

---

# **Adafruitsht31 Library Documentation**

***Release 1.0***

**Jerry Needell**

**Jan 17, 2020**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Installing from PyPI</b>	<b>5</b>
<b>3</b>	<b>Usage Example</b>	<b>7</b>
<b>4</b>	<b>Contributing</b>	<b>9</b>
<b>5</b>	<b>Documentation</b>	<b>11</b>
<b>6</b>	<b>Table of Contents</b>	<b>13</b>
6.1	Simple test . . . . .	13
6.2	adafruit_sht31d . . . . .	14
6.2.1	Implementation Notes . . . . .	14
<b>7</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>
	<b>Index</b>	<b>21</b>



CircuitPython module for the SHT31-D temperature and humidity 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

---

### 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-sht31d
```

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

```
sudo pip3 install adafruit-circuitpython-sht31d
```

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-sht31d
```



## CHAPTER 3

---

### Usage Example

---

You must import the library to use it:

```
import adafruit_sht31d
```

This driver takes an instantiated and active I2C object (from the `busio` or the `bitbangio` library) as an argument to its constructor. The way to create an I2C object depends on the board you are using. For boards with labeled SCL and SDA pins, you can:

```
from busio import I2C
from board import SCL, SDA

i2c = I2C(SCL, SDA)
```

Once you have created the I2C interface object, you can use it to instantiate the sensor object:

```
sensor = adafruit_sht31d.SHT31D(i2c)
```

And then you can start measuring the temperature and humidity:

```
print(sensor.temperature)
print(sensor.relative_humidity)
```

You can instruct the sensor to periodically measure the temperature and humidity, storing the result in its internal cache:

```
sensor.mode = adafruit_sht31d.MODE_PERIODIC
```

You can adjust the frequency at which the sensor periodically gathers data to: 0.5, 1, 2, 4 or 10 Hz. The following adjusts the frequency to 2 Hz:

```
sensor.frequency = adafruit_sht31d.FREQUENCY_2
```

The sensor is capable of storing eight results. The sensor stores these results in an internal FILO cache. Retrieving these results is similar to taking a measurement. The sensor clears its cache once the stored data is read. The sensor

always returns eight data points. The list of results is backfilled with the maximum output values of 130.0 °C and 100.01831417975366 % RH:

```
print(sensor.temperature)
print(sensor.relative_humidity)
```

The sensor will continue to collect data at the set interval until it is returned to single shot data acquisition mode:

```
sensor.mode = adafruit_sht31d.MODE_SINGLE
```

## CHAPTER 4

---

### Contributing

---

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



## CHAPTER 5

---

### Documentation

---

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





## 6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/sht31d\_simpletest.py

```
1 import time
2 import board
3 import busio
4 import adafruit_sht31d
5
6 # Create library object using our Bus I2C port
7 i2c = busio.I2C(board.SCL, board.SDA)
8 sensor = adafruit_sht31d.SHT31D(i2c)
9
10 loopcount = 0
11 while True:
12     print("\nTemperature: %0.1f C" % sensor.temperature)
13     print("Humidity: %0.1f %" % sensor.relative_humidity)
14     loopcount += 1
15     time.sleep(2)
16     # every 10 passes turn on the heater for 1 second
17     if loopcount == 10:
18         loopcount = 0
19         sensor.heater = True
20         print("Sensor Heater status =", sensor.heater)
21         time.sleep(1)
22         sensor.heater = False
23         print("Sensor Heater status =", sensor.heater)
```

## 6.2 adafruit\_sht31d

This is a CircuitPython driver for the SHT31-D temperature and humidity sensor.

- Author(s): Jerry Needell, Llewelyn Trahaearn

### 6.2.1 Implementation Notes

#### Hardware:

- Adafruit [Sensiron SHT31-D Temperature & Humidity Sensor Breakout](#) (Product ID: 2857)

#### Software and Dependencies:

- Adafruit CircuitPython firmware for the ESP8622 and M0-based boards: <https://github.com/adafruit/circuitpython/releases>
- Adafruit's Bus Device library: [https://github.com/adafruit/Adafruit\\_CircuitPython\\_BusDevice](https://github.com/adafruit/Adafruit_CircuitPython_BusDevice)

**class** `adafruit_sht31d.SHT31D(i2c_bus, address=68)`

A driver for the SHT31-D temperature and humidity sensor.

#### Parameters

- **i2c\_bus** – The `busio.I2C` object to use. This is the only required parameter.
- **address** (*int*) – (optional) The I2C address of the device.

#### **art**

Control accelerated response time This feature only affects ‘Periodic’ mode.

#### **clock\_stretching**

Control clock stretching. This feature only affects ‘Single’ mode.

#### **frequency**

Periodic data acquisition frequency Allowed values are the constants `FREQUENCY_*` Frequency can not be modified when ART is enabled

#### **heater**

Control device’s internal heater.

#### **mode**

Operation mode Allowed values are the constants `MODE_*` Return the device to ‘Single’ mode to stop periodic data acquisition and allow it to sleep.

#### **relative\_humidity**

The measured relative humidity in percent. ‘Single’ mode reads and returns the current humidity as a float. ‘Periodic’ mode returns the most recent readings available from the sensor’s cache in a FILO list of eight floats. This list is backfilled with with the sensor’s maximum output of 100.01831417975366 when the sensor is read before the cache is full.

#### **repeatability**

Repeatability Allowed values are the constants `REP_*`

#### **serial\_number**

Device serial number.

#### **status**

Device status.

#### **temperature**

The measured temperature in degrees celsius. 'Single' mode reads and returns the current temperature as a float. 'Periodic' mode returns the most recent readings available from the sensor's cache in a FILO list of eight floats. This list is backfilled with with the sensor's maximum output of 130.0 when the sensor is read before the cache is full.



## CHAPTER 7

---

### Indices and tables

---

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



### a

adafruit\_sht31d, [13](#)





## A

`adafruit_sht31d` (*module*), 13  
`art` (*adafruit\_sht31d.SHT31D attribute*), 14

## C

`clock_stretching` (*adafruit\_sht31d.SHT31D attribute*), 14

## F

`frequency` (*adafruit\_sht31d.SHT31D attribute*), 14

## H

`heater` (*adafruit\_sht31d.SHT31D attribute*), 14

## M

`mode` (*adafruit\_sht31d.SHT31D attribute*), 14

## R

`relative_humidity` (*adafruit\_sht31d.SHT31D attribute*), 14  
`repeatability` (*adafruit\_sht31d.SHT31D attribute*), 14

## S

`serial_number` (*adafruit\_sht31d.SHT31D attribute*), 14  
`SHT31D` (*class in adafruit\_sht31d*), 14  
`status` (*adafruit\_sht31d.SHT31D attribute*), 14

## T

`temperature` (*adafruit\_sht31d.SHT31D attribute*), 14