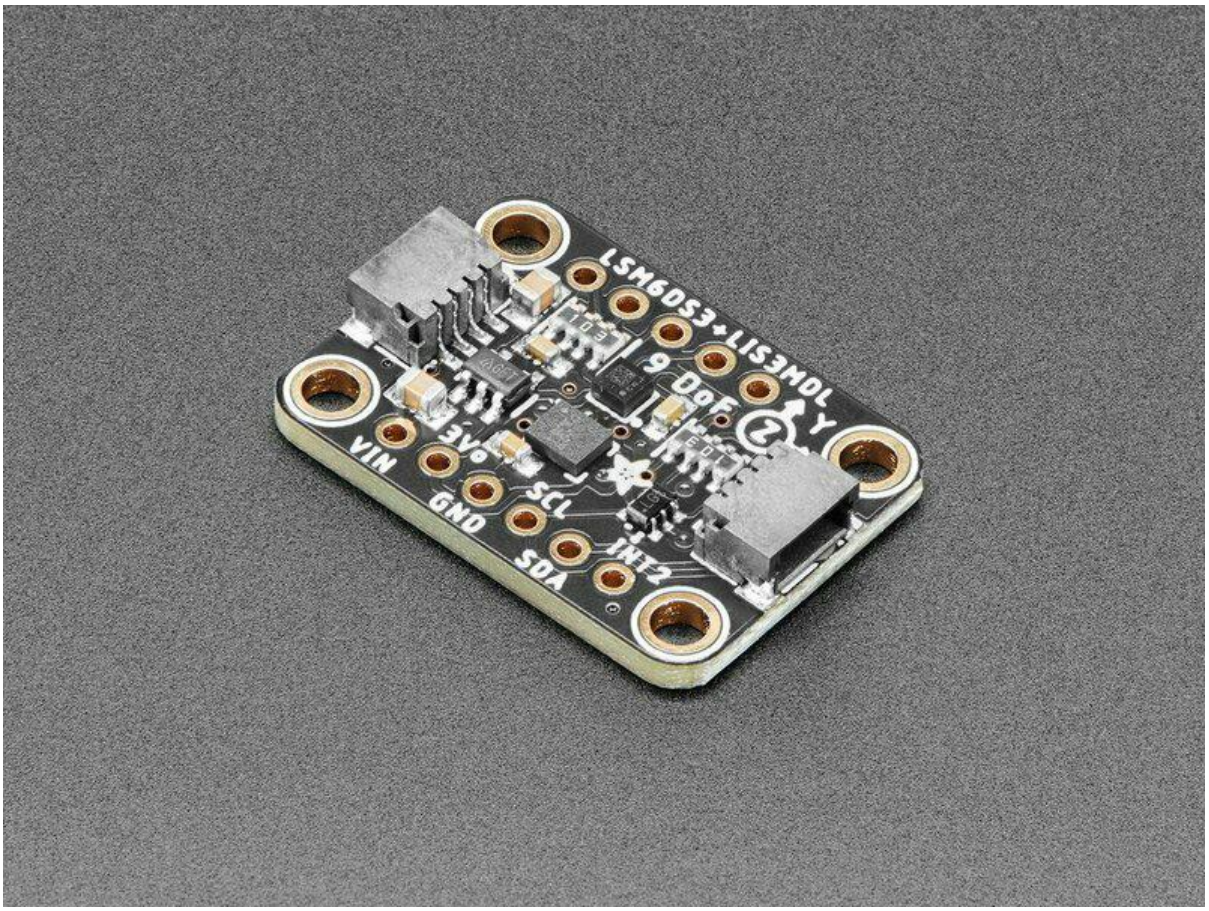
Adafruit LSM6DS3TR-C + LIS3MDL - Precision 9 DoF IMU

Created by Liz Clark

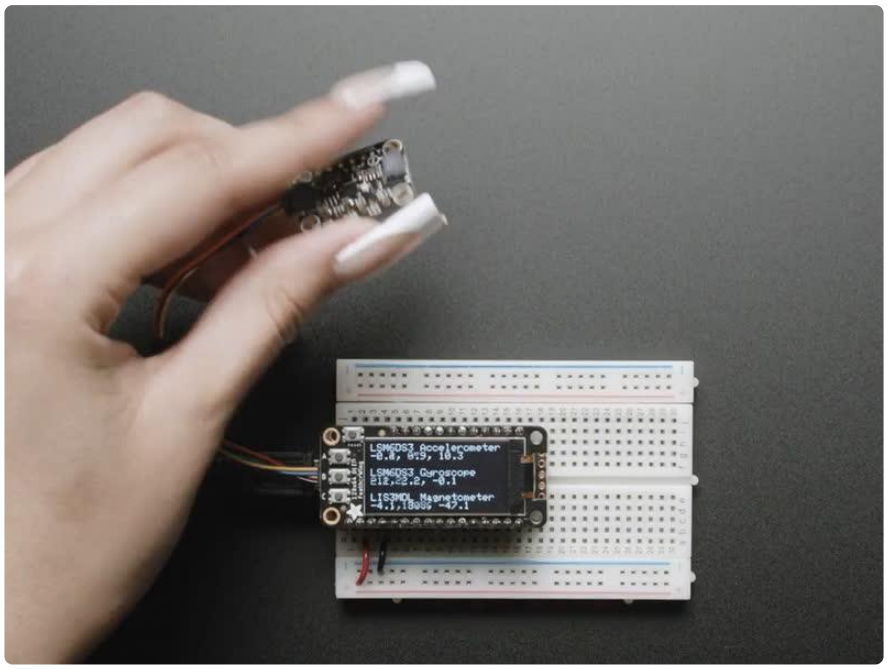


https://learn.adafruit.com/adafruit-lsm6ds3tr-c-lis3mdl-precision-9-dof-imu

Last updated on 2022-12-01 04:12:47 PM EST

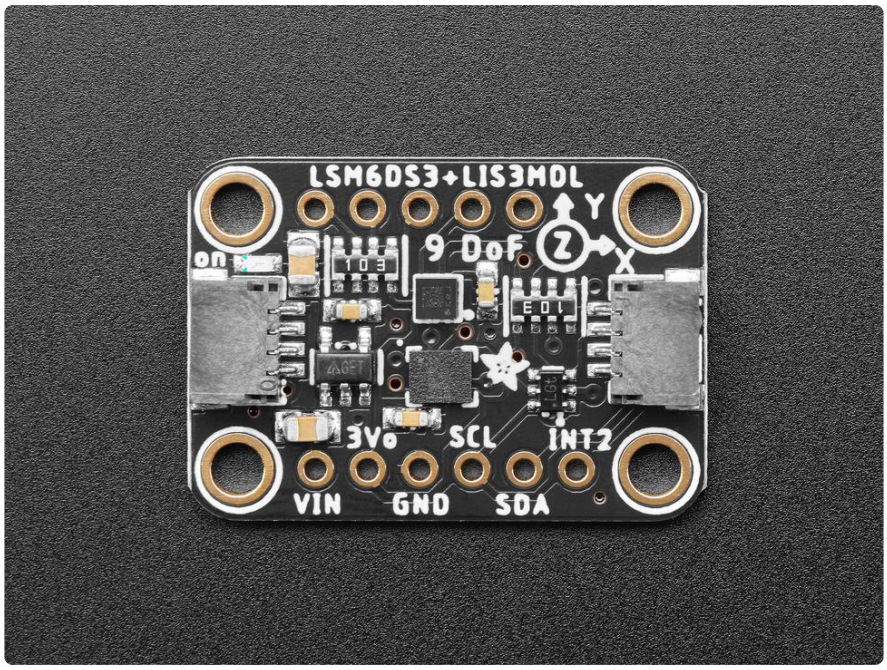
Table of Contents

Overview



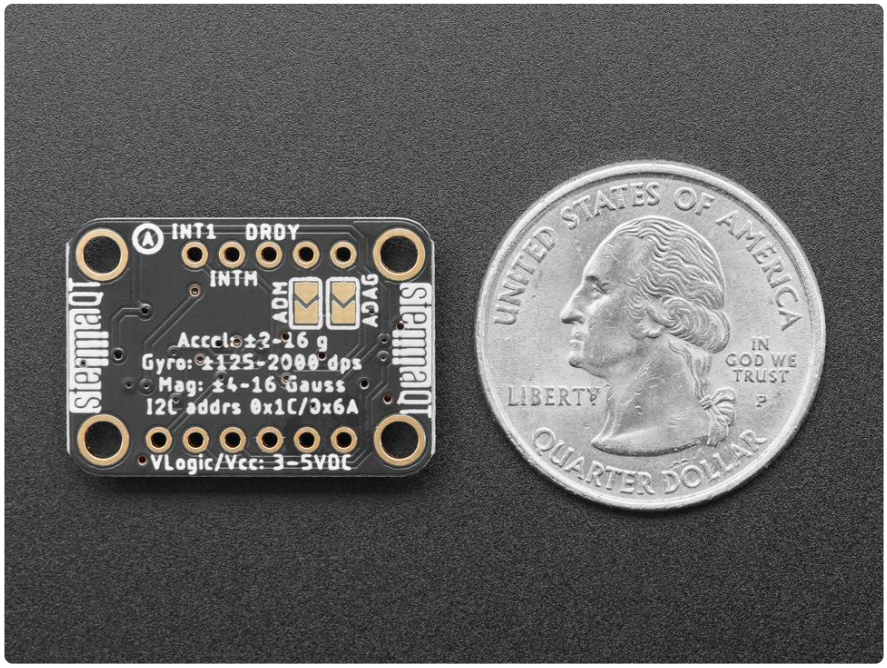
Add high-quality motion, direction, and orientation sensing to your Arduino project with this all-in-one 9 Degree of Freedom (9-DoF) sensor with sensors from ST. This little breakout contains two chips that sit side-by-side to provide 9 degrees of full-motion data.

The board includes an ST LSM6DS3TR-C (a.k.a LSM6DS3), a great entry-level 6-DoF IMU accelerometer + gyro. The 3-axis accelerometer can tell you which direction is down towards the Earth (by measuring gravity) or how fast the board is accelerating in 3D space. The 3-axis gyroscope can measure spin and twist.

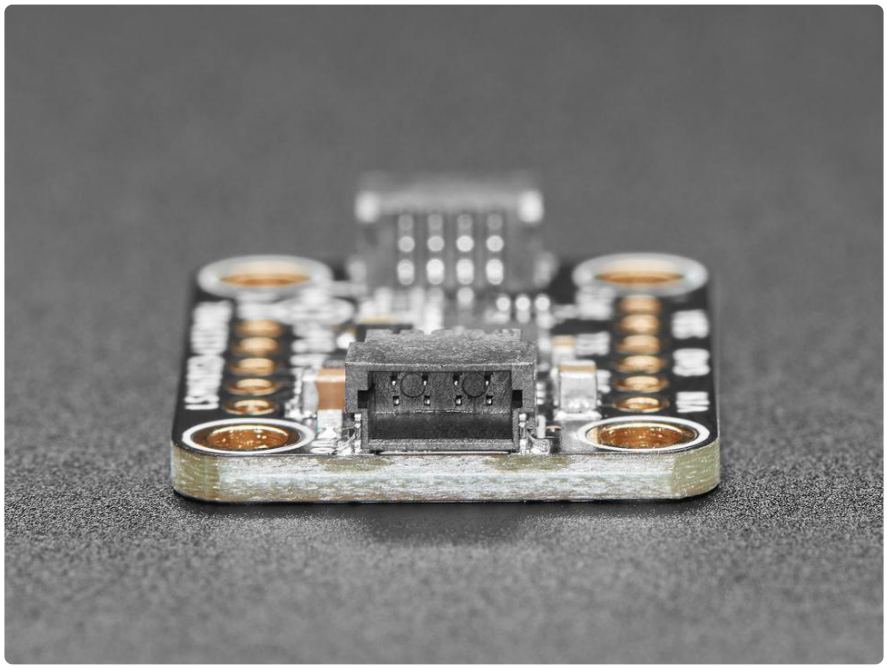


This chip is very similar to the now-discontinued LSM6DS33, a great entry-level IMU. As part of the illustrious LSM6DS family, it's well-established and well-supported, and this chip even has better performance! Note it is not firmware-compatible with the 'DS33, so you will need to recompile code (e.g our Arduino and Python libraries support the whole family but you do have to indicate which exact chip you're using)

It also includes a LIS3MDL 3-axis magnetometer that can sense where the strongest magnetic force is coming from, generally used to detect magnetic north. The three triple-axis sensors add up to 9 degrees of freedom, by combining this data you can orient the board. Check out our guide on how to do that!



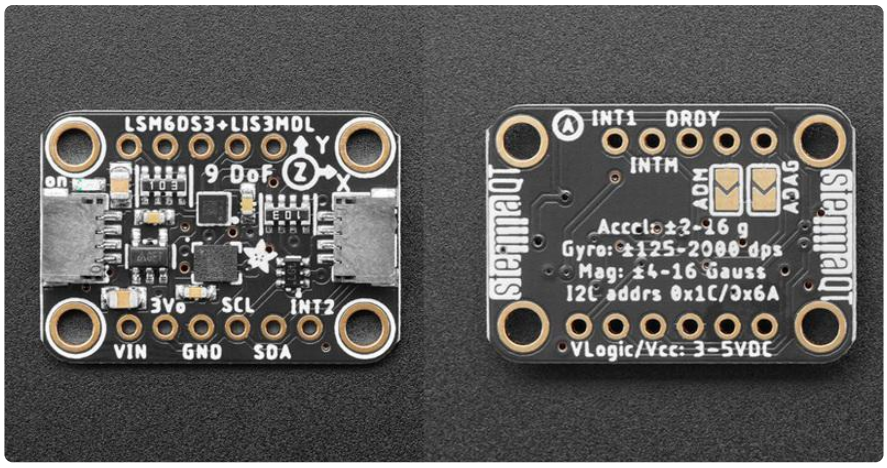
To make getting started fast and easy, we placed the sensors on a compact breakout board with voltage regulation and level-shifted inputs. That way you can use them with 3V or 5V power/logic devices without worry. To make usage simple, we expose only the I2C interface and some interrupt pins from each chip. The breakout comes fully assembled and tested, with some extra header so you can use it on a breadboard. Four mounting holes make for a secure connection.



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 like the STM32F405 Feather () with a plug-and-play cable to get 9 DoF data ASAP. You can change the I2C addresses on the back using the solder jumpers, to have two of these sensor boards on one bus. QT Cable is not included, but we have a variety in the shop ().

We also wrote libraries to help you get these sensors integrated with your Arduino/C++. This library covers the accel/gyro () and this library is for the magnetometer (). For advanced Arduino usage, ST has their own fully-featured library that includes extras such as FIFO management and tap detection () for the LSM6DS3TR-C and also for the LIS3MDL magnetometer ().

LSM6DS3TR-C + LIS3MDL Pinouts



The default I2C address for the LSM6DS3 accelerometer is 0x6A.

The default I2C address for the LIS3MDL magnetometer is 0x1C.

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 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 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 connectors to dev boards with S TEMMA QT connectors or to other things with various associated accessories ().

Address Jumpers

On the back of the board are two address jumpers, labeled ADM and AGAD, to the right of the I2C Addr label on the board silk. ADM controls the magnetometer's I2C address and AGAD controls the accelerometer's I2C address. These jumpers allow you to chain up to 2 of these boards on the same pair of I2C clock and data pins. To do so, you solder the jumper "closed" by connecting the two pads.

The default I2C address for the accelerometer is 0x6A. The other address options can be calculated by “adding” the AGAD to the base of 0x6A.

AGAD sets the lowest bit with a value of 1. The final address is 0x6A + AGAD which would be 0x6B.

If AGAD is soldered closed, the address is 0x6A + 1 = 0x6B

The table below shows all possible addresses for the accelerometer, and whether the pin should be high (closed) or low (open).

ADDR	AGAD
0x6A	L
0x6B	H

The default I2C address for the magnetometer is 0x1C. The other address options can be calculated by “adding” the ADM to the base of 0x1C.

ADM sets the lowest bit with a value of 2. The final address is 0x1C + ADM which would be 0x1E.

If ADM is soldered closed, the address is 0x1C + 2 = 0x1E

The table below shows all possible addresses for the magnetometer, and whether the pin should be high (closed) or low (open).

ADDR	ADM
0x1C	L
0x1E	H

I2C Address Pins

- ADM - LIS3MDL Magnetometer I2C address pin. Pulling this pin high will change the I2C address from 0x1C to 0x1E.
- AGAD - LSM6DS3TR-C Accel/Gyro I2C address pin. Pulling this pin high will change the I2C address from 0x6A to 0x6B.

Other Pins

- INT1 -This is the primary interrupt pin. You can setup the LSM6DS3TR-C to pull this low when certain conditions are met such as new measurement data being available. Consult the datasheet () for usage.
- INT2 -This is the primary interrupt pin. You can setup the LSM6DS3TR-C to pull this low when certain conditions are met such as new measurement data being available. Consult the datasheet () for usage.
- INTM - This is the primary interrupt pin for the Magnetometer. 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.

Power LED

- Power LED - In the upper left corner, above the STEMMA connector, on the front of the board, is the power LED, labeled on. It is the green LED.

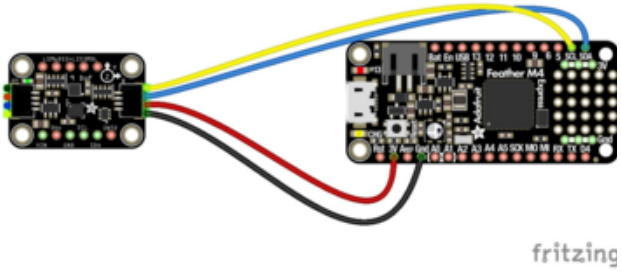
Python & CircuitPython

It's easy to use the LIS3MDL + LSM6DS3TR-C sensor combos with Python or CircuitPython, and the Adafruit_CircuitPython_LSM6DS () and Adafruit_CircuitPython_LIS3MDL () libraries. These libraries allow you to easily write Python code that read measurements from the accelerometer, gyro, and magnetometer.

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 ().

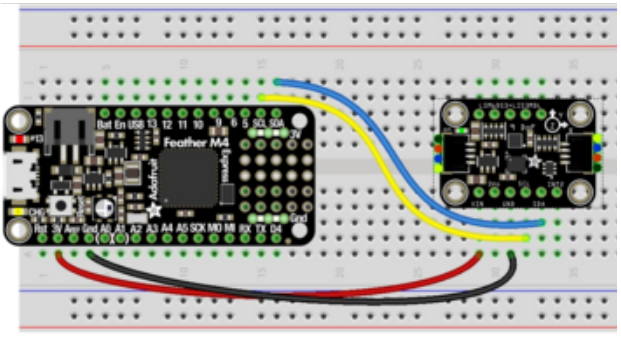
CircuitPython Microcontroller Wiring

First, wire up a LSM6DS3TR-C + LIS3MDL to your board exactly as shown below. Here's an example of wiring a Feather M4 to the sensor with I2C using one of the handy STEMMA QT () connectors:



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)

You can also use standard 0.100" pitch headers to wire it up on a breadboard:

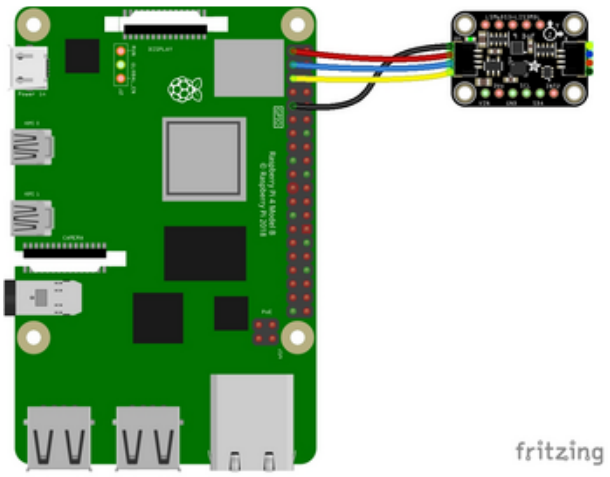


Board 3V to sensor VIN (red wire)
Board GND to sensor GND (black wire)
Board SCL to sensor SCL (yellow wire)
Board SDA to sensor SDA (blue wire)

Python Computer Wiring

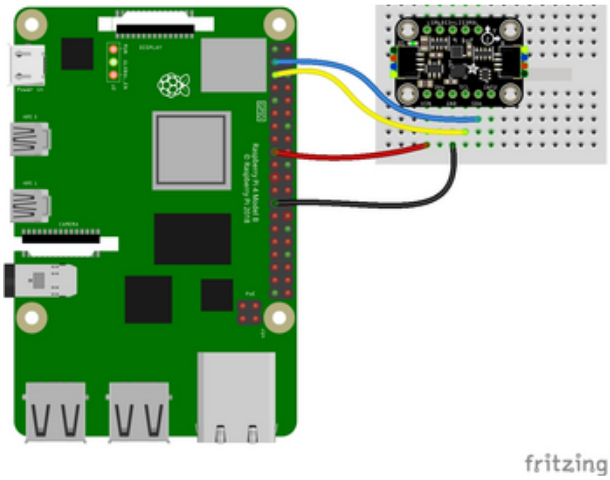
Since there's dozens of Linux computers/boards you can use, below shows wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported ().

Here's the Raspberry Pi wired to the sensor using I2C and a STEMMA QT () connector:



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

Finally here is an example of how to wire up a Raspberry Pi to the sensor using a solderless breadboard:



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

Python Installation of LIS3MDL and LSM6DS Libraries

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 commands:

- `pip3 install adafruit-circuitpython-lsm6ds`
- `pip3 install adafruit-circuitpython-lis3mdl`

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

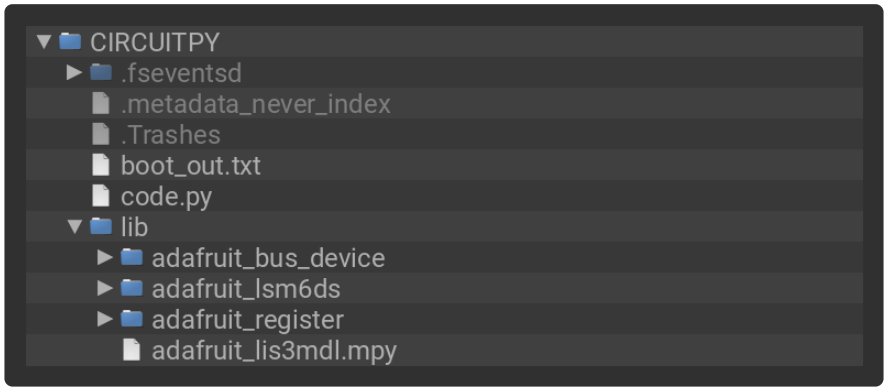
CircuitPython Usage

To use with CircuitPython, you need to first install the LSM6DS and LIS3MDL libraries, and their dependencies, into the lib folder on your CIRCUITPY drive. Then you need to update code.py with the example script.

Thankfully, we can do this in one go. In the example below, click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the c ode.py file to your CIRCUITPY drive.

Your CIRCUITPY/lib folder should contain the following folders and file:

- adafruit_bus_device/
- adafruit_lsm6ds/
- adafruit_register/
- adafruit_lis3mdl.mpy



Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing code.py with whatever you named the file:

```
python3 code.py
```

Example Code

Comment out the LSM6DSOX import at the beginning of the code.

```
from adafruit_lsm6ds.lsm6dsox import LSM6DSOX as LSM6DS
```

Then, uncomment the LSM6DS3 import to import the correct library for the LSM6DS3TR-C accelerometer.

```
# from adafruit_lsm6ds.lsm6ds3 import LSM6DS3 as LSM6DS
```

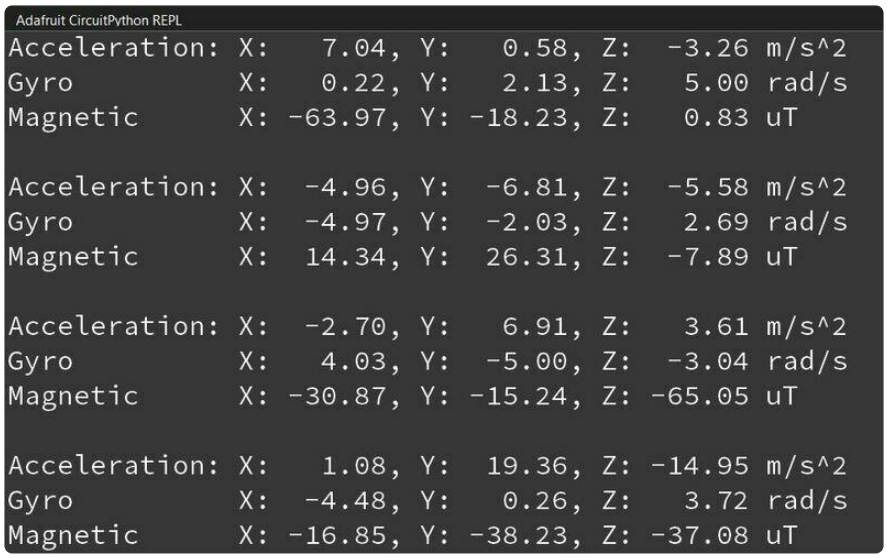
If the example fails with "Failed to find LIS3MDL chip", try bridging the AGAD pads on the back of the chip. There's a potential issue that causes this, and shorting the jumper to change the I2C address fixes that specific issue!

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

import time
import board
from adafruit_lsm6ds.lsm6dsox import LSM6DSOX as LSM6DS

# To use LSM6DS33, comment out the LSM6DSOX import line
# and uncomment the next line
# from adafruit_lsm6ds.lsm6ds33 import LSM6DS33 as LSM6DS

# To use ISM330DHCX, comment out the LSM6DSOX import line
# and uncomment the next line
# from adafruit_lsm6ds.lsm330dhcx import ISM330DHCX as LSM6DS

# To use LSM6DS3TR-C, comment out the LSM6DSOX import line
# and uncomment the next line
# from adafruit_lsm6ds.lsm6ds3 import LSM6DS3 as LSM6DS

from adafruit_lis3mdl import 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
accel_gyro = LSM6DS(i2c)
mag = LIS3MDL(i2c)
```

```
while True:
    acceleration = accel_gyro.acceleration
    gyro = accel_gyro.gyro
    magnetic = mag.magnetic
    print(
        "Acceleration: X:{0:7.2f}, Y:{1:7.2f}, Z:{2:7.2f} m/
s^2".format(*acceleration)
    )
    print("Gyro          X:{0:7.2f}, Y:{1:7.2f}, Z:{2:7.2f} rad/s".format(*gyro))
    print("Magnetic      X:{0:7.2f}, Y:{1:7.2f}, Z:{2:7.2f} uT".format(*magnetic))
    print("")
    time.sleep(0.5)
```

If running CircuitPython: Once everything is saved to the CIRCUITPY drive, connect to the serial console () to see the data printed out!

If running Python: The console output will appear wherever you are running Python.



Twist and turn your LSM6DS3TR-C + LIS3MDL and see the values from the accelerometer, gyroscope and magnetometer print out to the REPL!

First you import the necessary modules and libraries. Then you instantiate the sensor on I2C.

Then you're ready to read data from the sensor's accelerometer, gyroscope and magnetometer.

The following values will print out to the REPL from the sensor:

- Acceleration - The acceleration forces in the X, Y, and Z axes in m/s^2
- Gyro - The rotation measurement on the X, Y, and Z axes in degrees/sec
- Magnetic - The magnetic forces on the X, Y, and Z axes in micro-Teslas (uT)

That's all there is to using the LSM6DS3TR-C + LIS3MDL with CircuitPython!

LSM6DS3TR-C Python Docs

LSM6DS3TR-C Python Docs ()

LIS3MDL Python Docs

LIS3MDL Python Docs ()

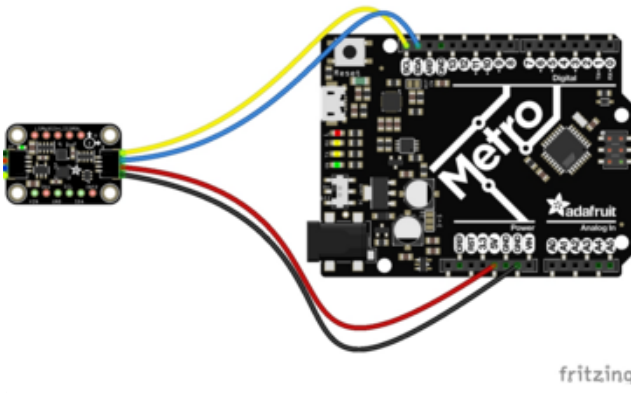
Arduino

Using the LSM6DS3TR-C + LIS3MDL with Arduino involves wiring up the sensor to your Arduino-compatible microcontroller, installing the Adafruit_LSM6DS () library, installing the Adafruit_LIS3MDL () library and running the provided example code.

Wiring

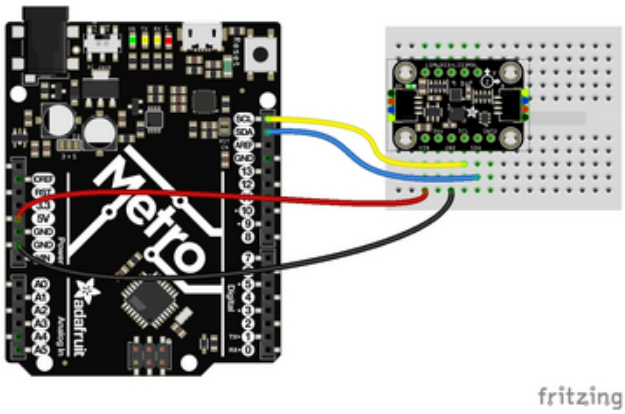
Wire as shown for a 5V board like an Uno. If you are using a 3V board, like an Adafruit Feather, wire the board's 3V pin to the LSM6DS3TR-C + LIS3MDL VIN.

Here is an Adafruit Metro wired up to the LSM6DS3TR-C + LIS3MDL using the STEMMA QT connector:



Board 5V 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)

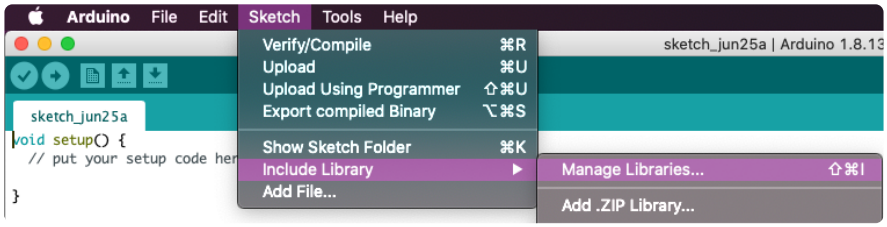
Here is an Adafruit Metro wired up using a solderless breadboard:



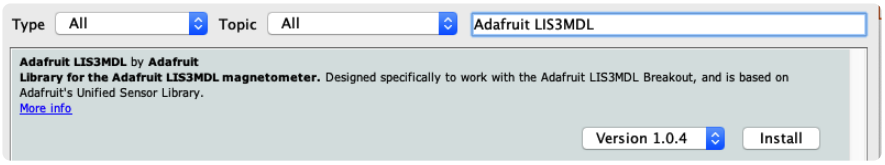
Board 5V to sensor VIN (red wire)
Board GND to sensor GND (black wire)
Board SCL to sensor SCL (yellow wire)
Board SDA to sensor SDA (blue wire)

Library Installation

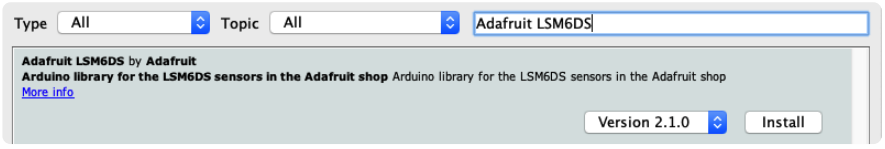
You can install the Adafruit LIS3MDL library and the Adafruit LSM6DS library for Arduino using the Library Manager in the Arduino IDE.



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



Follow the same process for the Adafruit LSM6DS library.



If asked about dependencies, click "Install all".

If the "Dependencies" window does not come up, then you already have the dependencies installed.

If the dependencies are already installed, you must make sure you update them through the Arduino Library Manager before loading the example!

Example Code

The example is written to work with the LIS3MDL + LSM6DSOX. To use with the LSM6DSTR-C + LIS3MDL, uncomment the following lines.

```
//#include &lt;Adafruit_LSM6DS3TRC.h&gt;
//Adafruit_LSM6DS3TRC lsm6ds;
```

```
// SPDX-FileCopyrightText: 2020 Kattni Rembor for Adafruit Industries
//
// SPDX-License-Identifier: MIT

// Basic demo for accelerometer, gyro, and magnetometer readings
// from the following Adafruit ST Sensor combo boards:
// * LSM6DSOX + LIS3MDL FeatherWing : https://www.adafruit.com/product/4565
// * ISM330DHCX + LIS3MDL FeatherWing https://www.adafruit.com/product/4569
// * LSM6DSOX + LIS3MDL Breakout : https://www.adafruit.com/product/4517
// * LSM6DS33 + LIS3MDL Breakout Lhttps://www.adafruit.com/product/4485

#include <Adafruit_LSM6DSOX.h>
Adafruit_LSM6DSOX lsm6ds;

// To use with the LSM6DS33+LIS3MDL breakout, uncomment these two lines
// and comment out the lines referring to the LSM6DSOX above
//#include <Adafruit_LSM6DS33.h>
//Adafruit_LSM6DS33 lsm6ds;

// To use with the ISM330DHCX+LIS3MDL Feather Wing, uncomment these two lines
// and comment out the lines referring to the LSM6DSOX above
//#include <Adafruit_ISM330DHCX.h>
//Adafruit_ISM330DHCX lsm6ds;

// To use with the LSM6D3TR-C+LIS3MDL breakout, uncomment these two lines
// and comment out the lines referring to the LSM6DSOX above
//#include <Adafruit_LSM6DS3TRC.h>
//Adafruit_LSM6DS3TRC lsm6ds;
```

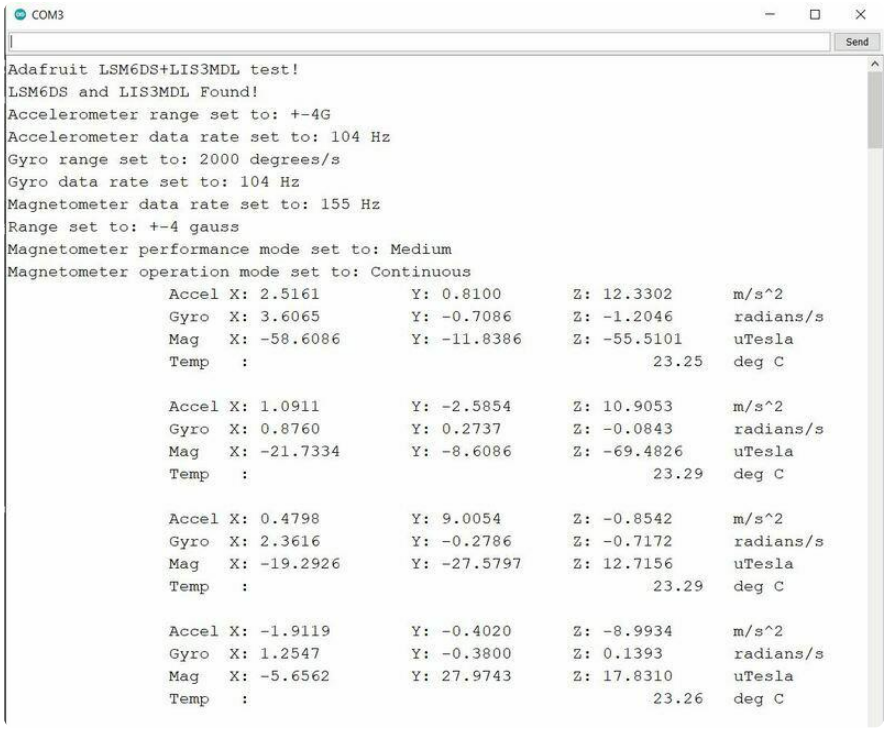
```
#include <Adafruit_LIS3MDL.h>
Adafruit_LIS3MDL lis3mdl;

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

  Serial.println("Adafruit LSM6DS+LIS3MDL test!");

  bool lsm6ds_success, lis3mdl_success;

  // hardware I2C mode, can pass in address & alt Wire

  lsm6ds_success = lsm6ds.begin_I2C();
  lis3mdl_success = lis3mdl.begin_I2C();

  if (!lsm6ds_success){
    Serial.println("Failed to find LSM6DS chip");
  }
  if (!lis3mdl_success){
    Serial.println("Failed to find LIS3MDL chip");
  }
  if (!(lsm6ds_success && lis3mdl_success)) {
    while (1) {
      delay(10);
    }
  }

  Serial.println("LSM6DS and LIS3MDL Found!");

  // lsm6ds.setAccelRange(LSM6DS_ACCEL_RANGE_2_G);
  Serial.print("Accelerometer range set to: ");
  switch (lsm6ds.getAccelRange()) {
  case LSM6DS_ACCEL_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case LSM6DS_ACCEL_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case LSM6DS_ACCEL_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case LSM6DS_ACCEL_RANGE_16_G:
    Serial.println("+-16G");
    break;
  }

  // lsm6ds.setAccelDataRate(LSM6DS_RATE_12_5_HZ);
  Serial.print("Accelerometer data rate set to: ");
  switch (lsm6ds.getAccelDataRate()) {
  case LSM6DS_RATE_SHUTDOWN:
    Serial.println("0 Hz");
    break;
  case LSM6DS_RATE_12_5_HZ:
    Serial.println("12.5 Hz");
    break;
  case LSM6DS_RATE_26_HZ:
    Serial.println("26 Hz");
    break;
  case LSM6DS_RATE_52_HZ:
    Serial.println("52 Hz");
    break;
  case LSM6DS_RATE_104_HZ:
    Serial.println("104 Hz");
    break;
  case LSM6DS_RATE_208_HZ:
    Serial.println("208 Hz");
```

```
    break;
  case LSM6DS_RATE_416_HZ:
    Serial.println("416 Hz");
    break;
  case LSM6DS_RATE_833_HZ:
    Serial.println("833 Hz");
    break;
  case LSM6DS_RATE_1_66K_HZ:
    Serial.println("1.66 KHz");
    break;
  case LSM6DS_RATE_3_33K_HZ:
    Serial.println("3.33 KHz");
    break;
  case LSM6DS_RATE_6_66K_HZ:
    Serial.println("6.66 KHz");
    break;
  }

  // lsm6ds.setGyroRange(LSM6DS_GYRO_RANGE_250_DPS );
  Serial.print("Gyro range set to: ");
  switch (lsm6ds.getGyroRange()) {
  case LSM6DS_GYRO_RANGE_125_DPS:
    Serial.println("125 degrees/s");
    break;
  case LSM6DS_GYRO_RANGE_250_DPS:
    Serial.println("250 degrees/s");
    break;
  case LSM6DS_GYRO_RANGE_500_DPS:
    Serial.println("500 degrees/s");
    break;
  case LSM6DS_GYRO_RANGE_1000_DPS:
    Serial.println("1000 degrees/s");
    break;
  case LSM6DS_GYRO_RANGE_2000_DPS:
    Serial.println("2000 degrees/s");
    break;
  case ISM330DHCX_GYRO_RANGE_4000_DPS:
    Serial.println("4000 degrees/s");
    break;
  }
  // lsm6ds.setGyroDataRate(LSM6DS_RATE_12_5_HZ);
  Serial.print("Gyro data rate set to: ");
  switch (lsm6ds.getGyroDataRate()) {
  case LSM6DS_RATE_SHUTDOWN:
    Serial.println("0 Hz");
    break;
  case LSM6DS_RATE_12_5_HZ:
    Serial.println("12.5 Hz");
    break;
  case LSM6DS_RATE_26_HZ:
    Serial.println("26 Hz");
    break;
  case LSM6DS_RATE_52_HZ:
    Serial.println("52 Hz");
    break;
  case LSM6DS_RATE_104_HZ:
    Serial.println("104 Hz");
    break;
  case LSM6DS_RATE_208_HZ:
    Serial.println("208 Hz");
    break;
  case LSM6DS_RATE_416_HZ:
    Serial.println("416 Hz");
    break;
  case LSM6DS_RATE_833_HZ:
    Serial.println("833 Hz");
    break;
  case LSM6DS_RATE_1_66K_HZ:
    Serial.println("1.66 KHz");
```

```
    break;
  case LSM6DS_RATE_3_33K_HZ:
    Serial.println("3.33 KHz");
    break;
  case LSM6DS_RATE_6_66K_HZ:
    Serial.println("6.66 KHz");
    break;
  }

  lis3mdl.setDataRate(LIS3MDL_DATARATE_155_HZ);
  // You can check the datarate by looking at the frequency of the DRDY pin
  Serial.print("Magnetometer 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.setPerformanceMode(LIS3MDL_MEDIUMMODE);
  Serial.print("Magnetometer 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("Magnetometer 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.setIntThreshold(500);
  lis3mdl.configInterrupt(false, false, true, // enable z axis
                          true, // polarity
                          false, // don't latch
                          true); // enabled!

}

void loop() {

  sensors_event_t accel, gyro, mag, temp;

  //  /* Get new normalized sensor events */
  lsm6ds.getEvent(&accel, &gyro, &temp);
  lis3mdl.getEvent(&mag);
```

```
  /* Display the results (acceleration is measured in m/s^2) */
  Serial.print("\t\tAccel X: ");
  Serial.print(accel.acceleration.x, 4);
  Serial.print(" \tY: ");
  Serial.print(accel.acceleration.y, 4);
  Serial.print(" \tZ: ");
  Serial.print(accel.acceleration.z, 4);
  Serial.println(" \tm/s^2 ");

  /* Display the results (rotation is measured in rad/s) */
  Serial.print("\t\tGyro  X: ");
  Serial.print(gyro.gyro.x, 4);
  Serial.print(" \tY: ");
  Serial.print(gyro.gyro.y, 4);
  Serial.print(" \tZ: ");
  Serial.print(gyro.gyro.z, 4);
  Serial.println(" \tradians/s ");

  /* Display the results (magnetic field is measured in uTesla) */
  Serial.print(" \t\tMag   X: ");
  Serial.print(mag.magnetic.x, 4);
  Serial.print(" \tY: ");
  Serial.print(mag.magnetic.y, 4);
  Serial.print(" \tZ: ");
  Serial.print(mag.magnetic.z, 4);
  Serial.println(" \tuTesla ");

  Serial.print("\t\tTemp   :\t\t\t\t\t");
  Serial.print(temp.temperature);
  Serial.println(" \tdeg C");
  Serial.println();
  delay(1000);

}
```

Upload the sketch to your board and open up the Serial Monitor (Tools -> Serial Monitor) at 115200 baud. You should see the values from the embedded temperature sensor, accelerometer, gyroscope and magnetometer being printed out. You'll see the values change depending on the movement of the sensor.



LSM6DS Arduino Docs

LSM6DS Arduino Docs ()

LIS3MDL Arduino Docs

LIS3MDL Arduino Docs ()

Downloads

Files

- LIS3MDL datasheet ()
- LSM6DS3TR-C Datasheet ()
- EagleCAD PCB files on GitHub ()
- 3D models on GitHub ()
- Fritzing object in the Adafruit Fritzing Library ()

Schematic and Fab Print