
AdafruitADXL34x Library Documentation

Release 1.0

Bryan Siepert

Jul 09, 2020

Contents

1	Dependencies	3
1.1	Installing from PyPI	3
2	Usage Example	5
3	Contributing	7
4	Documentation	9
5	Table of Contents	11
5.1	Simple test	11
5.2	Motion detection	11
5.3	Freefall detection	12
5.4	Tap detection	12
5.5	adafruit_adxl34x	13
5.5.1	Implementation Notes	13
6	Indices and tables	17
	Python Module Index	19
	Index	21

A CircuitPython driver for the ADXL34x family of accelerometers

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).

1.1 Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally [from PyPI](#). To install for current user:

```
pip3 install adafruit-circuitpython-adxl34x
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-adxl34x
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-adxl34x
```


CHAPTER 2

Usage Example

```
import time
import board
import busio
import adafruit_adxl34x
i2c = busio.I2C(board.SCL, board.SDA)
accelerometer = adafruit_adxl34x.ADXL345(i2c)
while True:
    print("%f %f %f"%accelerometer.acceleration)
    time.sleep(1)
```


CHAPTER 3

Contributing

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

CHAPTER 4

Documentation

For information on building library documentation, please check out [this guide](#).

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/adxl34x_simpletest.py

```
1 import time
2 import board
3 import busio
4 import adafruit_adxl34x
5
6 i2c = busio.I2C(board.SCL, board.SDA)
7
8 # For ADXL343
9 accelerometer = adafruit_adxl34x.ADXL343(i2c)
10 # For ADXL345
11 # accelerometer = adafruit_adxl34x.ADXL345(i2c)
12
13 while True:
14     print("%f %f %f" % accelerometer.acceleration)
15     time.sleep(0.2)
```

5.2 Motion detection

Use the accelerometer to detect motion.

Listing 2: examples/adxl34x_motion_detection_test.py

```
1 import time
2 import board
3 import busio
```

(continues on next page)

(continued from previous page)

```

4 import adafruit_adxl34x
5
6 i2c = busio.I2C(board.SCL, board.SDA)
7
8 # For ADXL343
9 accelerometer = adafruit_adxl34x.ADXL343(i2c)
10 # For ADXL345
11 # accelerometer = adafruit_adxl34x.ADXL345(i2c)
12
13 accelerometer.enable_motion_detection()
14 # alternatively you can specify the threshold when you enable motion detection for_
   ↳ more control:
15 # accelerometer.enable_motion_detection(threshold=10)
16
17 while True:
18     print("%f %f %f" % accelerometer.acceleration)
19
20     print("Motion detected: %s" % accelerometer.events["motion"])
21     time.sleep(0.5)

```

5.3 Freefall detection

Use the accelerometer to detect when something is dropped.

Listing 3: examples/adxl34x_freefall_detection_test.py

```

1 import time
2 import board
3 import busio
4 import adafruit_adxl34x
5
6 i2c = busio.I2C(board.SCL, board.SDA)
7
8 # For ADXL343
9 accelerometer = adafruit_adxl34x.ADXL343(i2c)
10 # For ADXL345
11 # accelerometer = adafruit_adxl34x.ADXL345(i2c)
12
13 accelerometer.enable_freefall_detection()
14 # alternatively you can specify attributes when you enable freefall detection for_
   ↳ more control:
15 # accelerometer.enable_freefall_detection(threshold=10,time=25)
16
17 while True:
18     print("%f %f %f" % accelerometer.acceleration)
19
20     print("Dropped: %s" % accelerometer.events["freefall"])
21     time.sleep(0.5)

```

5.4 Tap detection

The accelerometer can also be configured to detect taps.

Listing 4: examples/adxl34x_tap_detection_test.py

```

1 import time
2 import board
3 import busio
4 import adafruit_adxl34x
5
6 i2c = busio.I2C(board.SCL, board.SDA)
7
8 # For ADXL343
9 accelerometer = adafruit_adxl34x.ADXL343(i2c)
10 # For ADXL345
11 # accelerometer = adafruit_adxl34x.ADXL345(i2c)
12
13 accelerometer.enable_tap_detection()
14 # you can also configure the tap detection parameters when you enable tap detection:
15 # accelerometer.enable_tap_detection(tap_count=2, threshold=20, duration=50)
16
17 while True:
18     print("%f %f %f" % accelerometer.acceleration)
19
20     print("Tapped: %s" % accelerometer.events["tap"])
21     time.sleep(0.5)

```

5.5 adafruit_adxl34x

A driver for the ADXL34x 3-axis accelerometer family

- Author(s): Bryan Siepert

Based on drivers by K. Townsend and Tony DiCola

5.5.1 Implementation Notes

Hardware: <https://www.adafruit.com/product/1231>

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

class `adafruit_adxl34x.ADXL343` (*i2c*, *address*=83)
Stub class for the ADXL343 3-axis accelerometer

class `adafruit_adxl34x.ADXL345` (*i2c*, *address*=83)
Driver for the ADXL345 3 axis accelerometer

Parameters

- **i2c_bus** (*I2C*) – The I2C bus the ADXL345 is connected to.
- **address** – The I2C device address for the sensor. Default is 0x53.

acceleration

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

data_rate

The data rate of the sensor.

disable_freelfall_detection()

Disable freefall detection

disable_motion_detection()

Disable motion detection

disable_tap_detection()

Disable tap detection

enable_freelfall_detection(*, threshold=10, time=25)

Freefall detection parameters:

Parameters

- **threshold** (*int*) – The value that acceleration on all axes must be under to register as dropped. The scale factor is 62.5 mg/LSB.
- **time** (*int*) – The amount of time that acceleration on all axes must be less than **threshold** to register as dropped. The scale factor is 5 ms/LSB. Values between 100 ms and 350 ms (20 to 70) are recommended.

If you wish to set them yourself rather than using the defaults, you must use keyword arguments:

```
accelerometer.enable_freelfall_detection(time=30)
```

enable_motion_detection(*, threshold=18)

The activity detection parameters.

Parameters threshold (*int*) – The value that acceleration on any axis must exceed to register as active. The scale factor is 62.5 mg/LSB.

If you wish to set them yourself rather than using the defaults, you must use keyword arguments:

```
accelerometer.enable_motion_detection(threshold=20)
```

enable_tap_detection(*, tap_count=1, threshold=20, duration=50, latency=20, window=255)

The tap detection parameters.

Parameters

- **tap_count** (*int*) – 1 to detect only single taps, and 2 to detect only double taps.
- **threshold** (*int*) – A threshold for the tap detection. The scale factor is 62.5 mg/LSB. The higher the value the less sensitive the detection.
- **duration** (*int*) – This caps the duration of the impulse above **threshold**. Anything above **duration** won't register as a tap. The scale factor is 625 μ s/LSB
- **latency(double tap only)** (*int*) – The length of time after the initial impulse falls below **threshold** to start the window looking for a second impulse. The scale factor is 1.25 ms/LSB.
- **window(double tap only)** (*int*) – The length of the window in which to look for a second tap. The scale factor is 1.25 ms/LSB

If you wish to set them yourself rather than using the defaults, you must use keyword arguments:

```
accelerometer.enable_tap_detection(duration=30, threshold=25)
```

events

`events` will return a dictionary with a key for each event type that has been enabled. The possible keys are:

Key	Description
tap	True if a tap was detected recently. Whether it's looking for a single or double tap is determined by the tap param of <i>enable_tap_detection</i>
motion	True if the sensor has seen acceleration above the threshold set with <i>enable_motion_detection</i> .
freefall	True if the sensor was in freefall. Parameters are set when enabled with <i>enable_freefall_detection</i>

range

The measurement range of the sensor.

class `adafruit_adxl34x.DataRate`

An enum-like class representing the possible data rates. Possible values are

- `DataRate.RATE_3200_HZ`
- `DataRate.RATE_1600_HZ`
- `DataRate.RATE_800_HZ`
- `DataRate.RATE_400_HZ`
- `DataRate.RATE_200_HZ`
- `DataRate.RATE_100_HZ`
- `DataRate.RATE_50_HZ`
- `DataRate.RATE_25_HZ`
- `DataRate.RATE_12_5_HZ`
- `DataRate.RATE_6_25_HZ`
- `DataRate.RATE_3_13_HZ`
- `DataRate.RATE_1_56_HZ`
- `DataRate.RATE_0_78_HZ`
- `DataRate.RATE_0_39_HZ`
- `DataRate.RATE_0_20_HZ`
- `DataRate.RATE_0_10_HZ`

class `adafruit_adxl34x.Range`

An enum-like class representing the possible measurement ranges in +/- G.

Possible values are

- `Range.RANGE_16_G`
- `Range.RANGE_8_G`
- `Range.RANGE_4_G`
- `Range.RANGE_2_G`

CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`

a

adafruit_adxl34x, [13](#)

A

`acceleration` (*adafruit_adxl34x.ADXL345 attribute*), 13
`adafruit_adxl34x` (*module*), 13
`ADXL343` (*class in adafruit_adxl34x*), 13
`ADXL345` (*class in adafruit_adxl34x*), 13

D

`data_rate` (*adafruit_adxl34x.ADXL345 attribute*), 13
`DataRate` (*class in adafruit_adxl34x*), 15
`disable_freefall_detection()`
 (*adafruit_adxl34x.ADXL345 method*), 14
`disable_motion_detection()`
 (*adafruit_adxl34x.ADXL345 method*), 14
`disable_tap_detection()`
 (*adafruit_adxl34x.ADXL345 method*), 14

E

`enable_freefall_detection()`
 (*adafruit_adxl34x.ADXL345 method*), 14
`enable_motion_detection()`
 (*adafruit_adxl34x.ADXL345 method*), 14
`enable_tap_detection()`
 (*adafruit_adxl34x.ADXL345 method*), 14
`events` (*adafruit_adxl34x.ADXL345 attribute*), 14

R

`range` (*adafruit_adxl34x.ADXL345 attribute*), 15
`Range` (*class in adafruit_adxl34x*), 15