Adafruitsht31 Library Documentation

Release 1.0

Jerry Needell

Oct 21, 2019

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

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 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 3

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 4

Building locally

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-sht31d --library_
↪location .
```

4.1 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.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/sht31d_simpletest.py

```
import time
import board
import busio
import adafruit_sht31d

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
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)
```

5.2 adafruit_sht31d

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

- Author(s): Jerry Needell, Llewelyn Trahaearn

5.2.1 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

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 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index