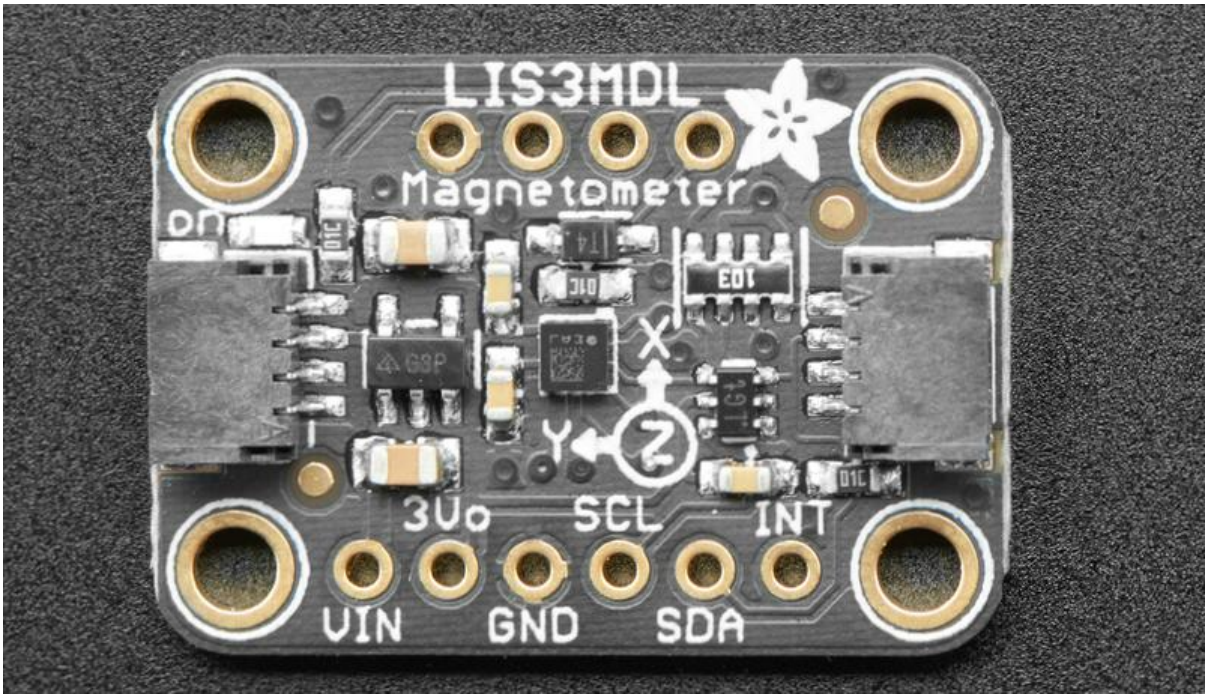
LIS3MDL Triple-axis Magnetometer

Created by Bryan Siepert

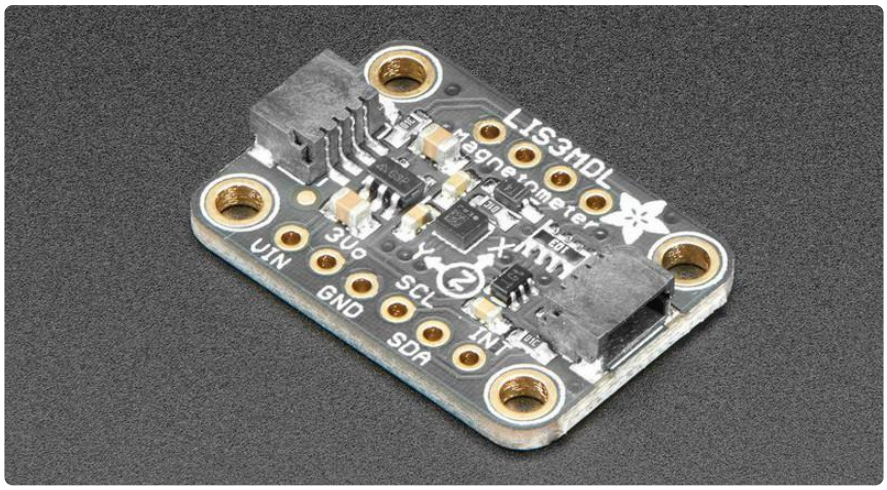


https://learn.adafruit.com/lis3mdl-triple-axis-magnetometer

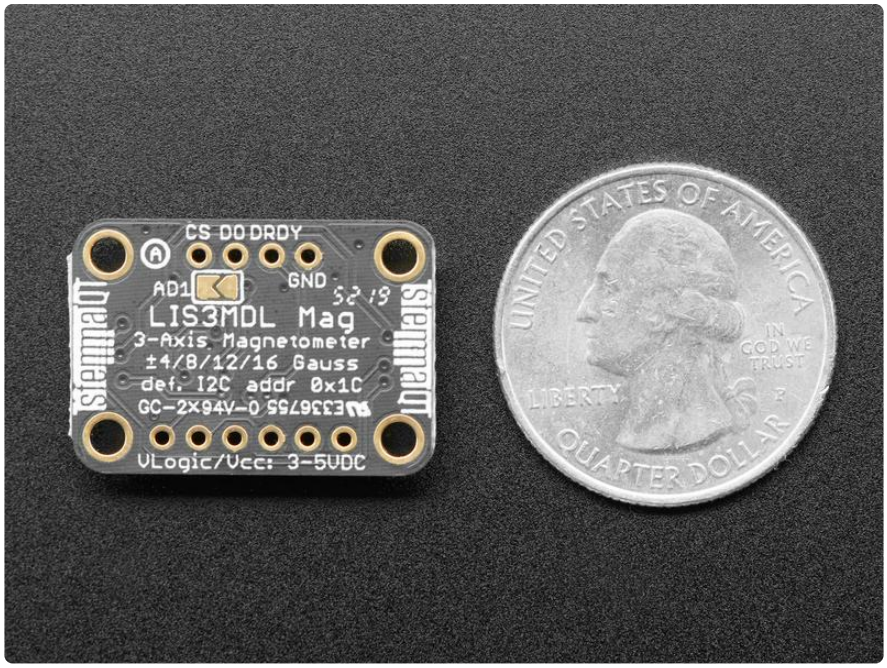
Last updated on 2022-12-01 03:48:59 PM EST

Table of Contents

Overview



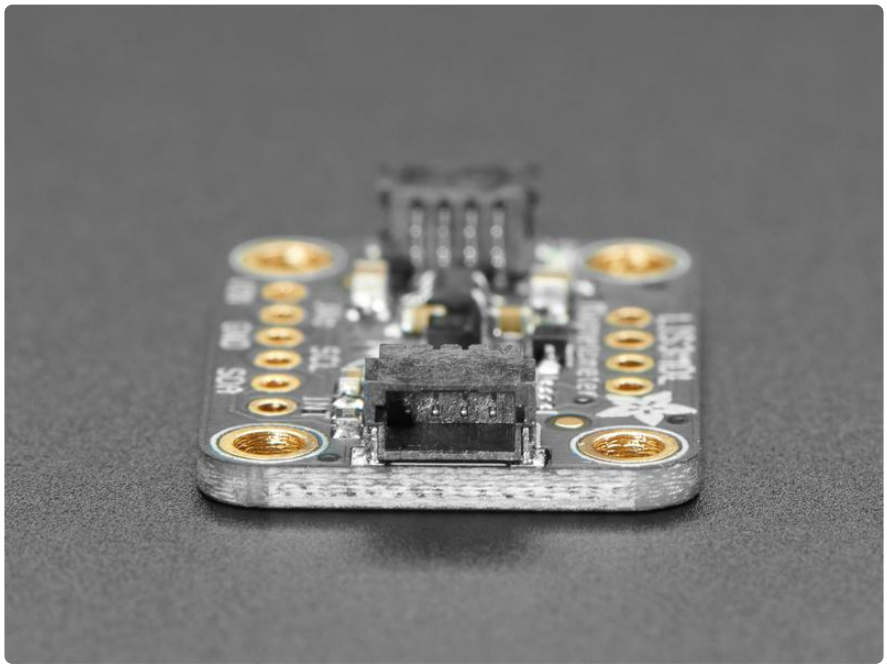
Sense the magnetic fields that surround us with this handy triple-axis magnetometer (compass) module. Magnetometers can sense where the strongest magnetic force is coming from, generally used to detect magnetic north, but can also be used for measuring magnetic fields. This sensor tends to be paired 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 LSM6DS33 ().



We based this breakout on ST's LIS3MDL, 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)

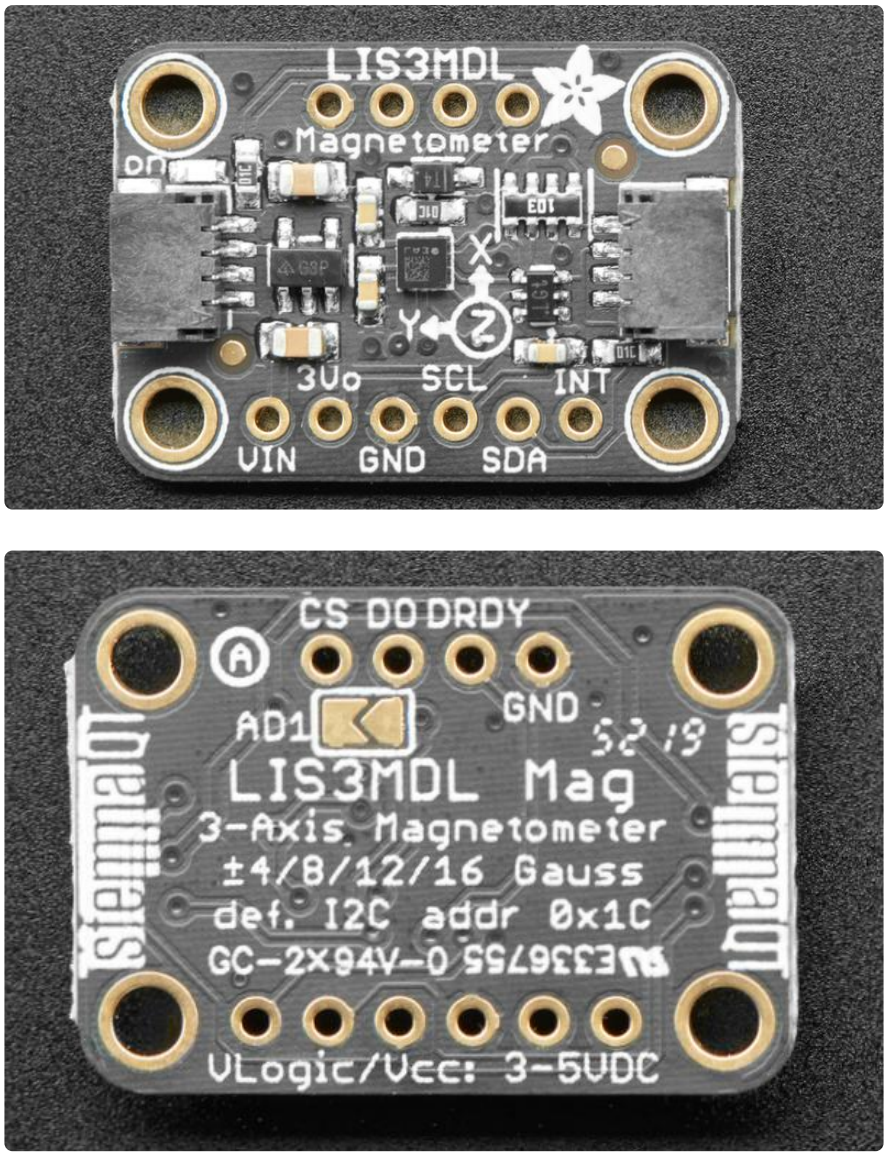
This sensor can sense ranges from +-4 gauss (+- 400 uTesla) up to +-16 gauss (+-16uTesla). For ultra high precision, 155 Hz update rate is recommended - but if you don't mind a little loss of precision, the sensor can output at 1000 Hz.



To make life easier so you can focus on your important work, we've taken the LIS3MDL and put it onto 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 LIS3MDL () 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 make I2C connections to dev boards with STEMMA QT connectors or to other things with various associated accessories ()
- DO/AD1 Jumper - I2C Address pin. Pulling this pin high or bridging the solder jumper on the back will change the I2C address from 0x1C to 0x1E. See this guide for how to check and verify the set address ().

The D0/AD1 pin is pulled to ground with a 10k resistor. For a few samples of the LIS3MDL, this may cause erratic setting of the I2C address. If you encounter the problem, tie the pin directly to ground or to 3.3V at the jumper on the back.

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 LIS3MDL
- DO - this is the Serial Data Out / Microcontroller In Sensor Out pin, for data sent from the LIS3MDL 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 LIS3MDL's to one microcontroller, have them share the SDA, SDO and SCL pins. Then assign each one a unique CS pin.

Other Pins

- INT -This is the interrupt pin. You can setup the LIS3MDL to pull this low when certain conditions are met such as a value exceeding a threshold. Consult the datasheet () for usage

- DRDY - The data ready pin. When measurement data is available the sensor will pull this pin low

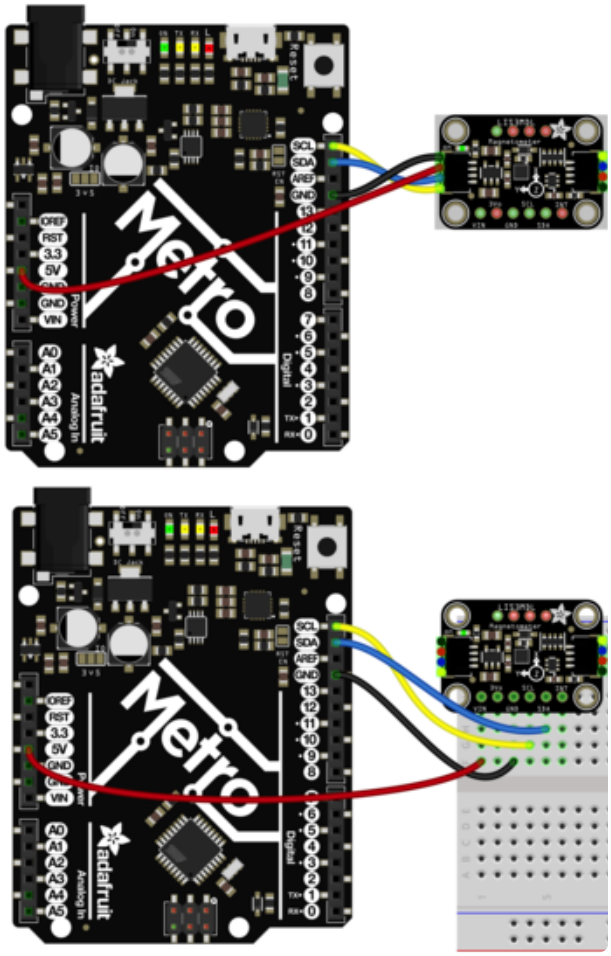
---

Arduino

I2C Wiring

Use this wiring if you want to connect via I2C interface

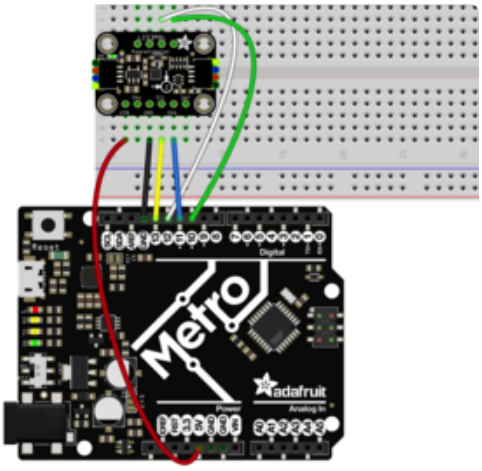
By default, the i2c address is 0x1C. If you short the AD1 solder jumper on the back of the board or add a jumper from DO to 3.3V the address will change to 0x1E



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:

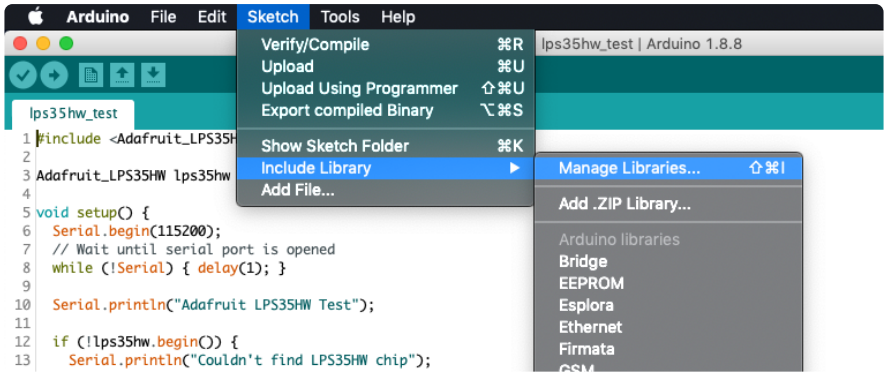


Connect Vin to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of
Connect GND to common power/data ground
Connect the SCK pin to Digital #13 but any pin can be used later
Connect the DO pin to Digital #12 but any pin can be used later
Connect the SDA pin to Digital #11 but any pin can be used later
Connect the CS pin Digital #10 but any pin can be used later

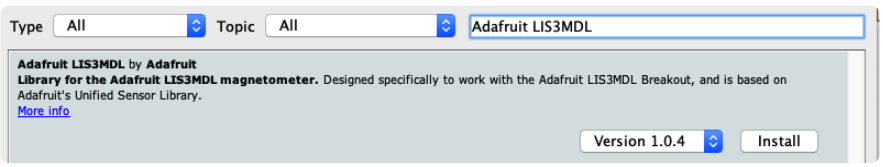
Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to others.

Library Installation

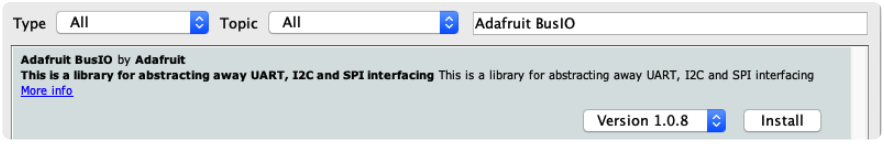
You can install the Adafruit LIS3MDL Library for Arduino using the Library Manager in the Arduino IDE.



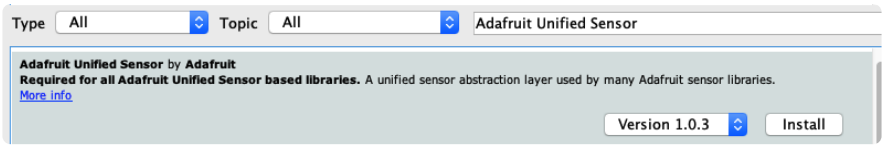
Click the Manage Libraries ... menu item, search for Adafruit LIS3MDL, and select the Adafruit LIS3MDL library:



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 LIS3MDL -> lis3mdl_demo and upload to your Arduino 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 (! lis3mdl.begin_I2C()) {
  //if (! lis3mdl.begin_SPI(LIS3MDL_CS)) {  // hardware SPI mode
  //if (! lis3mdl.begin_SPI(LIS3MDL_CS, LIS3MDL_CLK, LIS3MDL_MISO, LIS3MDL_MOSI))
{ // soft SPI
```

If the example fails with "Failed to find LIS3MDL chip", try connecting the D0/AD1 pin to ground. There's a potential issue that causes this, and connecting the pin to ground (or shorting the jumper to change the I2C address) fixes that specific issue!

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:

```
Adafruit LIS3MDL test!
LIS3MDL Found!
Performance mode set to: Medium
Operation mode set to: Continuous
Data rate set to: 155 Hz
Range set to: +-4 gauss

X:  1201        Y:  -15467      Z:  3802
        X: 17.55        Y: -226.06      Z: 55.57 uTesla


X:  1444        Y:  -14684      Z:  3581
        X: 21.10        Y: -214.62      Z: 52.34 uTesla


X:  1172        Y:  -14696      Z:  3651
        X: 16.62        Y: -214.44      Z: 53.93 uTesla
```

The sensor class in the magnetometer library reports X, Y and Z axis magnetometer readings directly in micro-Teslas (). The lis3mdl_demo 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 filed with respect to the X and Y axis, the device can be used as a compass.

Example Code

```
// Basic demo for magnetometer readings from Adafruit LIS3MDL

#include <Wire.h>
#include <Adafruit_LIS3MDL.h>
#include <Adafruit_Sensor.h>

Adafruit_LIS3MDL lis3mdl;
#define LIS3MDL_CLK 13
#define LIS3MDL_MISO 12
#define LIS3MDL_MOSI 11
#define LIS3MDL_CS 10

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

  Serial.println("Adafruit LIS3MDL test!");

  // Try to initialize!
  if (! lis3mdl.begin_I2C()) {          // hardware I2C mode, can pass in address &
alt Wire
  //if (! lis3mdl.begin_SPI(LIS3MDL_CS)) {  // hardware SPI mode
  //if (! lis3mdl.begin_SPI(LIS3MDL_CS, LIS3MDL_CLK, LIS3MDL_MISO, LIS3MDL_MOSI))
{ // soft SPI
    Serial.println("Failed to find LIS3MDL chip");
    while (1) { delay(10); }
  }
  Serial.println("LIS3MDL Found!");

  lis3mdl.setPerformanceMode(LIS3MDL_MEDIUMMODE);
  Serial.print("Performance mode set to: ");
  switch (lis3mdl.getPerformanceMode()) {
    case LIS3MDL_LOWPOWERMODE: Serial.println("Low"); break;
    case LIS3MDL_MEDIUMMODE: Serial.println("Medium"); break;
    case LIS3MDL_HIGHMODE: Serial.println("High"); break;
    case LIS3MDL_ULTRAHIGHMODE: Serial.println("Ultra-High"); break;
  }

  lis3mdl.setOperationMode(LIS3MDL_CONTINUOUSMODE);
  Serial.print("Operation mode set to: ");
  // Single shot mode will complete conversion and go into power down
  switch (lis3mdl.getOperationMode()) {
    case LIS3MDL_CONTINUOUSMODE: Serial.println("Continuous"); break;
    case LIS3MDL_SINGLEMODE: Serial.println("Single mode"); break;
    case LIS3MDL_POWERDOWNMODE: Serial.println("Power-down"); break;
  }

  lis3mdl.setDataRate(LIS3MDL_DATARATE_155_HZ);
```

```
  // You can check the datarate by looking at the frequency of the DRDY pin
  Serial.print("Data rate set to: ");
  switch (lis3mdl.getDataRate()) {
    case LIS3MDL_DATARATE_0_625_HZ: Serial.println("0.625 Hz"); break;
    case LIS3MDL_DATARATE_1_25_HZ: Serial.println("1.25 Hz"); break;
    case LIS3MDL_DATARATE_2_5_HZ: Serial.println("2.5 Hz"); break;
    case LIS3MDL_DATARATE_5_HZ: Serial.println("5 Hz"); break;
    case LIS3MDL_DATARATE_10_HZ: Serial.println("10 Hz"); break;
    case LIS3MDL_DATARATE_20_HZ: Serial.println("20 Hz"); break;
    case LIS3MDL_DATARATE_40_HZ: Serial.println("40 Hz"); break;
    case LIS3MDL_DATARATE_80_HZ: Serial.println("80 Hz"); break;
    case LIS3MDL_DATARATE_155_HZ: Serial.println("155 Hz"); break;
    case LIS3MDL_DATARATE_300_HZ: Serial.println("300 Hz"); break;
    case LIS3MDL_DATARATE_560_HZ: Serial.println("560 Hz"); break;
    case LIS3MDL_DATARATE_1000_HZ: Serial.println("1000 Hz"); break;
  }

  lis3mdl.setRange(LIS3MDL_RANGE_4_GAUSS);
  Serial.print("Range set to: ");
  switch (lis3mdl.getRange()) {
    case LIS3MDL_RANGE_4_GAUSS: Serial.println("+-4 gauss"); break;
    case LIS3MDL_RANGE_8_GAUSS: Serial.println("+-8 gauss"); break;
    case LIS3MDL_RANGE_12_GAUSS: Serial.println("+-12 gauss"); break;
    case LIS3MDL_RANGE_16_GAUSS: Serial.println("+-16 gauss"); break;
  }

  lis3mdl.setIntThreshold(500);
  lis3mdl.configInterrupt(false, false, true, // enable z axis
                          true, // polarity
                          false, // don't latch
                          true); // enabled!
}

void loop() {
  lis3mdl.read();      // get X Y and Z data at once
  // Then print out the raw data
  Serial.print("\nX:  "); Serial.print(lis3mdl.x);
  Serial.print("  \tY:  "); Serial.print(lis3mdl.y);
  Serial.print("  \tZ:  "); Serial.println(lis3mdl.z);

  /* Or....get a new sensor event, normalized to uTesla */
  sensors_event_t event;
  lis3mdl.getEvent(&event);
  /* Display the results (magnetic field is measured in uTesla) */
  Serial.print("\tX: "); Serial.print(event.magnetic.x);
  Serial.print(" \tY: "); Serial.print(event.magnetic.y);
  Serial.print(" \tZ: "); Serial.print(event.magnetic.z);
  Serial.println(" uTesla ");

  delay(100);
  Serial.println();
}
```

Arduino Docs

Arduino Docs ()

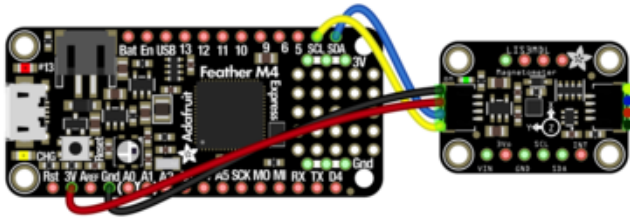
Python & CircuitPython

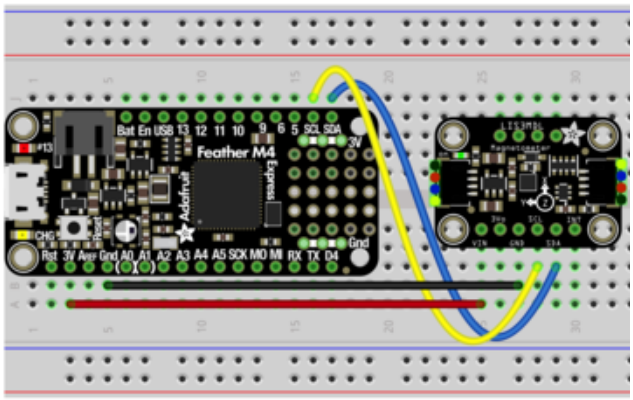
It's easy to use the LIS3MDL sensor with CircuitPython and the Adafruit CircuitPython LIS3MDL () library. This library will allow you to easily write Python code that reads the magnetometer values from the sensor.

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 LIS3MDL breakout to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C:



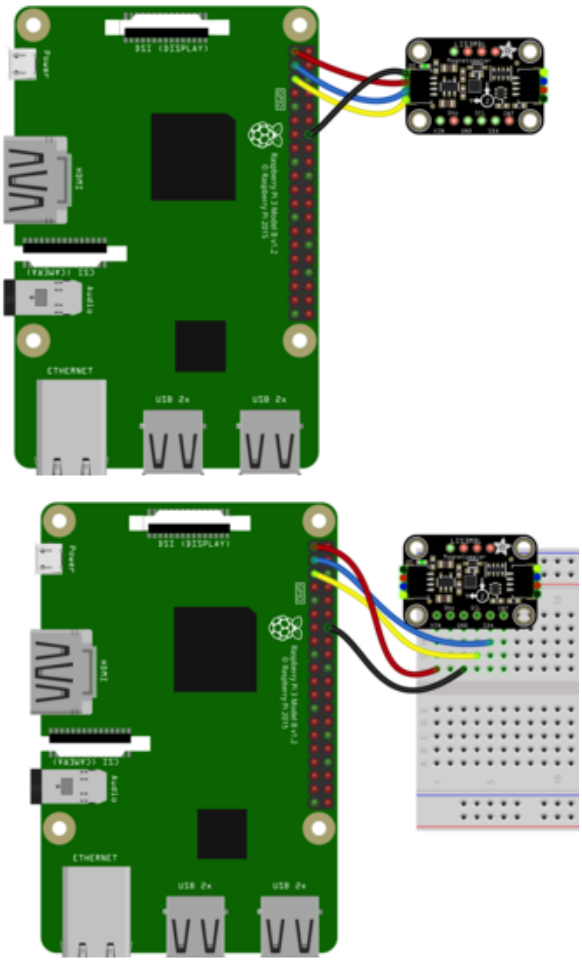


Board 3V to sensor VIN (red wire)
Board GND to sensor GND (black wire)
Board SCL to sensor SCL (yellow wire)
Board SDA to sensor SDA (blue wire)

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 LIS3MDL Library

You'll need to install the Adafruit CircuitPython LIS3MDL () 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_lis3mdl.mpy
- adafruit_bus_device
- adafruit_register

Before continuing make sure your board's lib folder or root filesystem has the adafruit_lis3mdl.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 LIS3MDL 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-lis3mdl`

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:

If the example fails with "Failed to find LIS3MDL chip", try connecting the D0/AD1 pin to ground. There's a potential issue that causes this, and connecting the pin to ground (or shorting the jumper to change the I2C address) fixes that specific issue!

```
import time
import board
import busio
import adafruit_lis3mdl

i2c = busio.I2C(board.SCL, board.SDA)
sensor = adafruit_lis3mdl.LIS3MDL(i2c)
```

```
>>> import time
>>> import board
>>> import busio
>>> import adafruit_lis3mdl
>>> i2c = busio.I2C(board.SCL, board.SDA)
>>> sensor = adafruit_lis3mdl.LIS3MDL(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))
```

```
>>> 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))
X:     -0.12, Y:     -0.04, Z:     -0.00 uT
>>>
```

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_lis3mdl

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
sensor = adafruit_lis3mdl.LIS3MDL(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

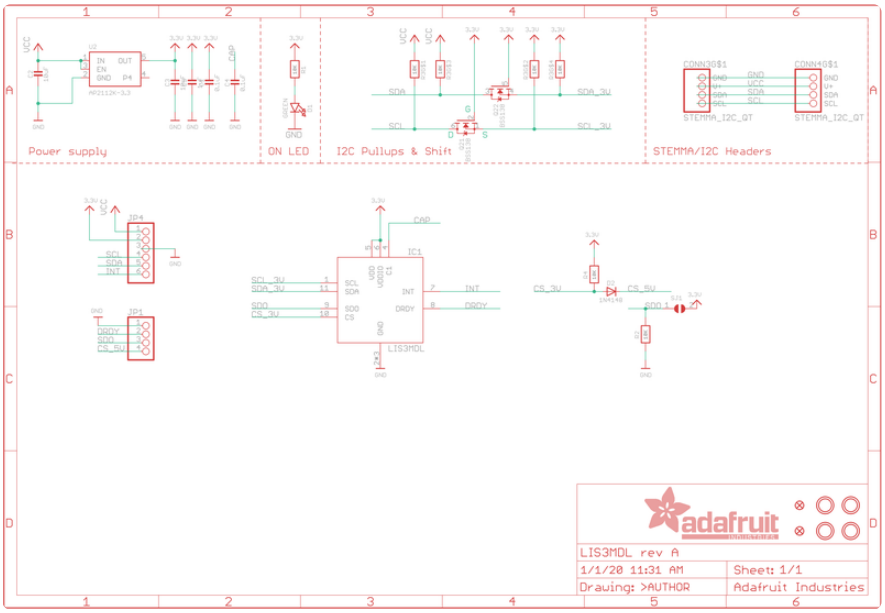
Python Docs ()

Downloads

Files

- LIS3MDL Datasheet ()
- EagleCAD files on GitHub ()
- 3D models on GitHub ()
- Fritzing object in Adafruit Fritzing Library ()

Schematic



Fab Print