Adafruitsht31 Library Documentation

Release 1.0

Jerry Needell

Apr 26, 2021

Contents

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.

```
from board

i2c = board.I2C()
```

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 simlilar 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 ºC 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.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/sht31d_simpletest.py

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

import time
import board
import adafruit_sht31d

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()
sensor = adafruit_sht31d.SHT31D(i2c)

loopcount = 0
while True:
    print("\nTemperature: %0.1f C" % sensor.temperature)
    print("Humidity: %0.1f %%" % sensor.relative_humidity)
    loopcount += 1
    time.sleep(2)
    # every 10 passes turn on the heater for 1 second
    if loopcount == 10:
        loopcount = 0
        sensor.heater = True
        print("Sensor Heater status =", sensor.heater)
        time.sleep(1)
        sensor.heater = False
        print("Sensor Heater status =", sensor.heater)
```

6.2 Simple Mode

Example in how to use the sensor in simple mode

Listing 2: examples/sht31d_simple_mode.py

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

import board
import adafruit_sht31d

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()
sensor = adafruit_sht31d.SHT31D(i2c)

print("\033[1mSensor\033[0m = SHT31-D")
print("\033[1mSerial Number\033[0m = ", sensor.serial_number, "\n")

for i in range(3):
    if i == 0:
        sensor.repeatability = adafruit_sht31d.REP_LOW
        print("\033[1m\033[36mLow Repeatability:\033[0m\n")
    if i == 1:
        sensor.repeatability = adafruit_sht31d.REP_MED
        print("\n\033[1m\033[36mMedium Repeatability:\033[0m\n")
    if i == 2:
        sensor.repeatability = adafruit_sht31d.REP_HIGH
        sensor.clock_stretching = True
        print("\n\033[1m\033[36mHigh Repeatability:\033[0m")
        print("\033[1m\033[95mClock Stretching:\033[0m \033[92mEnabled\033[0m\n")
    for itr in range(3):
        print("\033[1mTemperature:\033[0m %0.3f °C" % sensor.temperature)
        print("\033[1mHumidity:\033[0m %0.2f %%" % sensor.relative_humidity, "\n")
```

6.3 Periodic Mode

Example in how to use the sensor in periodic mode

Listing 3: examples/sht31d_simpletest.py

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

import time
import board
import adafruit_sht31d

# Create sensor object, communicating over the board's default I2C bus
i2c = board.I2C()
sensor = adafruit_sht31d.SHT31D(i2c)

loopcount = 0
while True:
    print("\nTemperature: %0.1f C" % sensor.temperature)
```

(continues on next page)

(continued from previous page)

```
    print("Humidity: %0.1f %%" % sensor.relative_humidity)
    loopcount += 1
    time.sleep(2)
    # every 10 passes turn on the heater for 1 second
    if loopcount == 10:
        loopcount = 0
        sensor.heater = True
        print("Sensor Heater status =", sensor.heater)
        time.sleep(1)
        sensor.heater = False
        print("Sensor Heater status =", sensor.heater)
```

6.4 adafruit_sht31d

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

- Author(s): Jerry Needell, Llewelyn Trahaearn

6.4.1 Implementation Notes

Hardware:

- Adafruit SHT31-D temperature and humidity sensor Breakout: https://www.adafruit.com/product/2857

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://circuitpython.org/downloads
- Adafruit's Bus Device library: 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** (*I2C*) – The I2C bus the SHT31-D is connected to
- **address** (*int*) – (optional) The I2C address of the device. Defaults to `0x44`

Quickstart: Importing and using the SHT31-D

Here is an example of using the *SHT31D* class. First you will need to import the libraries to use the sensor

```
import board
import adafruit_sht31d
```

Once this is done you can define your `board.I2C` object and define your sensor object

```
i2c = board.I2C()   # uses board.SCL and board.SDA
sht = adafruit_sht31d.SHT31D(i2c)
```

Now you have access to the temperature and humidity the the *temperature* and *relative_humidity* attributes

```
temperature = sht.temperature
humidity = sht.relative_humidity
```

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

Python Module Index

a

Index