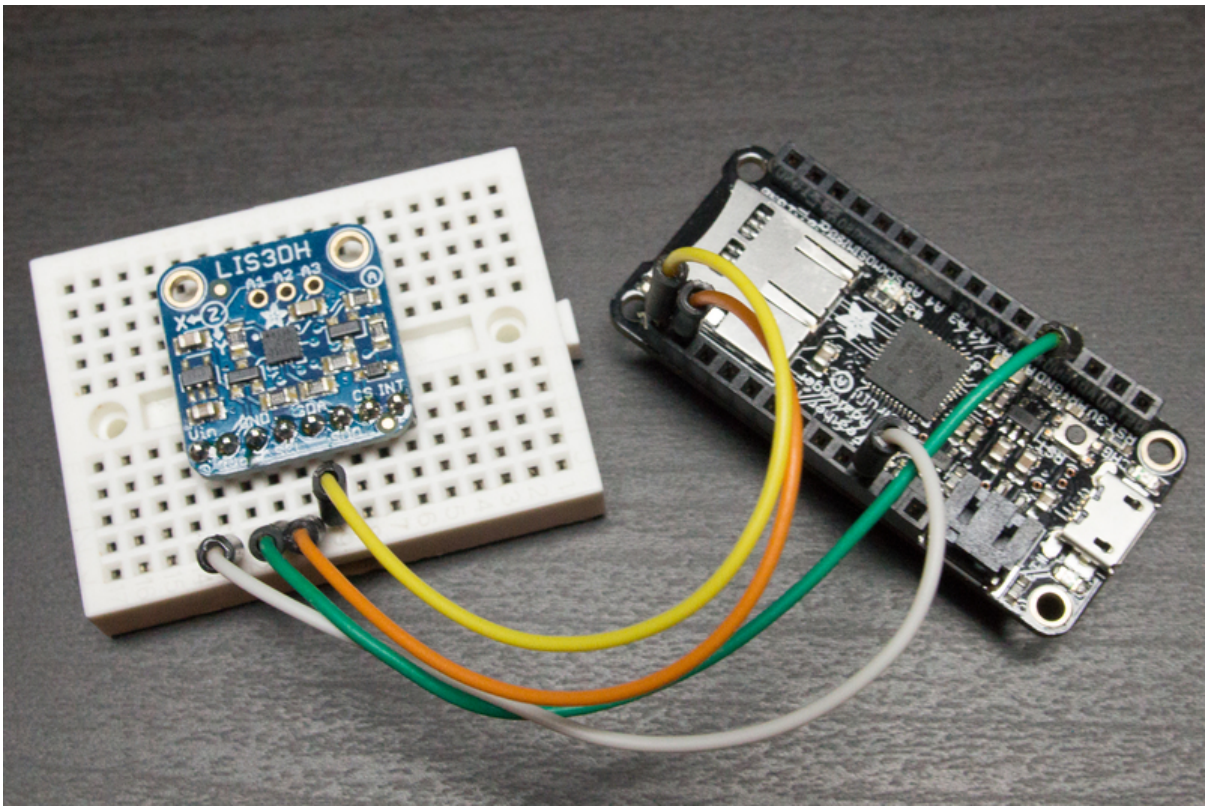
CircuitPython Hardware: LIS3DH Accelerometer

Created by Tony DiCola



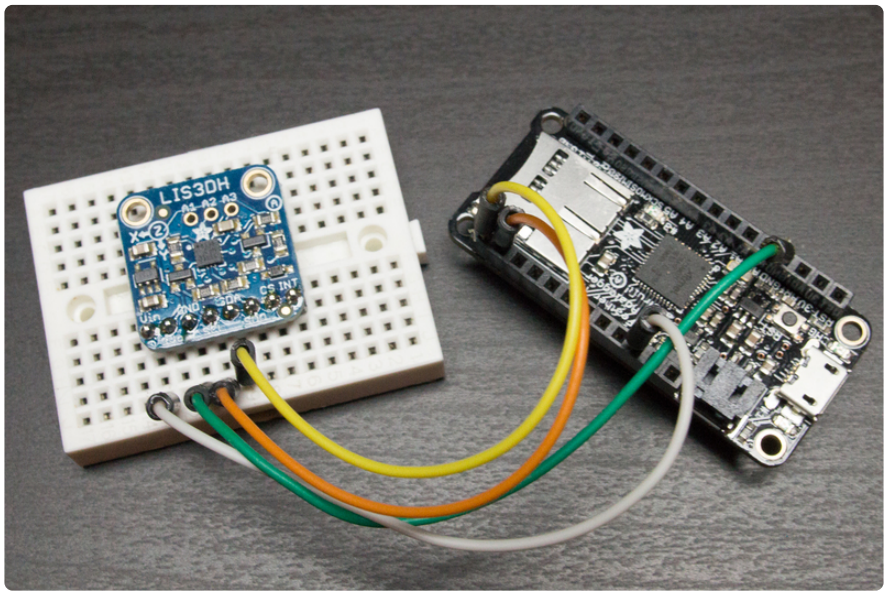
https://learn.adafruit.com/circuitpython-hardware-lis3dh-accelerometer

Last updated on 2024-06-03 02:05:14 PM EDT

Table of Contents

Overview

The examples in this guide are no longer supported. Check out the LIS3DH sensor guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-lis3dh-triple-axis-accelerometer-breakout



Acceleration makes the world go around--literally! It's the force that causes movement like a car accelerating away from a stop light or an object falling to the ground from gravity when dropped. Accelerometers are small sensors that can detect the force of acceleration and are great for detecting motion and orientation. The LIS3DH triple-axis accelerometer (http://adafru.it/2809) in particular is an inexpensive and easy to use accelerometer with features like X, Y, Z axis acceleration and click detection. This guide will show you how to wire the LIS3DH to a board like the ESP8266 or SAMD21/M0 and start reading acceleration values from it in CircuitPython!

Before you get started you'll want to be familiar with CircuitPython, MicroPython, and the LIS3DH by reading these guides:

- MicroPython Basics: What is MicroPython? (https://adafru.it/pXa)
- MicroPython Basics: How to Load MicroPython on a Board (https://adafru.it/pNB)
- MicroPython Basics: Load Files & Run Code (https://adafru.it/s1f)
- Adafruit LIS3DH Triple-Axis Accelerometer Breakout (https://adafru.it/uBr)

Continue on to learn about the hardware needed to follow this guide.

Hardware

The examples in this guide are no longer supported. Check out the LIS3DH sensor guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-lis3dh-triple-axis-accelerometer-breakout

Parts

You'll need the following hardware to follow this guide:

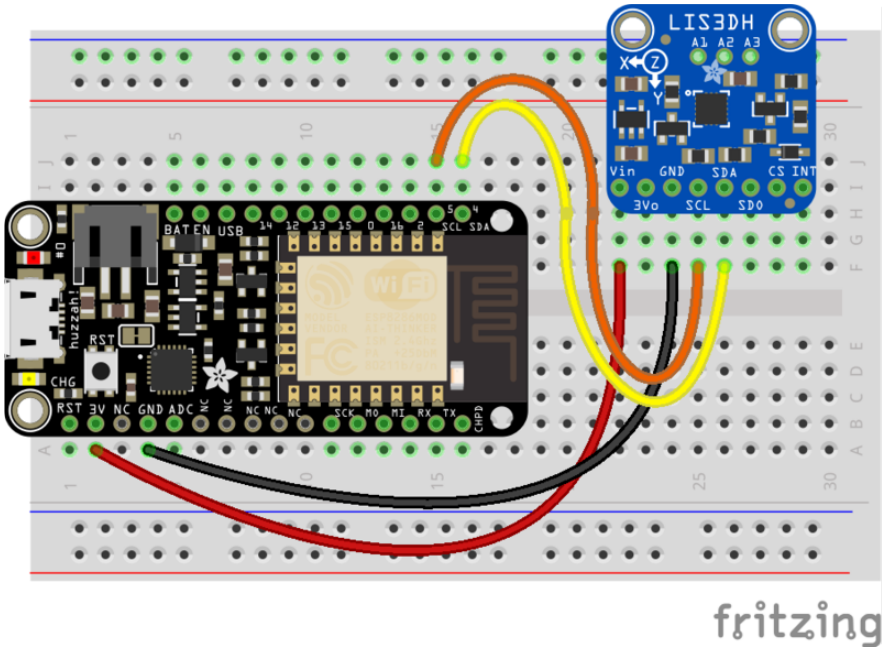
- **Board running CircuitPython.** Currently ESP8266 or SAMD21-based boards are supported by CircuitPython. The Feather HUZZAH ESP8266 (http://adafru.it/2821) or Feather M0 boards (http://adafru.it/2772) are great options that can easily be connected to an accelerometer. See the guide on loading MicroPython & CircuitPython firmware (https://adafru.it/pWF) on a board for details on loading CircuitPython. **Remember you want to run Adafruit CircuitPython (https://adafru.it/tBa) and not MicroPython to follow this guide!**
- LIS3DH accelerometer breakout board. **(http://adafru.it/2809)**
- Breadboard **(http://adafru.it/64)**, hookup wires **(http://adafru.it/153)**, **and** soldering tools **(http://adafru.it/136)**. You'll need to solder headers to the accelerometer breakout, be sure to see the guide to excellent soldering (https://adafru.it/dxy) if you're new to it.

Start by following the LIS3DH breakout guide (https://adafru.it/uBr) to assemble and test the board. Then continue on below to learn how to wire it to a Feather for use with CircuitPython.

Wiring

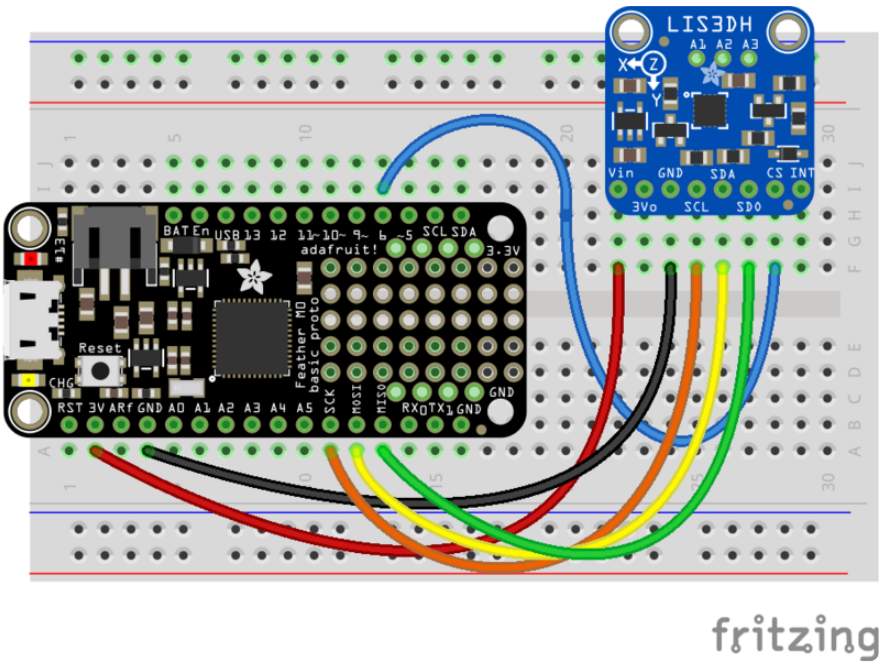
There are two ways to connect the LIS3DH accelerometer to a board, either with an I2C or SPI connection. The I2C connection requires just a couple data lines and is the recommended way to use the accelerometer. However if for some reason you can't use I2C the board also supports a SPI interface which uses a few more data lines. See below for details on wiring up either interface to your board.

I2C Wiring



- **LIS3DH Vin** to **board 3V (or 5V) output** - red wire.
- **LIS3DH GND** to **board GND/ground** - black wire.
- **LIS3DH SCL** to **board SCL (I2C clock)** - orange wire.
- **LIS3DH SDA** to **board SDA (I2C data)** - yellow wire.

SPI Wiring



- **LIS3DH Vin** to **board 3V (or 5V) output** - red wire.
- **LIS3DH GND** to **board GND/ground** - black wire.
- **LIS3DH SCL** to **board SCK (SPI clock)** - orange wire.
- **LIS3DH SDA** to **board MOSI (SPI microcontroller out/sensor in)** - yellow wire.
- **LIS3DH SDO** to **board MISO (SPI microcontroller in/sensor out)** - green wire.
- **LIS3DH CS** to **board pin #6 (or any free digital I/O pin)** - blue wire.

Once the LIS3DH is wired to your board continue on to learn how to install the CircuitPython modules necessary to control it from your Python code.

Software

The examples in this guide are no longer supported. Check out the LIS3DH sensor guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-lis3dh-triple-axis-accelerometer-breakout

Adafruit CircuitPython Module Install

To use the LIS3DH with your Adafruit CircuitPython (https://adafru.it/tCy) board you'll need to install the Adafruit_CircuitPython_LIS3DH (https://adafru.it/uBs) module on

your board. **Remember this module is for Adafruit CircuitPython firmware and not MicroPython.org firmware!**

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/tBa) for your board (note you need to be running the **CircuitPython 1.0.0-rc1 or greater** release for this library and examples).

Next download the latest `adafruit-circuitpython-lis3dh*.zip` file from the releases page of the Adafruit_CircuitPython_LIS3DH GitHub repository (https://adafru.it/uBt) that matches your version of CircuitPython. You'll need to unzip this file and copy the entire **adafruit_lis3dh** directory to the board's root filesystem.

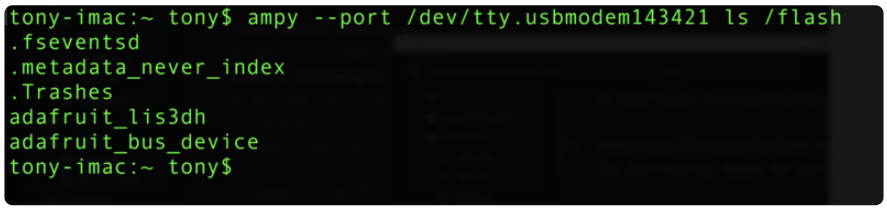
This guide is for 3.0.0+ of the LIS3DH library. Make sure to grab the latest version!

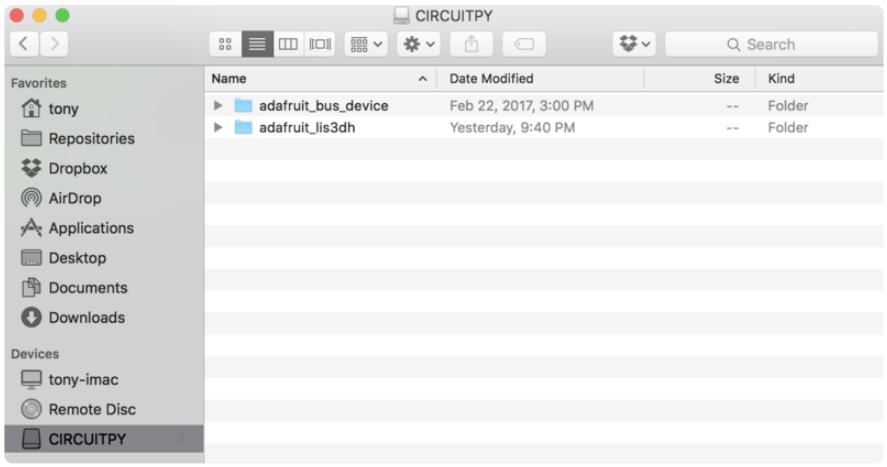
If your board supports USB mass storage, like the SAMD21 CircuitPython port, then simply drag the files to the board's file system. **Note on boards without external SPI flash, like a Feather M0 or Trinket/Gemma M0, you might run into issues on Mac OSX with hidden files taking up too much space when drag and drop copying, see this page for a workaround (https://adafru.it/u1d).**

If your board doesn't support USB mass storage, like the ESP8266, then use a tool like ampy to copy the file to the board (https://adafru.it/s1f). You can use the latest version of ampy and its new directory copy command (https://adafru.it/q2A) to easily move module directories to the board.

In addition you'll need the Adafruit CircuitPython Bus Device (https://adafru.it/u0b) module installed on your board. Just like installing the LIS3DH module as mentioned above, download the latest release .zip file and copy the folder inside it to the board's root filesystem.

Before continuing make sure your board's root filesystem has the **adafruit_lis3dh**, and **adafruit_bus_device** folders/modules copied over.





Examples & Usage

To learn how to use the LIS3DH module code you can look at a few examples included with the library (https://adafru.it/uBu):

- lis3dh_simpletest.py **(https://adafru.it/C6P)** - This example demonstrates reading the X, Y, Z axis of the accelerometer and prints their values (in gravities or Gs).
- adc.py **(https://adafru.it/uBw)** - This example demonstrates reading the analog to digital converters on the board.
- tap.py **(https://adafru.it/C6Q)** - This example demonstrates detecting double taps.

Intialization

In all of the examples you'll notice they have the same initialization code at the top. This is how you initialize the I2C or SPI bus (depending on how you have the board wired to your hardware) and setup the LIS3DH module:

```
import board
import adafruit_lis3dh

# Uncomment _one_ of the hardware setups below depending on your wiring:

# Hardware I2C setup:
import busio
i2c = busio.I2C(board.SCL, board.SDA)
lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c)

# Hardware I2C setup on CircuitPlayground Express:
# import busio
# i2c = busio.I2C(board.ACCELEROMETER_SCL, board.ACCELEROMETER_SDA)
# lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c, address=0x19)

# Software I2C setup:
```

```
#import bitbangio
#i2c = bitbangio.I2C(board.SCL, board.SDA)
#lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c)

# Hardware SPI setup:
#import busio
#import digitalio
#spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
#cs = digitalio.DigitalInOut(board.D6)  # Set to appropriate CS pin!
#lis3dh = adafruit_lis3dh.LIS3DH_SPI(spi, cs)
```

Notice that there are four example ways to initialize the library. The first way is uncommented and shows initializing the library with a hardware I2C configuration. This is perfect SAMD21-based boards like the Feather M0 which support a hardware I2C interface.

The second is very similar to the first except it shows hardware I2C setup on the CircuitPlayground Express where the LIS3D is built in.

The third way is with a software I2C configuration. The only difference is that the bitbangio module is used in place of the busio module. This can be useful to run I2C on pins without hardware support. (On boards with no hardware support, such as the ESP8266, busio will automatically bitbang I2C.)

The fourth way is with a hardware SPI configuration. Notice here you must specify a CS or chip select pin as one of the digital IO pins on your board. Be sure to set this pin to the right value for your wiring.

Update the example code so that **only one** of the initialization options is uncommented.

Accelerometer Usage

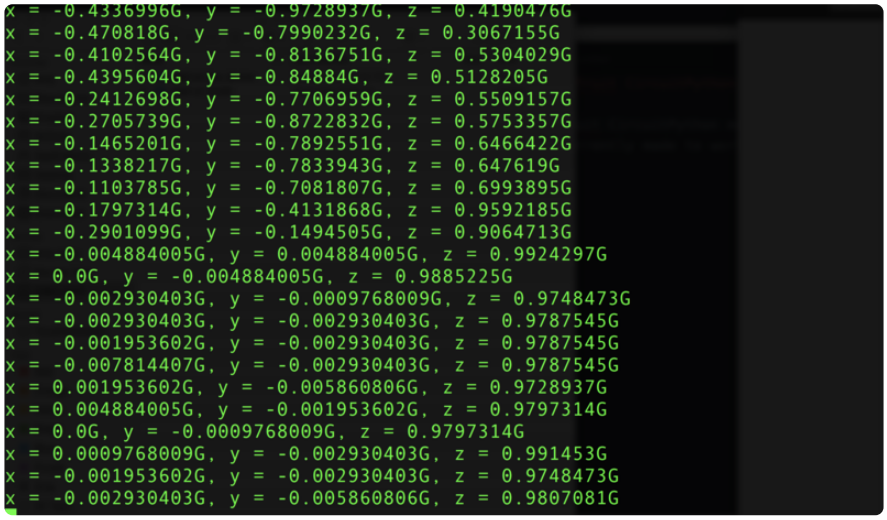
Update the lis3dh_simpletest.py example (https://adafru.it/C6P) initialization as mentioned above and then use a tool like ampy to run it on your board. You'll want to run the example with the **-n** option so that ampy runs the script and doesn't wait for any output, like:

```
ampy --port /dev/serial/port run -n lis3dh_simpletest.py
```

(change the --port value to the name of your board's serial port)

Now open the board's serial terminal and you should see the accelerometer X, Y, Z values printed a few times a second. Try moving the board around and notice how

the values change! The values are in gravities or Gs which are common units for measuring acceleration (1G is ~9.8m/s^2, or typical Earth gravity).



If you examine the lis3dh_simpletest.py code (https://adafru.it/C6P) you can see how it initializes and reads the accelerometer values:

```
# Set range of accelerometer (can be RANGE_2_G, RANGE_4_G, RANGE_8_G or RANGE_16_G).
lis3dh.range = adafruit_lis3dh.RANGE_2_G

# Loop forever printing accelerometer values
while True:
    # Read accelerometer values (in m / s ^ 2).  Returns a 3-tuple of x, y,
    # z axis values.
    x, y, z = lis3dh.acceleration
    print('x = {}G, y = {}G, z = {}G'.format(x / 9.806, y / 9.806, z / 9.806))
    # Small delay to keep things responsive but give time for interrupt processing.
    time.sleep(0.1)
```

First the range of the accelerometer is set by changing the **range** property on the LIS3DH object. Notice how you can set it to one of four possible ranges, where a 2G range gives you a lot of accuracy in a small range vs. up to a 16G range with less accuracy over a much wider range. You'll need to pick the right range for your usage needs, although starting with a simple 2G range is a smart idea.

Next in the main loop you can see the **acceleration** property is read. This property is a 3-tuple with the X, Y, Z acceleration values that were read by the sensor.
Remember these values are in meters per second squared so you might need to convert to other units depending on your needs.

That's all there is to reading the accelerometer values!

Tap Detection Usage

Update the tap.py example (https://adafru.it/C6Q) initialization as mentioned above and then use a tool like ampy to run it on your board. You'll want to run the example with the **-n** option so that ampy runs the script and doesn't wait for any output, like:

```
ampy --port /dev/serial/port run -n tap.py
```

(change the --port value to the name of your board's serial port)

Now open the board's serial terminal and double tap on the LIS3DH chip with your finger. You should see the terminal print out messages when it detects a tap event. By default the example will try to detect double taps:

If you examine the tap.py code (https://adafru.it/uBx) you'll see how it configures and detects double tap events:

```
# Set range of accelerometer (can be RANGE_2_G, RANGE_4_G, RANGE_8_G or RANGE_16_G).
lis3dh.range = adafruit_lis3dh.RANGE_2_G

# Set tap detection to double taps.  The first parameter is a value:
#  - 0 = Disable tap detection.
#  - 1 = Detect single taps.
#  - 2 = Detect double taps.
# The second parameter is the threshold and a higher value means less sensitive
# tap detection.  Note the threshold should be set based on the range above:
#  - 2G = 40-80 threshold
#  - 4G = 20-40 threshold
#  - 8G = 10-20 threshold
#  - 16G = 5-10 threshold
lis3dh.set_tap(2, 80)

# Loop forever printing if a double tap is detected. A single double tap may cause
multiple
# `tapped` reads to return true so we make sure the last once was False.
last_tap = False
while True:
    tap = lis3dh.tapped
    if tap and not last_tap:
        print('Double tap!')
    last_tap = tap
```

Like with reading the accelerometer the **range** property is first set to adjust the range of acceleration the sensor can read. Then notice the **set_tap** function is called and passed a couple parameters:

- **Type of tap event to detect** - Set this to 0 to disable tap detection, 1 to detect only single taps, and 2 to detect only double tap events.
- **Tap threshold** - This is a number that is the threshold for tap detection (see the LIS3DH datasheet (https://adafru.it/uBy) for exact details on its meaning). In general this value depends on the range that was set for the accelerometer. See the example values mentioned in the comments and try setting it appropriately in your code.

In addition the **set_tap** function can take the following optional keyword parameters to further adjust the tap detection. See the LIS3DH datasheet (https://adafru.it/uBy) for more details on what they mean and how they change the detection. (Note that the datasheet calls tap detection "click".)

- **time_limit** - This is the TIMELIMIT register value and defaults to 10.
- **time_latency** - This is the TIMELATENCY register value and defaults to 20.
- **time_window** - This is the TIMEWINDOW register value and defaults to 255.

Once the **set_tap** function is called to configure tap detection, the main loop below it will read the `tapped` attribute to get the current tap detection state. It will be True when a double tap recently occurred, even if we already read that it happened. So, we only print when the `tapped` changes from `False` to `True` by remembering what it was last time in `last_tap`.

That's all there is to detecting tap events!

Analog to Digital Converter Usage

Update the adc.py example (https://adafru.it/uBw) initialization as mentioned above and then use a tool like ampy to run it on your board. You'll want to run the example with the **-n** option so that ampy runs the script and doesn't wait for any output, like:

```
ampy --port /dev/serial/port run -n adc.py
```

(change the --port value to the name of your board's serial port)

Now open the board's serial terminal and notice every second the A1 ADC channel value is printed, both as a raw and millivolt output:

Remember the ADC on the LIS3DH is somewhat limited and can only measure voltages in the range of ~900mV to ~1200mV!

If you examine the adc.py code (https://adafru.it/uBw) you'll see how it configures and reads the ADC:

```
# Loop forever printing ADC readings.
while True:
    # Read raw ADC value.  Specify which ADC to read: 1, 2, or 3.
    adc1_raw = lis3dh.read_adc_raw(1)
    # Or read the ADC value in millivolts:
    adc1_mV = lis3dh.read_adc_mV(1)
    print('ADC 1 = {} ({} mV)'.format(adc1_raw, adc1_mV))
    time.sleep(1)
```

The **read_adc_raw** function is called and passed in the number of the ADC channel to read (1, 2, or 3). This function returns the raw value returned by the ADC register, see the LIS3DH guide (https://adafru.it/uBz) and datasheet (https://adafru.it/uBy) for more details on its meaning.

As a convenience the **read_adc_mV** function can also be called and passed the number of the ADC channel to read. This function converts the raw reading into a millivolt value and returns it. Remember only voltages in the range of ~900mV to ~1200mV can be read by the LIS3DH ADC!

That's all there is to reading the analog to digital converter on the LIS3DH!