

---

# **AdafruitMPL3115A2 Library Documentation**

***Release 1.0***

**Tony DiCola**

**Feb 22, 2018**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Building locally</b>	<b>9</b>
4.1	Zip release files . . . . .	9
4.2	Sphinx documentation . . . . .	9
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	adafruit_mpl3115a2 . . . . .	12
<b>6</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>



CircuitPython module for the MPL3115A2 barometric pressure & temperature sensor.



# 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](#).





## CHAPTER 2

---

### Usage Example

---

See `examples/simpletest.py` for a demo of the usage.



## CHAPTER 3

---

### Contributing

---

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



### 4.1 Zip release files

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-mpl3115a2 --
↪library_location .
```

### 4.2 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.

## 5.1 Simple test

Ensure your device works with this simple test.

Listing 5.1: examples/simpletest.py

```
1  # Simple demo of the MPL3115A2 sensor.
2  # Will read the pressure and temperature and print them out every second.
3  # Author: Tony DiCola
4  import time
5
6  import board
7  import busio
8
9  import adafruit_mpl3115a2
10
11
12  # Initialize the I2C bus.
13  i2c = busio.I2C(board.SCL, board.SDA)
14
15  # Initialize the MPL3115A2.
16  sensor = adafruit_mpl3115a2.MPL3115A2(i2c)
17  # Alternatively you can specify a different I2C address for the device:
18  #sensor = adafruit_mpl3115a2.MPL3115A2(i2c, address=0x10)
19
20  # You can configure the pressure at sealevel to get better altitude estimates.
21  # This value has to be looked up from your local weather forecast or meteorological
22  # reports. It will change day by day and even hour by hour with weather
23  # changes. Remember altitude estimation from barometric pressure is not exact!
24  # Set this to a value in pascals:
25  sensor.sealevel_pressure = 102250
26
27  # Main loop to read the sensor values and print them every second.
28  while True:
```

```
29 pressure = sensor.pressure
30 print('Pressure: {0:0.3f} pascals'.format(pressure))
31 altitude = sensor.altitude
32 print('Altitude: {0:0.3f} meters'.format(altitude))
33 temperature = sensor.temperature
34 print('Temperature: {0:0.3f} degrees Celsius'.format(temperature))
35 time.sleep(1.0)
```

## 5.2 adafruit\_mpl3115a2

CircuitPython module for the MPL3115A2 barometric pressure & temperature sensor. See `examples/simpletest.py` for a demo of the usage.

- Author(s): Tony DiCola

**class** `adafruit_mpl3115a2.MPL3115A2` (*i2c*, \*, *address*=<*sphinx.ext.autodoc.\_MockObject object*>)

Instance of the MPL3115A2 sensor. Must specify the following parameters when creating an instance of this device: - *i2c*: The I2C bus connected to the sensor.

In addition you can specify the following optional keyword arguments: - *address*: The I2C address of the device if it's different from the default.

### **altitude**

Read the altitude as calculated based on the sensor pressure and previously configured pressure at sea-level. This will return a value in meters. Set the sea-level pressure by updating the `sealevel_pressure` property first to get a more accurate altitude value.

### **pressure**

Read the barometric pressure detected by the sensor in Pascals.

### **sealevel\_pressure**

Read and write the pressure at sea-level used to calculate altitude. You must look this up from a local weather or meteorological report for the best accuracy. This is a value in Pascals.

### **temperature**

Read the temperature as measured by the sensor in degrees Celsius.



## CHAPTER 6

---

### Indices and tables

---

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



### **a**

adafruit\_mpl3115a2, [12](#)



## A

`adafruit_mpl3115a2` (module), [12](#)

`altitude` (`adafruit_mpl3115a2.MPL3115A2` attribute), [12](#)

## M

`MPL3115A2` (class in `adafruit_mpl3115a2`), [12](#)

## P

`pressure` (`adafruit_mpl3115a2.MPL3115A2` attribute), [12](#)

## S

`sealevel_pressure` (`adafruit_mpl3115a2.MPL3115A2` attribute), [12](#)

## T

`temperature` (`adafruit_mpl3115a2.MPL3115A2` attribute), [12](#)