ST 9-DoF Combo Breakouts and Wings

Created by Kattni Rembor

https://learn.adafruit.com/st-9-dof-combo

Last updated on 2023-08-29 03:07:10 PM EDT
# Table of Contents

## Overview
- LSM6DSOX or ISM330DHCX + LIS3MDL Pinouts
  - Breakout Power Pins
  - FeatherWing Power Pins
  - I2C Logic Pins
  - I2C Address Pins
  - Other Pins
  - FeatherWing Pins
- LSM6DS33 + LIS3MDL Pinouts
  - Power Pins
  - I2C Logic Pins
  - I2C Address Pins
  - Other Pins
- LSM6DS3TR-C + LIS3MDL Pinouts
  - Power Pins
  - I2C Logic Pins
  - Address Jumpers
  - I2C Address Pins
  - Other Pins
  - Power LED

## Assembly
- Arduino
  - Wiring
  - Library Installation
  - Load Example
  - Example Code
- LIS3MDL Arduino Docs
- LSM6DS Arduino Docs

## Python & CircuitPython
- CircuitPython Microcontroller Wiring
- Python Computer Wiring
- Python Installation of LIS3MDL and LSM6DS Libraries
- CircuitPython Installation of LIS3MDL and LSM6DS Libraries
- CircuitPython & Python Usage
- Full Example Code
- LIS3MDL Python docs
- LSM6DS Python Docs

## AHRS Fusion of 9 DoF Sensors
Downloads

- Files
- LSM6DSOX + LIS3MDL breakout Fab Print and Schematic
- LSM6DSOX + LIS3MDL FeatherWing Fab Print and Schematic
- LSM6DS33 + LIS3MDL Fab Print and Schematic
- LSM6DS3TR-C + LIS3MDL Fab Print and Schematic
- ISM330DHCX + LIS3MDL Fab Print and Schematic
Overview

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

The breakout and FeatherWings include an LSM6DSOX, LSM6DS3TR-C, ISM330DHC, ISM330DHCX, or LSM6DS33. These are 6-DoF IMUs with 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 that can measure spin and twist. The three triple-axis sensors add up to 9 degrees of freedom.

The LSM6DS33 and LSM6DS3TR-C are lower cost IMU's. As of July 2022, the LSM6DS33 is end of life.

This LSM6DSOX is a mid-range cost IMU has very low gyro zero rate and noise () compared to the LSM6DS33, so it's excellent for orientation fusion usage: you'll get less drift and faster responses.

The LSM6DSOX, LSM6DS33 and LSM6DS3TR-C all have flexible data rates and ranges. For the accelerometer: ±2/±4/±8/±16 g at 1.6 Hz to 6.7KHz update rate. For the gyroscope: ±125/±250/±500/±1000/±2000 dps at 12.5 Hz to 6.7 KHz. There's also some nice extras, such as built-in tap detection, activity detection, and pedometer/step counter. The LSM6DSOX also has a programmable finite state machine / machine learning core that can perform some basic gesture recognition.

The ISM330DHCX also has flexible data rates and ranges. For the accelerometer: ±2/±4/±8/±16 g at 1.6 Hz to 6.7KHz update rate. For the gyroscope: ±125/±250/±500/±1000/±2000/±4000 dps at 12.5 Hz to 6.7 KHz. In particular, this is one of the few gyro's we stock with 4000 dps range, usually they top out at 2000. This sensor has extra calibration and compensation circuits to give it excellent performance in a wide environmental range from -40 to +105°C.

Also included is 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.
For the FeatherWings, both sensors are connected over the shared I2C bus, so you can use it with any and all Feathers! We also break out the interrupt pins and address-selection jumpers in case you want multiple Feathers or have I2C address conflicts. We've got both Arduino (C/C++) and CircuitPython libraries available so you can use it with any Feather board and get data readings in under 5 minutes. Four mounting holes make for a secure connection.
To make getting started fast and easy for non-Feather use, 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.

For the breakouts, since the sensors are wired together over 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.
LSM6DSOX or ISM330DHCX + LIS3MDL
Pinouts

Breakout 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
FeatherWing Power Pins

- 3.3V - this is the 3.3V input to the FeatherWing, we use the Feather's power regulator to generate the clean 3V power needed.
- GND - common ground for power and logic

On the earlier versions of the FeatherWing, SCL and SDA are swapped on the silk. The pins themselves are accurate. The issue is only with the labels printed on the board.

I2C Logic Pins

- SCL - I2C clock pin, connect to your microcontroller's I2C clock line. On the breakouts, this pin is level shifted so you can use 3-5V logic. On the FeatherWing, there is no level shifter. There's a 10K pullup on this pin.
- SDA - I2C data pin, connect to your microcontroller's I2C data line.
- On the breakouts, this pin is level shifted so you can use 3-5V logic. On the FeatherWing, there is no level shifter. 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 ()

I2C Address Pins

- ADM / Mag Addr - LIS3MDL Magnetometer I2C address pin. Pulling this pin high or bridging the solder jumper on the back will change the I2C address from 0x1C to 0x1E.
• ADAG / A/G Addr - LSM6DSOX or ISM330DHCX Accel/Gyro I2C address pin. Pulling this pin high or bridging the solder jumper on the back will change the I2C address from 0x6A to 0x6B.

Other Pins

• INT1 - This is the primary interrupt pin for the Accel/Gyro. You can setup the LSM6DSOX or ISM330DHCX 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 secondary interrupt pin for the Accel/Gyro. You can setup the LSM6DSOX or ISM330DHCX 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.

FeatherWing Pins

• SPI Logic pins. Located toward the center of the FeatherWing, these are the same for both the LIS3MDL and LSM6DSOX or ISM330DHCX. 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 Sensor Data In / Microcontroller Out Sensor In pin, for data sent from your processor to the LIS3MDL or LSM6DSOX.
  ◦ SDO - this is the Sensor Data Out / Microcontroller In Sensor Out pin, for data sent from the LIS3MDL, ISM330DHCX, or LSM6DSOX to your processor.
  ◦ CS - this is the Chip Select pin, drop it low to start an SPI transaction. It’s an input to the chip.

• External I2C/SPI pins - Control a separate sensor on a separate I2C/SPI bus.

  ◦ SCL - I2C clock pin OR the SPI Clock pin.
  ◦ SDA - I2C data pin OR the Sensor Data In / Microcontroller Out Sensor In pin, for data sent from your processor to the LIS3MDL, ISM330DHCX, or LSM6DSOX.
- SDO - this is the Sensor Data Out / Microcontroller In Sensor Out pin, for data sent from the LIS3MDL, ISM330DHCX, or LSM6DSOX to your processor.
- CS - this is the Chip Select pin, drop it low to start an SPI transaction. Its an input to the chip.

LSM6DS33 + LIS3MDL 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 ().

**I2C Address Pins**

• ADM - LIS3MDL Magnetometer I2C address pin. Pulling this pin high or bridging the solder jumper on the back will change the I2C address from 0x1C to 0x1E.
• AGAD - LSM6DS33 Accel/Gyro I2C address pin. Pulling this pin high or bridging the solder jumper on the back will change the I2C address from 0x6A to 0x6B.

**Other Pins**

• INT1 - This is the primary interrupt pin for the Accel/Gyro. You can setup the LSM6DS33 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.

**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 connect them to dev boards with STEMMA 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).
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).

<table>
<thead>
<tr>
<th>ADDR</th>
<th>AGAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x6A</td>
<td>L</td>
</tr>
<tr>
<td>0x6B</td>
<td>H</td>
</tr>
</tbody>
</table>

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.

Assembly

When putting together your Featherwings, think about how you want it to connect, you can use stacking headers:

Or plain female socket headers:
The most common method of attachment for the featherwing is putting stacking or female headers on the Feather mainboard and then putting the Wing on top:

But don't forget, you can also put the stacking headers on the wing and stack the Feather on top of it!

Arduino

Wiring

The following shows connecting to an Arduino using I2C.
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
The final results should resemble the illustration above, showing an Adafruit Metro development board.

For the FeatherWing:
Plug the FeatherWing into a Feather. This image is the Wing plugged into a Feather M4.

Both the breakouts and FeatherWing use I2C for communication only!

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.

Then follow the same process for the Adafruit BusIO library.
Finally follow the same process for the Adafruit Unified Sensor library:

Load Example

Download the example below. Open up File -> Open and navigate to the downloaded lis3mdl_lsm6ds_test and upload to your Arduino wired up to the sensor.

The example is written to work with the LIS3MDL + LSM6DSOX. To use with the LIS3MDL + LSM6DS33, uncomment the following lines.

```
#include "Adafruit_LSM6DS33.h"
Adafruit_LSM6DS33 lsm6ds;
```

To use with the LSM6DS3TR-C + LIS3MDL, uncomment the following lines.

```
#include "Adafruit_LSM6DS3TRC.h"
Adafruit_LSM6DS3TRC lsm6ds;
```

To use with the LIS3MDL + ISM330DHCX, uncomment the following lines.

```
#include "Adafruit_ISM330DHCX.h"
Adafruit_ISM330DHCX lsm6ds;
```

Once you upload the code and open the Serial Monitor (Tools->Serial Monitor) at 1152 00 baud, you will see the current configuration printed, followed by the accelerometer, gyro, and temperature measurements. You should see something similar to this:
Example Code

```c
#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
```
```cpp
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 */
ls67ds.getEvent(&accel, &gyro, &temp);
lis3mdl.getEvent(&mag);

/**< Display the results (acceleration is measured in m/s^2) */
Serial.print("\t\t.Accel 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\t.Gyro 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(" \trad/s ");
Serial.print(" \tZ: ");
Serial.print(gyro.gyro.z, 4);
Serial.println(" \tradians/s ");

/* Display the results (magnetic field is measured in uTesla) */
Serial.print(" \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("\tTemp   :					");
Serial.print(temp.temperature);
Serial.println(" \tdeg C");
Serial.println();
delay(1000);
}

LIS3MDL Arduino Docs

LSM6DS Arduino Docs

Python & CircuitPython

It's easy to use the LIS3MDL + LSM6DSOX, LSM6DS33 and LSM6DS3TR-C sensor combos with Python and CircuitPython, and the Adafruit CircuitPython LIS3MDL () and Adafruit CircuitPython LSM6DS () 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 the board to a microcontroller using I2C. The following shows wiring each board to a Feather M4 Express.
Breakout Board

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)
FeatherWing

Plug the FeatherWing into the Feather.

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](https://circuitpython.org/). Here's the Raspberry Pi wired with 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)
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 Installation of LIS3MDL and LSM6DS Libraries

You’ll need to install the Adafruit CircuitPython LIS3MDL () and the Adafruit CircuitPython LSM6DS () libraries on your CircuitPython board. The LSM6DS library works with the LSM6DSOX, LSM6DS33 and LSM6DS3TR-C.

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 ()

Copy the following folders and file from the bundle to the lib folder on your CIRCUITPY drive:

- `adafruit_lismdl.mpy`
- `adafruit_lsm6ds/
- `adafruit_bus_device/
- `adafruit_register/

Before continuing make sure your board’s lib folder or root filesystem has the adafruit_lis3mdl.mpy, adafruit_lsm6ds/, adafruit_bus_device/, and adafruit_register/ file and folders copied over.

Next connect to the board's serial REPL () so you are at the CircuitPython >>> prompt.
CircuitPython & Python Usage

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

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

```python
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
accel_gyro = LSM6DS(i2c)
mag = LIS3MDL(i2c)
```

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

- **acceleration** - The acceleration forces in the X, Y, and Z axes in m/s²
- **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)

For example, to print out the acceleration, gyro and magnetic measurements use this code:

```python
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/s²\n".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))
```
Full Example Code

```python
# 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)
```

LIS3MDL Python docs

[LIS3MDL Python docs ()](#)

LSM6DS Python Docs

[LSM6DS Python Docs ()](#)

AHRS Fusion of 9 DoF Sensors

[AHRS Fusion of 9 DoF Sensors ()](#)
Downloads

Files

- LIS3MDL datasheet
- LSM6DSOX datasheet
- LSM6DS33 datasheet
- LSM6DS3TR-C Datasheet
- ISM330DHCX datasheet
- LSM6DS33 + LIS3MDL Fritzing object in the Adafruit Fritzing library
- LSM6DSOX + LIS3MDL FeatherWing Fritzing object in the Adafruit Fritzing library
- LSM6DSOX + LIS3MDL breakout Fritzing object in the Adafruit Fritzing Library
- ISM330DHCX + LIS3MDL FeatherWing Fritzing object in the Adafruit Fritzing library
- LSM6DSOX + LIS3MDL EagleCAD files on GitHub
- LSM6DS33 + LIS3MDL EagleCAD files on GitHub
- LSM6DS3TR-C + LIS3MDL EagleCAD PCB files on GitHub
- ISM330DHCX + LIS3MDL EagleCAD files on GitHub
- LSM6DSOX + LIS3MDL models on GitHub
- FeatherWing LSM6DSOX + LIS3MDL models GitHub

LSM6DSOX + LIS3MDL breakout Fab Print and Schematic

![LSM6DSOX + LIS3MDL breakout PCB and Schematic](image-url)
LSM6DSOX + LIS3MDL FeatherWing Fab Print and Schematic
LSM6DS33 + LIS3MDL Fab Print and Schematic
LSM6DS3TR-C + LIS3MDL Fab Print and Schematic
ISM330DHCX + LIS3MDL Fab Print and Schematic