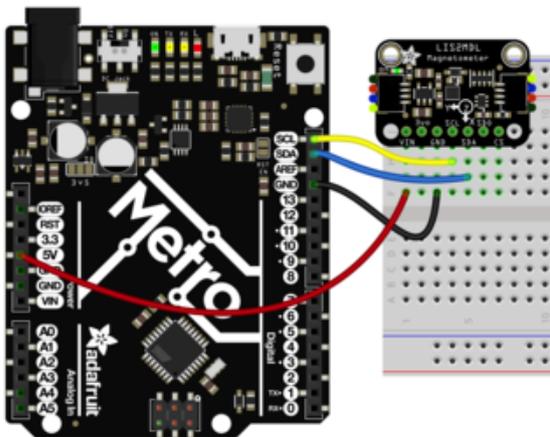


Connect board VIN (red wire) to Arduino 5V if you are running a 5V board Arduino (Uno, etc.). If your board is 3V, connect to that instead.

Connect board GND (black wire) to Arduino GND

Connect board SCL (yellow wire) to Arduino SCL

Connect board SDA (blue wire) to Arduino SDA



## SPI Wiring

Since this is a SPI-capable sensor, we can use hardware or 'software' SPI. To make wiring identical on all microcontrollers, we'll begin with 'software' SPI. The following pins should be used:

Connect Vin to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of.

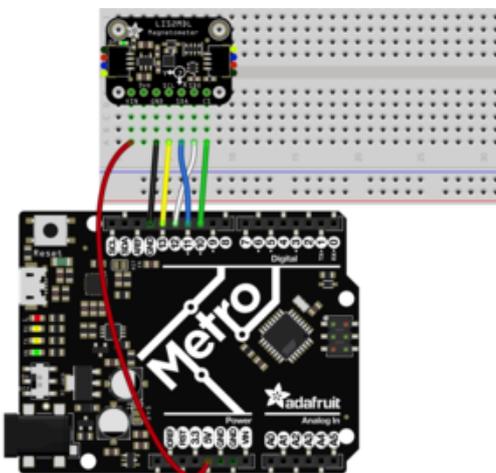
Connect GND to common power/data ground

Connect the SCL pin to Digital #13 but any pin can be used later

Connect the DO pin to Digital #12 but any pin can be used later

Connect the SDA pin to Digital #11 but any pin can be used later

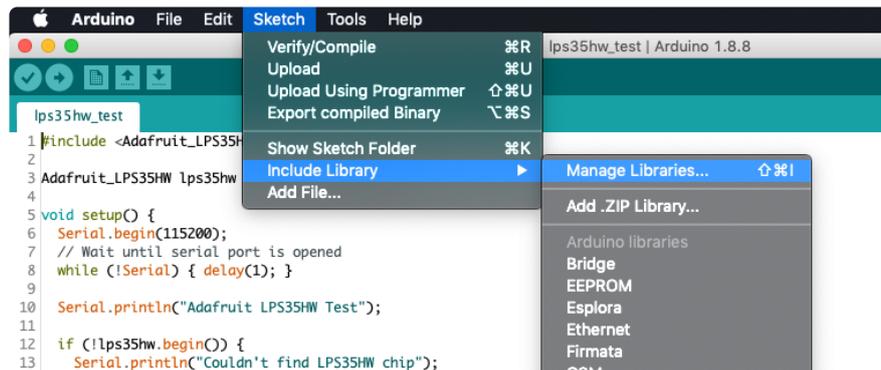
Connect the CS pin Digital #10 but any pin can be used later



Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to others.

## Library Installation

You can install the Adafruit LIS2MDL Library for Arduino using the Library Manager in the Arduino IDE.



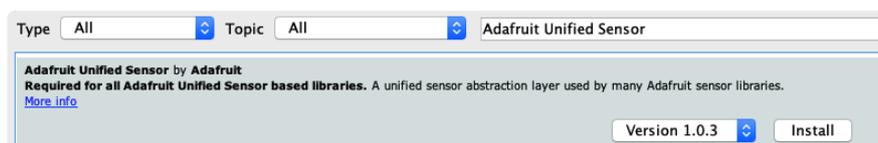
Click the Manage Libraries ... menu item, search for Adafruit LIS2MDL, and select the Adafruit LIS2MDL library:



Then follow the same process for the Adafruit BusIO library.



Finally follow the same process for the Adafruit Unified Sensor library:



## Load Example

Open up File -> Examples -> Adafruit LIS2MDL -> magsensor and upload to your Arduino which has been wired up to the sensor.

Depending on whether you are using I2C or SPI, change the pin names and comment or uncomment the following lines.

```
if (!lis2mdl.begin()) { // I2C mode
  //if (! lis2mdl.begin_SPI(LIS2MDL_CS)) { // hardware SPI mode
  //if (! lis2mdl.begin_SPI(LIS2MDL_CS, LIS2MDL_CLK, LIS2MDL_MISO, LIS2MDL_MOSI))
  { // soft SPI
```

Once you upload the code and open the Serial Monitor (Tools->Serial Monitor) at 115200 baud, you will see the current configuration printed, followed by magnetometer measurements, similar to this:

```
Magnetometer Test
-----
Sensor:      LIS2MDL
Driver Ver:  1
Unique ID:   12345
Max Value:   0.00 uT
Min Value:   0.00 uT
Resolution:  0.00 uT
-----
X: 17.40 Y: 65.55 Z: -29.40 uT
X: 16.35 Y: 65.25 Z: -30.00 uT
X: 16.65 Y: 65.85 Z: -29.40 uT
```

The sensor class in the magnetometer library reports X, Y and Z axis magnetometer readings directly in [micro-Teslas \(\)](#). The magsensor example code reads from the sensor and prints the micro-Tesla readings to the Serial Monitor.

In the absence of any strong local magnetic fields, the sensor readings should reflect the magnetic field of the earth (between 20 and 60 micro-Teslas). When the sensor is held level, by calculating the angle of the magnetic field with respect to the X and Y axis, the device can be used as a compass.

## Example Code

```
#include <Adafruit_LIS2MDL.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

/* Assign a unique ID to this sensor at the same time */
Adafruit_LIS2MDL lis2mdl = Adafruit_LIS2MDL(12345);
#define LIS2MDL_CLK 13
#define LIS2MDL_MISO 12
#define LIS2MDL_MOSI 11
#define LIS2MDL_CS 10

void setup(void) {
  Serial.begin(115200);
  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("LIS2MDL Magnetometer Test");
```

```

Serial.println("");

/* Enable auto-gain */
lis2mdl.enableAutoRange(true);

/* Initialise the sensor */
if (!lis2mdl.begin()) { // I2C mode
//if (! lis2mdl.begin_SPI(LIS2MDL_CS)) { // hardware SPI mode
//if (! lis2mdl.begin_SPI(LIS2MDL_CS, LIS2MDL_CLK, LIS2MDL_MISO, LIS2MDL_MOSI))
{ // soft SPI
  /* There was a problem detecting the LIS2MDL ... check your connections */
  Serial.println("Ooops, no LIS2MDL detected ... Check your wiring!");
  while (1) delay(10);
}

/* Display some basic information on this sensor */
lis2mdl.printSensorDetails();
}

void loop(void) {
  /* Get a new sensor event */
  sensors_event_t event;
  lis2mdl.getEvent(&event);

  /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
  Serial.print("X: ");
  Serial.print(event.magnetic.x);
  Serial.print(" ");
  Serial.print("Y: ");
  Serial.print(event.magnetic.y);
  Serial.print(" ");
  Serial.print("Z: ");
  Serial.print(event.magnetic.z);
  Serial.print(" ");
  Serial.println("uT");

  /* Note: You can also get the raw (non unified values) for */
  /* the last data sample as follows. The .getEvent call populates */
  /* the raw values used below. */
  // Serial.print("X Raw: "); Serial.print(lis2mdl.raw.x); Serial.print(" ");
  // Serial.print("Y Raw: "); Serial.print(lis2mdl.raw.y); Serial.print(" ");
  // Serial.print("Z Raw: "); Serial.print(lis2mdl.raw.z); Serial.println("");

  /* Delay before the next sample */
  delay(100);
}

```

---

## Arduino Docs

[Arduino Docs \(\)](#)

---

## Python & CircuitPython

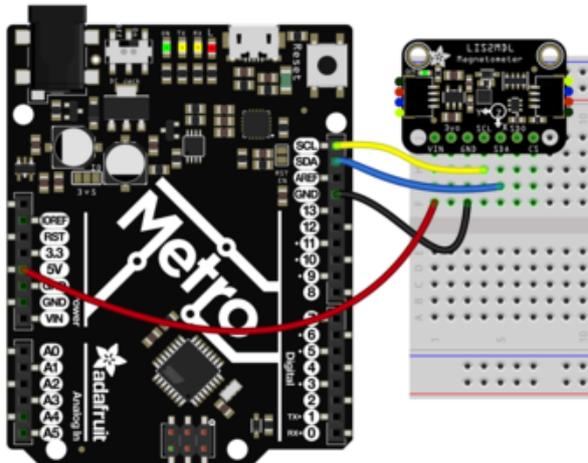
It's easy to use the LIS2MDL sensor with CircuitPython and the [Adafruit CircuitPython LIS2MDL \(\)](#) library. This library will allow you to easily write Python code that reads the magnetometer values from the sensor.

You can use this sensor with any CircuitPython microcontroller board or with a computer that has GPIO and Python [thanks to Adafruit\\_Blinka, our CircuitPython-for-Python compatibility library](#) ().

## CircuitPython Microcontroller Wiring

First wire up a LIS2MDL breakout to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C:

[sensors\\_d\\_python\\_feather\\_wiring\\_qt.png](#)

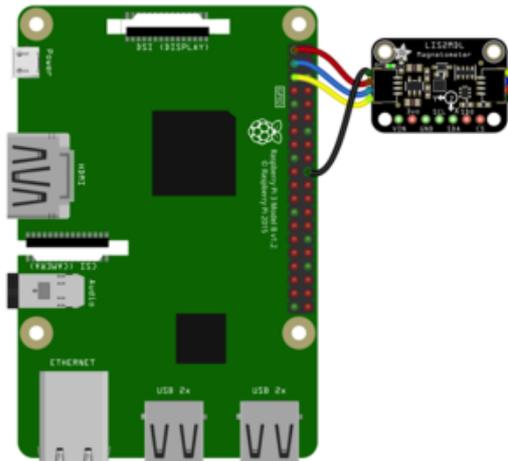


- Board 3V to sensor VIN (red wire)
- Board GND to sensor GND (black wire)
- Board SCL to sensor SCL (yellow wire)
- Board SDA to sensor SDA (blue wire)

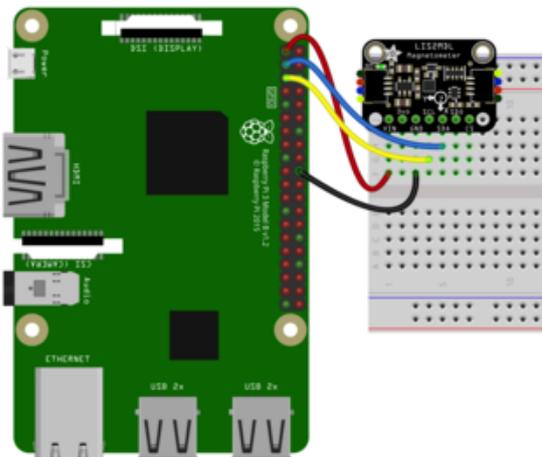
## Python Computer Wiring

Since there's dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](#) ().

Here's the Raspberry Pi wired to the sensor using I2C:



Pi 3V to sensor VCC (red wire)  
Pi GND to sensor GND (black wire)  
Pi SCL to sensor SCL (yellow wire)  
Pi SDA to sensor SDA (blue wire)



## CircuitPython Installation of LIS2MDL Library

You'll need to install the [Adafruit CircuitPython LIS2MDL \(\)](#) 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 CircuitPython starter guide has [a great page on how to install the library bundle \(\)](#).

For non-Express boards like the Trinket M0 or Gemma M0, you'll need to manually install the necessary libraries from the bundle:

- adafruit\_lis2mdl.mpy
- adafruit\_bus\_device
- adafruit\_register

Before continuing make sure your board's lib folder has the adafruit\_lis2mdl.mpy, adafruit\_bus\_device, and adafruit\_register files and folders copied over.

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

## Python Installation of LIS2MDL Library

You'll need to install the Adafruit\_Blinka library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. [Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready](#) ()!

Once that's done, from your command line, run the following command:

- `sudo pip3 install adafruit-circuitpython-lis2mdl`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported!

## CircuitPython & Python Usage

To demonstrate the usage of the sensor, we'll initialize it and read the magnetometer measurements from the board's Python REPL.

Run the following code to import the necessary modules and initialize the I2C connection with the sensor:

```
import time
import board
import adafruit_lis2mdl

i2c = board.I2C()
sensor = adafruit_lis2mdl.LIS2MDL(i2c)
```

Now you're ready to read values from the magnetometer using the magnetic property which returns a 3-tuple of the X, Y, and Z magnetometer readings in micro-Teslas (uT)

```
mag_x, mag_y, mag_z = sensor.magnetic
print('X:{0:10.2f}, Y:{1:10.2f}, Z:{2:10.2f} uT'.format(mag_x, mag_y, mag_z))
```

## Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

""" Display magnetometer data once per second """

import time
import board
import adafruit_lis2mdl

i2c = board.I2C() # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C() # For using the built-in STEMMMA QT connector on a
microcontroller
sensor = adafruit_lis2mdl.LIS2MDL(i2c)

while True:
    mag_x, mag_y, mag_z = sensor.magnetic

    print("X:{0:10.2f}, Y:{1:10.2f}, Z:{2:10.2f} uT".format(mag_x, mag_y, mag_z))
    print("")
    time.sleep(1.0)
```

---

## Python Docs

[Python Docs \(\)](#)

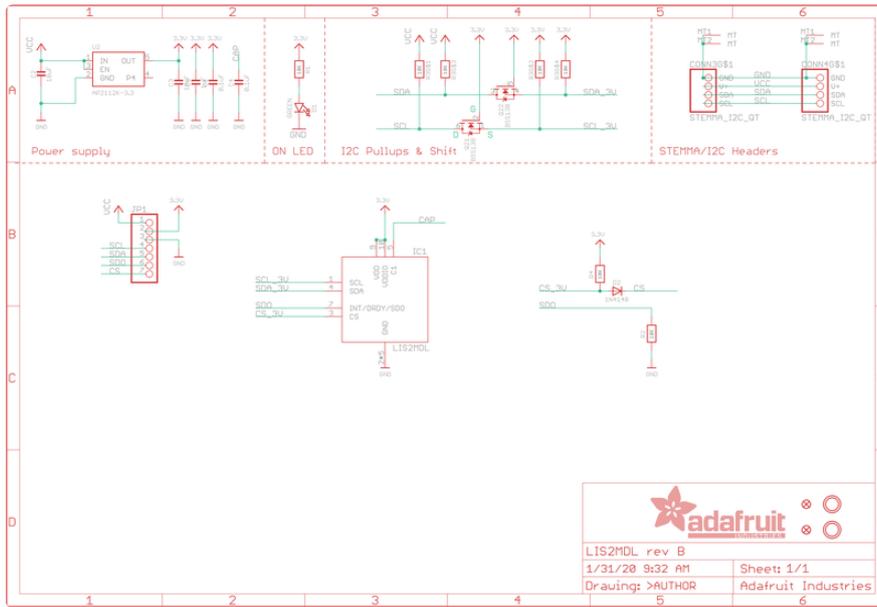
---

## Downloads

### Files

- [LI23MDL Datasheet \(\)](#)
- [EagleCAD files on GitHub \(\)](#)
- [Fritzing object in Adafruit Fritzing Library \(\)](#)

# Schematic



# Fab Print

