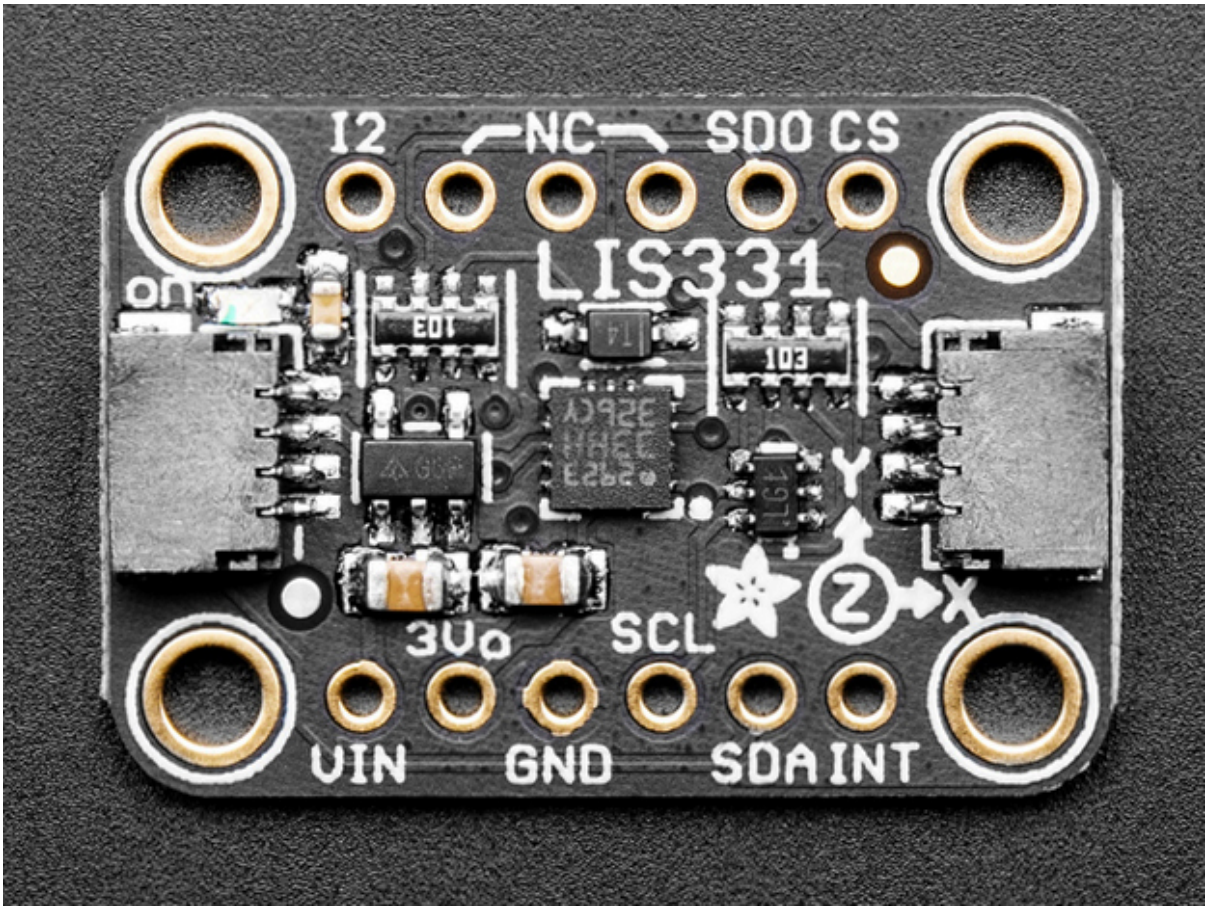
THE DATASHEET OF

ZCKY061

Adafruit H3LIS331 and LIS331 High-g 3-Axis Accelerometers

Created by Bryan Siepert

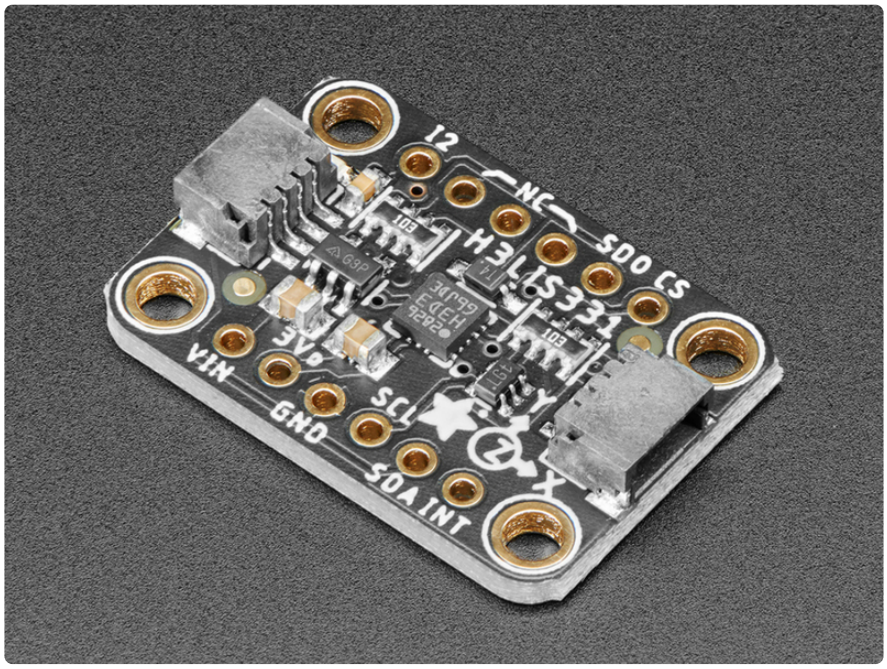


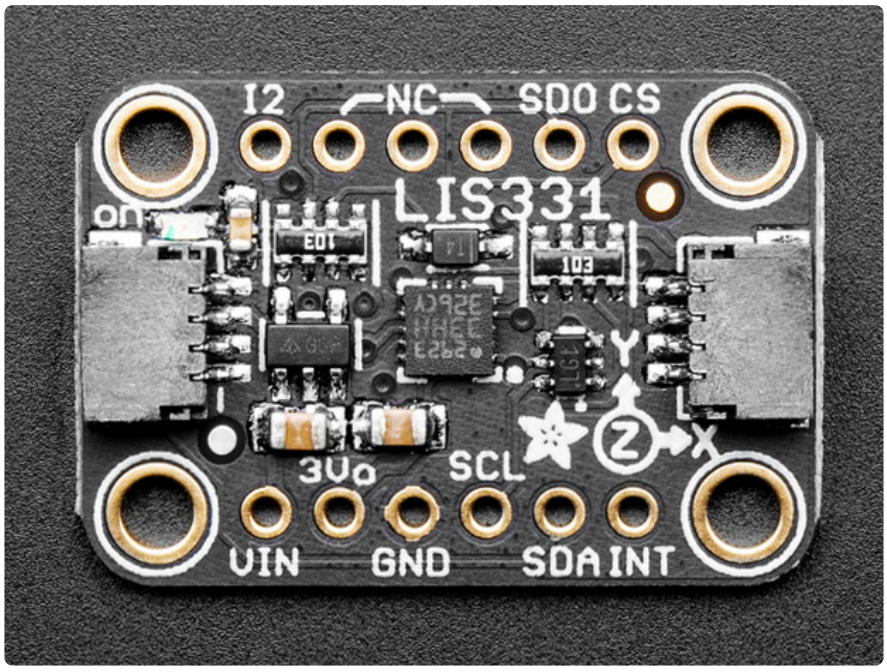
https://learn.adafruit.com/adafruit-h3lis331-and-lis331hh-high-g-3-axis-accelerometers

Last updated on 2025-08-13 04:43:42 PM EDT

Table of Contents

Overview

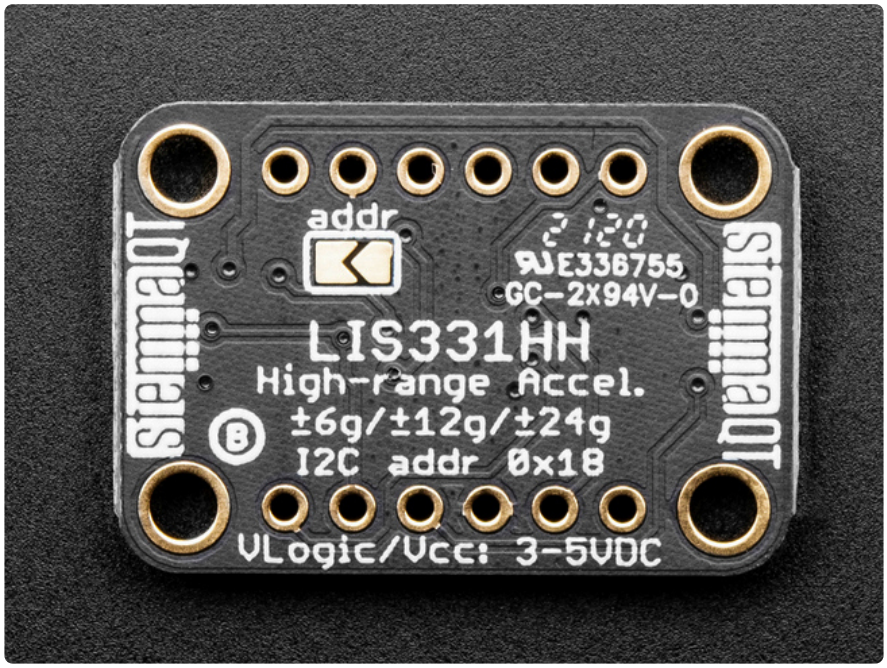


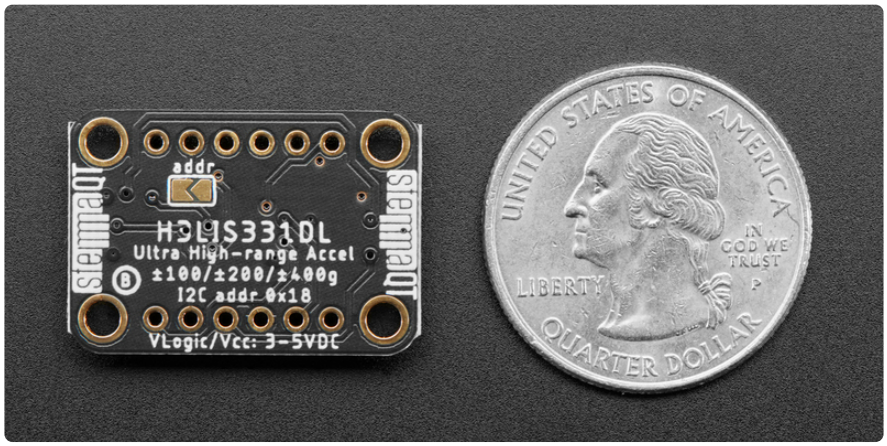


It's not hard to find an accelerometer that can measure accelerations up to 16g, but if you need an accelerometer that can measure even larger amounts of acceleration, your options narrow (The ICM20649 (http://adafru.it/4464) is a great sensor and can measure up to ±30g).

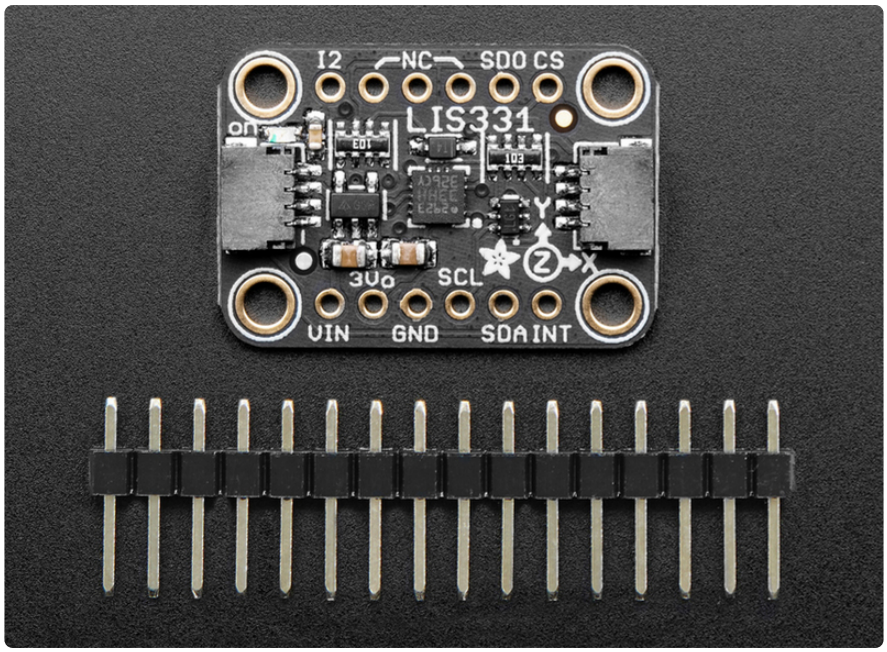
Enter the **LIS331 family** of accelerometers from ST, including the **H3LIS331** and **LIS331 HH**. As their model numbers may suggest, the LIS331s are close cousins to the venerable LIS3DH (http://adafru.it/2809) accelerometer that is on every Circuit

Playground, from the Circuit Playground Classic to the newest Circuit Playground Bluefruit. The LIS331s however can measure a wider range of acceleration values.

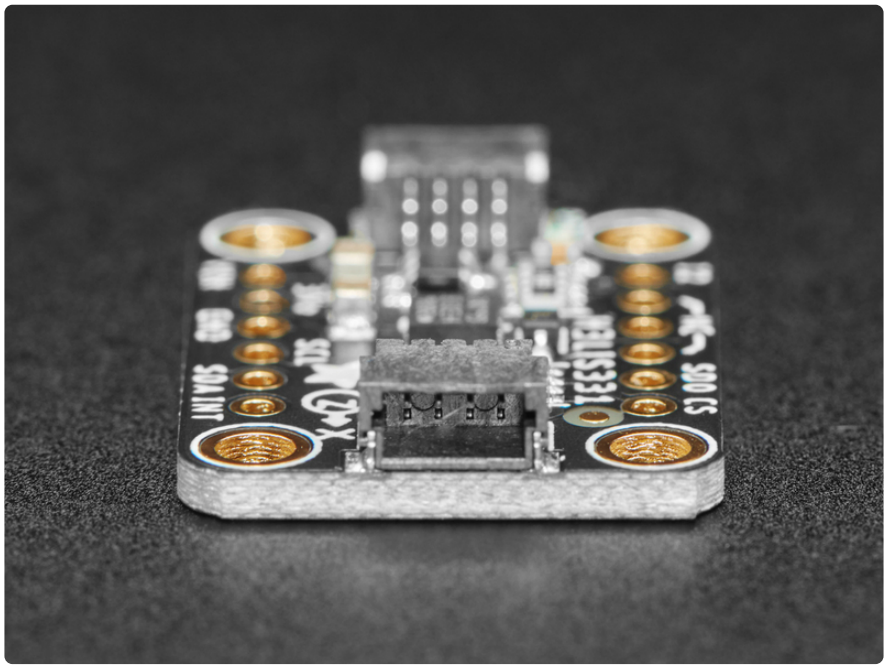




The **LIS331HH** is capable of measuring up to **±24g** on each of its three axes! Should that not be enough for you the **H3LIS331** will certainly have you covered. At the reasonable cost of some signal noise at lower "human level" accelerations, the **H3LIS 331** can measure up to **±400g!** The mind boggles at trying to think of what would need to measure that much acceleration. Perhaps you bought a surplus Sprint missile (https://adafru.it/LBO) to start your own space program, or maybe you have an idea for a rocket sled (https://adafru.it/LBP) based pizza delivery startup. The sky is the limit!



In addition to their substantial measurement capability, the LIS331s have built in and **configurable high-pass and low-pass filters** to adjust the readings to your application. **Adjustable data** rates also allow you to adjust how frequently to take measurements depending on your power budget, and **SPI and I2C** interfaces give them flexibility to allow them to be used in a range of applications.

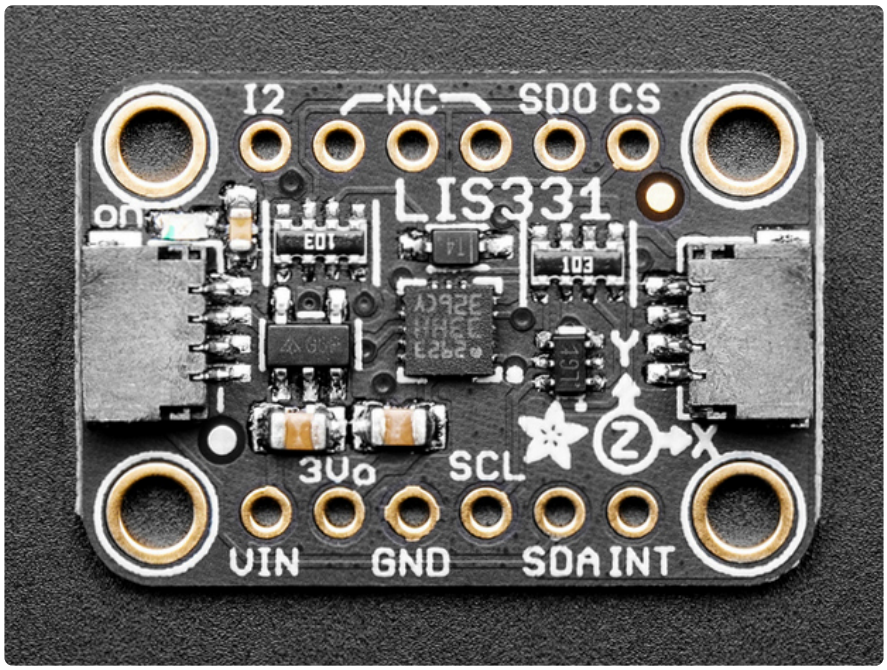


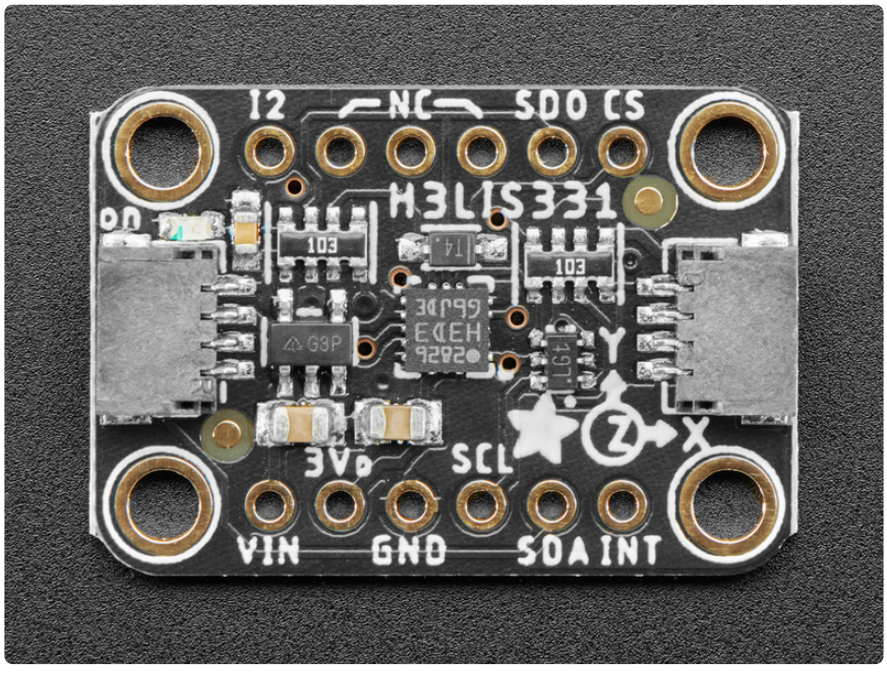
The breakout for the LIS331 family takes one of these little dynamos and puts it on a custom made PCB in the **STEMMA QT** form factor (https://adafru.it/LBQ), making them easy to interface with. The STEMMA QT connectors (https://adafru.it/JqB) on either side are compatible with the SparkFun Qwiic (https://adafru.it/Fpw) I2C connectors. This allows you to make solderless connections between your development board and the LIS331s or to chain them with a wide range of other sensors and accessories

using a **compatible cable** (https://adafru.it/JnB). We've of course broken out all the pins to standard headers and added a voltage regulator and level shifting so allow you to use it with either 3.3V or 5V systems such as the Metro M4 or Arduino Uno respectively.

Fancy as they are, breakouts alone won't get you far, so we've written libraries for **CircuitPython** and **Arduino** along with example code to make them simple to use. The documentation for how to use the libraries as well as wiring diagrams on the following pages will show you how to use them to get you started.

Pinouts





Power Pins

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

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

I2C Pins

- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line. Has a **10K pullup** already on it.
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line. Has a **10K pullup** already on it.

- To use I2C, keep the **CS** pin either disconnected or tied to a high (3-5V) logic level.

- **SDO** - When in I2C mode, this pin can be used for address selection. When connected to **GND** or left open, the address is **0x18** - it can also be connected to 3.3V to set the address to **0x19**

SPI 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 the **S**PI **Cl**oc**k** pin, its an input to the chip
- **SDA** - this is the **S**erial **Da**ta In / **M**icrocontroller **O**ut **S**ensor **I**n pin, for data sent from your processor to the LIS3DH
- **SDO** - this is the **S**erial **D**ata **O**ut / **M**icrocontroller **I**n **S**ensor **O**ut pin, for data sent from the LIS331's to your processor. It's 3.3V logic level out
- **CS** - this is the **C**hip **S**elect pin, drop it low to start an SPI transaction. Its an input to the chip

If you want to connect multiple LIS331's to one microcontroller, have them share the SDI, SDO and SCK pins. Then assign each one a unique CS pin.

Other pins

- **INT, I2 -** The interrupt output pins. You can configure an interrupt to trigger on different events such as data being available to either pin. Consult the datasheet linked on the Downloads page for more information.
- **NC** - These are **No Connect** pins, they are used for mechanical stability only

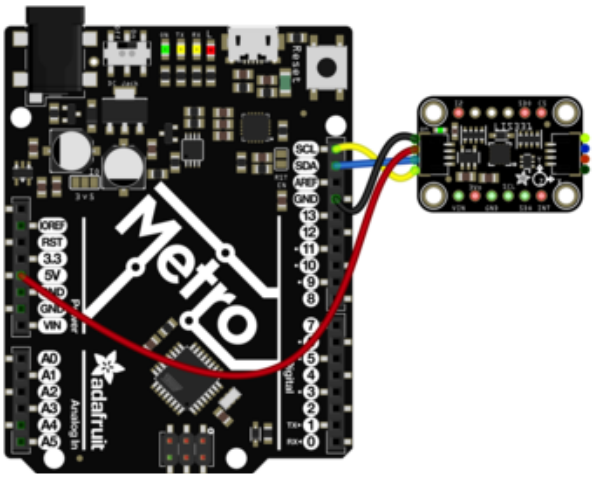
---

Arduino
I2C Wiring

Use this wiring if you want to connect via I2C interface

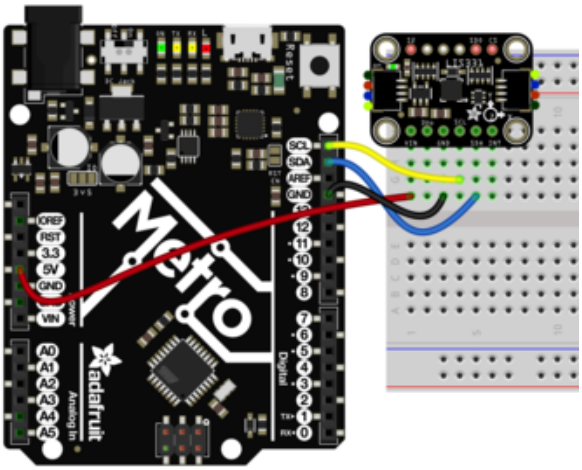
By default, the i2c address is **0x18**. If you add a jumper from **SDO** to **3Vo** the address will change to **0x19**

Here is an example using the STEMMA QT connector for solderless wiring



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**

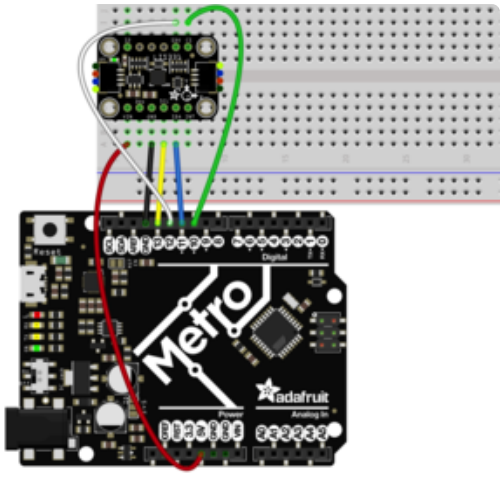
And an example using a breadboard:



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:

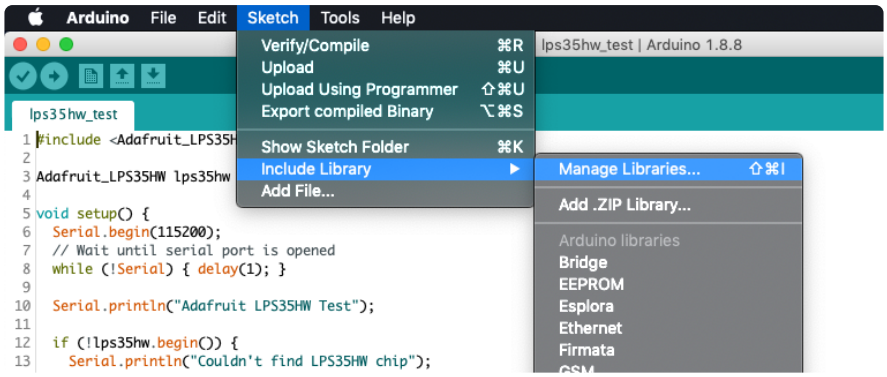


Connect **Vin** to the power supply, 3V or 5V is fine. Use the same voltage that the microcontroller logic is based off of
Connect **GND** to common power/data ground
Connect the **SCK** pin to **Digital #13** but any pin can be used later
Connect the S**DO** pin to **Digital #12** but any pin can be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

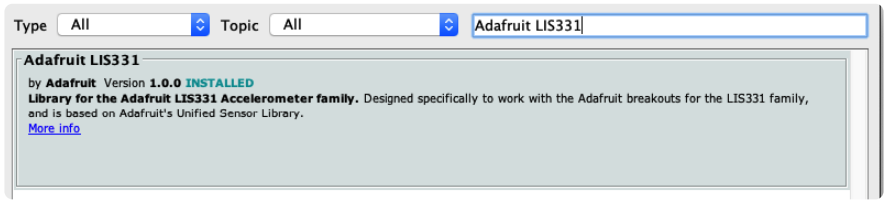
Later on, once we get it working, we can adjust the library to use hardware SPI if you desire, or change the pins to others.

Library Installation

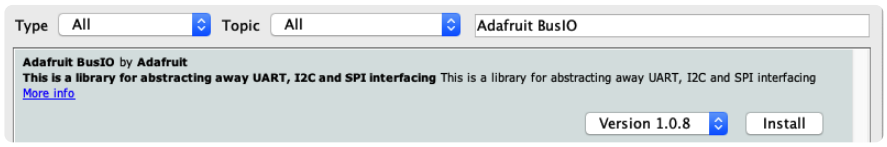
You can install the **Adafruit LIS331 l**ibrary for Arduino using the Library Manager in the Arduino IDE.



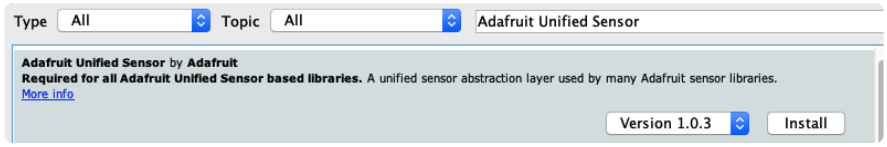
Click the **Manage Libraries ...** menu item, search for **Adafruit LIS331**, and select the **Adafruit LIS331** library:



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 LIS331 -> lis331hh_accel_demo** if you are using the **LIS331HH**

If you are using the **H3LIS331** use **File -> Examples -> Adafruit LIS331 -> h3lis331_accel_demo**

If you are using **SPI**, change the pin names and comment or uncomment the following lines.

```
if (! lis.begin_I2C()) {
  //if (!lis.begin_SPI(LIS331HH_CS)) {
  //if (!lis.begin_SPI(LIS331HH_CS, LIS331HH_SCK, LIS331HH_MISO, LIS331HH_MOSI)) {
```

After opening the demo file for the sensor you are using, upload to your Arduino wired up to the sensor. Once you upload the code, you will see the **acceleration** values for the X, Y, and Z axes being printed when you open the Serial Monitor (**Tools->Serial Monitor**) at **115200 baud**, similar to this:

```
LIS331HH test!
LIS331HH found!
Range set to: 6 g
Data rate set to: 1000 Hz
               X: 0.32         Y: 0.00         Z: 10.14 m/s^2

               X: 0.20         Y: -0.03        Z: 9.80 m/s^2

               X: 0.09         Y: -0.23        Z: 9.65 m/s^2
```

Acceleration is calculated as how much a velocity (meters per second) changes in a set amount of time, normally 1 second. So you will see acceleration values listed in **meters per second per second**, or **meters per second squared**. If space is short you may see **m/s^2**

Normally, sitting on a table, you'll see X and Y are close to **0** and Z will be about 1 **g** or ~9.8 **m/s^2** because the accelerometer is measuring the force of gravity!

You can also move around the board to measure your movements and also try tilting it to see how the gravity 'force' appears at different axes.

```
X: -5.49        Y: -7.56        Z: -16.18 m/s^2

X: 13.16        Y: -0.52        Z: 21.63 m/s^2

X: -14.39       Y: 2.53         Z: 8.79 m/s^2
```

Accelerometer ranges

Accelerometers don't 'naturally' spit out a "meters per second squared" value. Instead, they give you a raw value. In this sensor's case, its a number ranging from -32768 and 32767 (a full 16-bit range). Depending on the sensor range, this number scales between the min and max of the range. E.g. if the accelerometer is set to **+-2g** then 32767 is +2g of force, and -32768 is -2g. If the range is set to **+-16g**, then those two number correlate to +16g and -16g respectively. So knowing the range is key to deciphering the data!

You can set, and get the range with:

```
lis.setRange(LIS331HH_RANGE_6_G);   // 6, 12, or 24 G
```

Not every accelerometer has the same range. If you're using the mega-ranged **H3LIS331** you'll use

```
lis.setRange(H3LIS331_RANGE_100_G);   // 100, 200, or 400 G!
```

Example Code

```
// Basic demo for accelerometer readings from Adafruit LIS331HH

#include <Wire.h>
#include <SPI.h>
#include <Adafruit_LIS331HH.h>
#include <Adafruit_Sensor.h>

// Used for software SPI
#define LIS331HH_SCK 13
#define LIS331HH_MISO 12
#define LIS331HH_MOSI 11
// Used for hardware & software SPI
#define LIS331HH_CS 10

Adafruit_LIS331HH lis = Adafruit_LIS331HH();

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

  Serial.println("LIS331HH test!");

  //if (!lis.begin_SPI(LIS331HH_CS)) {
//  if (!lis.begin_SPI(LIS331HH_CS, LIS331HH_SCK, LIS331HH_MISO, LIS331HH_MOSI)) {
  if (! lis.begin_I2C()) {   // change this to 0x19 for alternative i2c address
    Serial.println("Couldnt start");
    while (1) yield();
  }
  Serial.println("LIS331HH found!");

 lis.setRange(LIS331HH_RANGE_6_G);   // 6, 12, or 24 G
  Serial.print("Range set to: ");
  switch (lis.getRange()) {
    case LIS331HH_RANGE_6_G: Serial.println("6 g"); break;
    case LIS331HH_RANGE_12_G: Serial.println("12 g"); break;
    case LIS331HH_RANGE_24_G: Serial.println("24 g"); break;
  }
  // lis.setDataRate(LIS331_DATARATE_50_HZ);
  Serial.print("Data rate set to: ");
  switch (lis.getDataRate()) {

    case LIS331_DATARATE_POWERDOWN: Serial.println("Powered Down"); break;
    case LIS331_DATARATE_50_HZ: Serial.println("50 Hz"); break;
    case LIS331_DATARATE_100_HZ: Serial.println("100 Hz"); break;
    case LIS331_DATARATE_400_HZ: Serial.println("400 Hz"); break;
    case LIS331_DATARATE_1000_HZ: Serial.println("1000 Hz"); break;
    case LIS331_DATARATE_LOWPOWER_0_5_HZ: Serial.println("0.5 Hz Low Power"); break;
    case LIS331_DATARATE_LOWPOWER_1_HZ: Serial.println("1 Hz Low Power"); break;
    case LIS331_DATARATE_LOWPOWER_2_HZ: Serial.println("2 Hz Low Power"); break;
    case LIS331_DATARATE_LOWPOWER_5_HZ: Serial.println("5 Hz Low Power"); break;
    case LIS331_DATARATE_LOWPOWER_10_HZ: Serial.println("10 Hz Low Power"); break;

  }
}

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

  /* Display the results (acceleration is measured in m/s^2) */
  Serial.print("\t\tX: "); Serial.print(event.acceleration.x);
  Serial.print(" \tY: "); Serial.print(event.acceleration.y);
  Serial.print(" \tZ: "); Serial.print(event.acceleration.z);
  Serial.println(" m/s^2 ");
```

```
  Serial.println();

  delay(1000);
}
```

Arduino Docs

Arduino Docs (https://adafru.it/Lcw)

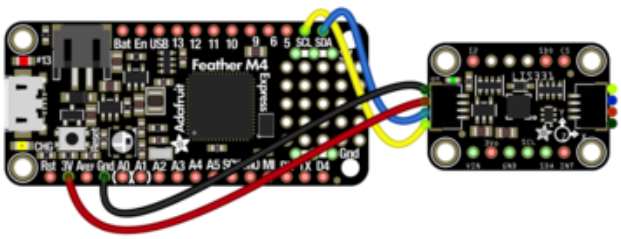
Python & CircuitPython

It's easy to use the **LIS331** sensors with Python or CircuitPython, and the Adafruit CircuitPython LIS331 (https://adafru.it/LBR) module. This module allows you to easily write Python code that reads acceleration from the either **LIS331HH** or **H3LIS331** sensors.

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 (https://adafru.it/BSN).

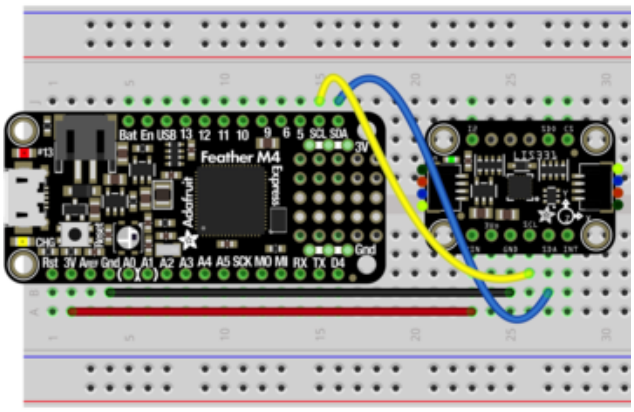
CircuitPython Microcontroller Wiring

First wire up a LIS331 to your board exactly as shown on the previous pages for Arduino. You can use either I2C or SPI wiring, although it's recommended to use I2C for simplicity. Here's an example of wiring a Feather M4 to the sensor with I2C using one of the handy **STEMMA QT** (https://adafru.it/Ft4) 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 the standard **0.100" pitch** headers to wire it up on a breadboard:

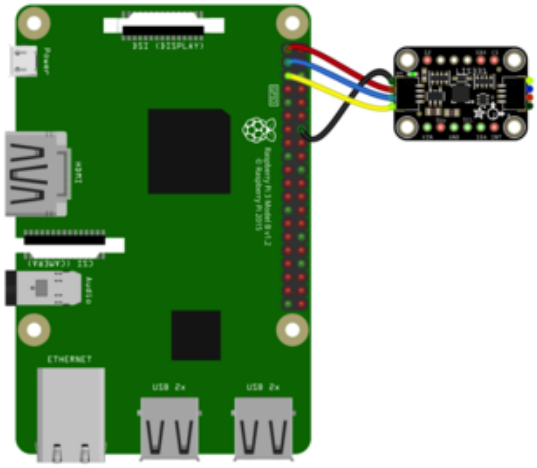


Board 3V to **sensor VIN (red wire)**
Board GND to **sensor GND (black wire)**
Board SCL to **sensor SCL (yellow wire)**
Board SDA to **sensor SDA (blue wire)**

Python Computer Wiring

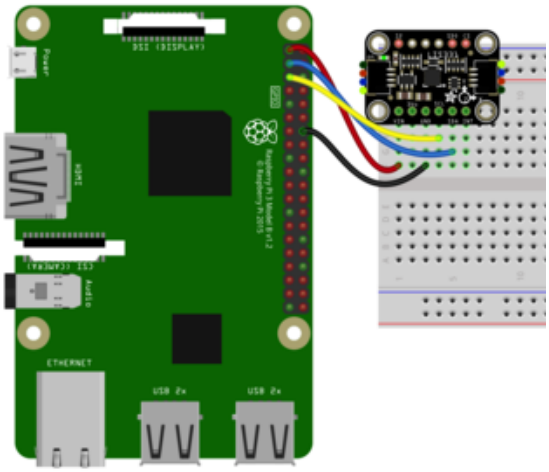
Since there's dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, [please visit the guide for CircuitPython on Linux to see whether your platform is supported](https://adafru.it/BSN) (https://adafru.it/BSN).

Here's the Raspberry Pi wired to the sensor using I2C and a **STEMMA QT** (https://adafru.it/Ft4) connector:



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)**

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



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 LIS331 Library

You'll need to install the Adafruit CircuitPython LIS331 (https://adafru.it/LBR) library on your CircuitPython board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Amd) 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 (https://adafru.it/ENC). Our CircuitPython starter guide has a great page on how to install the library bundle (https://adafru.it/ABU).

> Unfortunately the CircuitPython LIS331 library is currently too large for M0 devices such as the Circuit Playground Express, ItsyBitsy M0, Gemma M0, and Trinket M0. Please use an M4 device such as the Metro M4 until we can shrink it down

Before continuing make sure your board's **lib** folder or root filesystem has the **adafruit_lis331.mpy, adafruit_bus_device**, and **adafruit_register** files and folders copied over.

Next connect to the board's serial REPL (https://adafru.it/Awz)so you are at the CircuitPython `>>>` prompt.

Python Installation of LIS331 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 (https://adafru.it/BSN)!

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

- `sudo pip3 install adafruit-circuitpython-lis331`

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 acceleration measurements from the board's Python REPL.

> These examples will show using the LIS331HH. To use the H3LIS331 simple follow the instructions in the example to uncomment the line for the H3LIS331 and comment out the LIS331HH line

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

```
import time
import board
import adafruit_lis331

i2c = board.I2C()
# uncomment this line and comment out the one after if using the H3LIS331
# lis = adafruit_lis331.H3LIS331(i2c)
lis = adafruit_lis331.LIS331HH(i2c)
```

```
>>> import time
>>> import board
>>> import adafruit_lis331
>>> i2c = board.I2C()
>>> lis = adafruit_lis331.LIS331HH(i2c)
```

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

- **acceleration** - The x, y, z acceleration values returned in a 3-tuple and are in m / s ^ 2..

For example to print the acceleration values for all three axes:

```
>>> print("Acceleration : X: %.2f, Y:%.2f, Z:%.2f ms^2" % lis.acceleration)
Acceleration : X: -0.23, Y:0.11, Z:9.65 ms^2
```

That's all it takes to get measuring acceleration using the LIS331 accelerometer breakouts!

Full Example Code

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

import time

import board

import adafruit_lis331

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
lis = adafruit_lis331.LIS331HH(i2c)

while True:
    print(
        f"Acceleration : X: {lis.acceleration[0]:.2f}, Y:{lis.acceleration[1]:.2f},
Z:{lis.acceleration[2]:.2f} ms^2"
    )
    time.sleep(0.1)
```

Python Docs

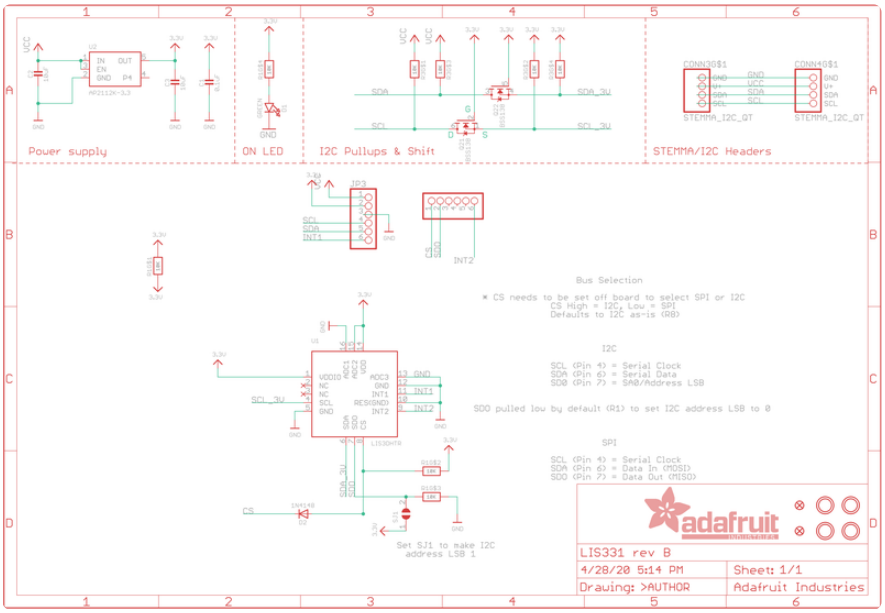
Python Docs (https://adafru.it/Lcx)

Downloads

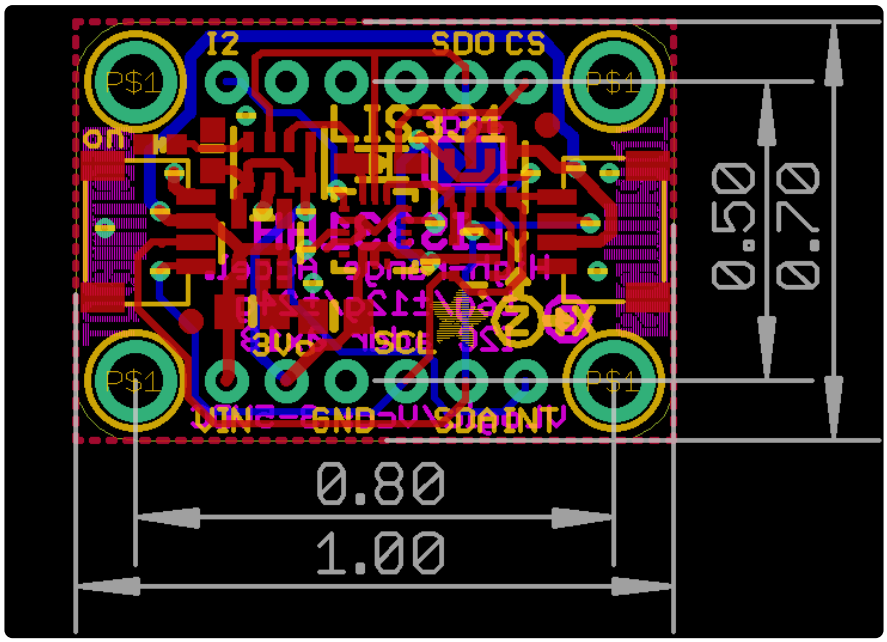
Files

- LIS331HH Datasheet (https://adafru.it/1apO)
- H3LIS331 Datasheet (https://adafru.it/1apP)
- EagleCAD files on GitHub (https://adafru.it/LBU)
- Fritzing object in Adafruit Fritzing Library (https://adafru.it/LBV)

Schematic



Fab Print



Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View ZCKY061 on WIN SOURCE
- Manufacturer Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management