Adafruit CircuitPython DHT Library Documentation

Release 1.0

Mike McWethy

Apr 06, 2021

Contents

CircuitPython support for the DHT11 and DHT22 temperature and humidity devices.

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython

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-dht
```

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

```
sudo pip3 install adafruit-circuitpython-dht
```

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-dht
```

CHAPTER 3

Usage Example

3.1 Hardware Set-up

The DHT11 and DHT22 devices both need a pull-resistor on the data signal wire. This resistor is in the range of 1k to 5k. Please check your device datasheet for the appropriate value.

3.2 Basics

Of course, you must import the library to use it:

```
import adafruit_dht
```

The DHT type devices use single data wire, so import the board pin

```
from board import <pin>
```

Now, to initialize the DHT11 device:

```
dht_device = adafruit_dht.DHT11(<pin>)
```

OR initialize the DHT22 device:

```
dht_device = adafruit_dht.DHT22(<pin>)
```

3.3 Read temperature and humidity

Now get the temperature and humidity values

```
temperature = dht_device.temperature
humidity = dht_device.humidity
```

These properties may raise an exception if a problem occurs. You should use try/raise logic and catch RuntimeError and then retry getting the values after at least 2 seconds. If you try again to get a result within 2 seconds, cached values are returned.

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/dht_simpletest.py

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

import time
import board
import adafruit_dht

# Initial the dht device, with data pin connected to:
dhtDevice = adafruit_dht.DHT22(board.D18)

# you can pass DHT22 use_pulseio=False if you wouldn't like to use pulseio.
# This may be necessary on a Linux single board computer like the Raspberry Pi,
# but it will not work in CircuitPython.
# dhtDevice = adafruit_dht.DHT22(board.D18, use_pulseio=False)

while True:
    try:
        # Print the values to the serial port
        temperature_c = dhtDevice.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        print(
            "Temp: {:.1f} F / {:.1f} C    Humidity: {}% ".format(
                temperature_f, temperature_c, humidity
            )
        )

```

(continues on next page)

(continued from previous page)

```
    except RuntimeError as error:
        # Errors happen fairly often, DHT's are hard to read, just keep going
        print(error.args[0])
        time.sleep(2.0)
        continue
    except Exception as error:
        dhtDevice.exit()
        raise error

    time.sleep(2.0)
```

6.2 Time calibration advance test

Check what is the best time your sensor.

Listing 2: examples/dht_time_calibration_advance.py

```
# SPDX-FileCopyrightText: 2021 yeyeto2788 for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
This script let's you check the best timing for you sensor as other people have face␣
↪timing issues
as seen on issue https://github.com/adafruit/Adafruit_CircuitPython_DHT/issues/66.

By changing the variables values below you will be able to check the best timing for␣
↪you sensor,
take into account that by most datasheets the timing for the sensor are 0.001 DHT22␣
↪and
0.018 for DHT11 which are the default values of the library.
"""

import json
import time

import board

import adafruit_dht

# Change the pin used below
pin_to_use = "PG6"

# Maximum number of tries per timing
max_retries_per_time = 10
# Minimum wait time from where to start testing
min_time = 1500
# Maximum wait time on where to stop testing
max_time = 2000
# Increment on time
time_increment = 100

# Variable to store all reads on a try
reads = {}

```

(continues on next page)

(continued from previous page)

```
initial_msg = f"""
\nInitializing test with the following parameters:

- Maximum retries per waiting time: {max_retries_per_time}
- Start time (ms): {min_time}
- End time (ms): {max_time}
- Increment time (ms): {time_increment}

This execution will try to read the sensor {max_retries_per_time} times
for {len(range(min_time, max_time, time_increment))} different wait times values.

"""
# Print initial message on the console.
print(initial_msg)

for milliseconds in range(min_time, max_time, time_increment):
    # Instantiate the DHT11 object.
    dhtDevice = adafruit_dht.DHT11(pin=getattr(board, pin_to_use))
    # Change the default wait time for triggering the read.
    # pylint: disable=protected-access
    dhtDevice._trig_wait = milliseconds

    # pylint: disable=protected-access
    print(f"Using 'trig_wait' of {dhtDevice._trig_wait}")
    # Reset the read count for next loop
    reads_count = 0

    # Create the key on the reads dictionary with the milliseconds used on
    # this try.
    if milliseconds not in reads:
        reads[milliseconds] = {"total_reads": 0}

    for try_number in range(0, max_retries_per_time):
        try:
            # Read temperature and humidity
            temperature = dhtDevice.temperature
            humidity = dhtDevice.humidity
            read_values = {"temperature": temperature, "humidity": humidity}

            if try_number not in reads[milliseconds]:
                reads[milliseconds][try_number] = read_values

            reads_count += 1
        except RuntimeError as e:
            time.sleep(2)
        else:
            time.sleep(1)

    reads[milliseconds]["total_reads"] = reads_count

    print(f"Total read(s): {reads[milliseconds]['total_reads']}\n")
    dhtDevice.exit()

# Gather the highest read numbers from all reads done.
best_result = max([reads[milliseconds]["total_reads"] for milliseconds in reads])

# Gather best time(s) in milliseconds where we got more reads
```

(continues on next page)

(continued from previous page)

```
best_times = [
    milliseconds
    for milliseconds in reads
    if reads[milliseconds]["total_reads"] == best_result
]
print(
    f"Maximum reads: {best_result}  out of {max_retries_per_time} with the "
    f"following times: {', '.join([str(t) for t in best_times])}"
)

# change the value on the line below to see all reads performed.
print_all = False
if print_all:
    print(json.dumps(reads))
```

6.3 adafruit_dhtlib

CircuitPython support for the DHT11 and DHT22 temperature and humidity devices.

- Author(s): Mike McWethy

class adafruit_dht.**DHT11**(*pin*, *use_pulseio=True*)
Support for DHT11 device.

Parameters **pin** (*Pin*) – digital pin used for communication

class adafruit_dht.**DHT22**(*pin*, *use_pulseio=True*)
Support for DHT22 device.

Parameters **pin** (*Pin*) – digital pin used for communication

class adafruit_dht.**DHTBase**(*dht11*, *pin*, *trig_wait*, *use_pulseio*)
base support for DHT11 and DHT22 devices

exit()
Cleans up the PulseIn process. Must be called explicitly

humidity
humidity current reading. It makes sure a reading is available

Raises RuntimeError exception for checksum failure and for insuffcient data returned from the device (try again)

measure()
measure runs the communications to the DHT11/22 type device. if successful, the class properties temperature and humidity will return the reading returned from the device.

Raises RuntimeError exception for checksum failure and for insuffcient data returned from the device (try again)

temperature
temperature current reading. It makes sure a reading is available

Raises RuntimeError exception for checksum failure and for insuffcient data returned from the device (try again)

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index