sparkfun$_{q}wiic_{c}cs811$

Release 0.0.9

Jul 15, 2019

Contents:

1 Contents **3**

2 Supported Platforms **5**

3 Dependencies **7**

4 Documentation **9**

5 Installation **11**
5.1 PyPi Installation 11

6 Raspberry Pi Use **13**

7 Example Use **15**

8 Table of Contents **17**
8.1 API Reference 17
8.1.1 qwiic_ccs811 17
8.2 Basic Operation 21
8.3 Adjust Sensor Settings 22
8.4 Set Enviromental Values 24

9 Indices and tables **27**

Python Module Index **29**

Index **31**

Python module for the qwiic ccs811 sensor, which is part of the SparkFun Qwiic Environmental Combo Breakout

This python package is a port of the existing SparkFun CCS811 Arduino Library

This package can be used in conjunction with the overall SparkFun qwiic Python Package

New to qwiic? Take a look at the entire SparkFun qwiic ecosystem.

warning Using this sensor on a Raspberry Pi? :warning:

Your system might need modification. See this *note*.

CHAPTER 1

Contents

- *Supported Platforms*
- *Dependencies*
- *Installation*
- *Documentation*
- *Example Use*

CHAPTER 2

Supported Platforms

The qwiic CCS811 Python package current supports the following platforms:

- Raspberry Pi
- NVidia Jetson Nano
- Google Coral Development Board

CHAPTER 3

Dependencies

This driver package depends on the qwiic I2C driver: Qwiic_I2C_Py

CHAPTER 4

Documentation

The SparkFun qwiic CCS811 module documentation is hosted at ReadTheDocs

CHAPTER 5

Installation

5.1 PyPi Installation

This repository is hosted on PyPi as the sparkfun-qwiic-ccs811 package. On systems that support PyPi installation via pip, this library is installed using the following commands

For all users (note: the user must have sudo privileges):

```
sudo pip install sparkfun-qwiic-ccs811
```

For the current user:

```
pip install sparkfun-qwiic-ccs811
```

To install, make sure the setuptools package is installed on the system.

Direct installation at the command line:

```
python setup.py install
```

To build a package for use with pip:

```
python setup.py sdist
```

A package file is built and placed in a subdirectory called dist. This package file can be installed using pip.

```
cd dist
pip install sparkfun_qwiic_ccs811-<version>.tar.gz
```

CHAPTER 6

Raspberry Pi Use

For this sensor to work on the Raspberry Pi, I2C clock stretching must be enabled.

To do this:

- Login as root to the target Raspberry Pi
- Open the file /boot/config.txt in your favorite editor (vi, nano . . . etc)
- Scroll down until the block that contains the following is found: .. code-block:: ini

  dtparam=i2c_arm=on dtparam=i2s=on dtparam=spi=on
- Add the following line: .. code-block:: ini

  # Enable I2C clock stretching dtparam=i2c_arm_baudrate=10000
- Save the file
- Reboot the raspberry pi

CHAPTER 7

Example Use

See the examples directory for more detailed use examples.

```
from __future__ import print_function
import qwiic_ccs811
import time
import sys

def runExample():

    print("\nSparkFun CCS811 Sensor Basic Example \n")
    mySensor = qwiic_ccs811.QwiicCcs811()

    if mySensor.isConnected() == False:
        print("The Qwiic CCS811 device isn't connected to the system. Please check␣
↪your connection", \
            file=sys.stderr)
        return

    mySensor.begin()

    while True:

        mySensor.readAlgorithmResults()

        print("CO2:\t%.3f" % mySensor.getCO2())

        print("tVOC:\t%.3f\n" % mySensor.getTVOC())


        time.sleep(1)


if __name__ == '__main__':
    try:
```

(continues on next page)

(continued from previous page)

```
        runExample()
    except (KeyboardInterrupt, SystemExit) as exErr:
        print("\nEnding Basic Example")
        sys.exit(0)
```

CHAPTER 8

Table of Contents

8.1 API Reference

8.1.1 qwiic_ccs811

Python module for the qwiic ccs811 sensor, which is part of the [SparkFun Qwiic Environmental Combo Breakout](https://www.sparkfun.com/products/14348)

This python package is a port of the existing [SparkFun CCS811 Arduino Library](https://github.com/sparkfun/SparkFun_CCS811_Arduino_Library)

This package can be used in conjunction with the overall [SparkFun qwiic Python Package](https://github.com/sparkfun/Qwiic_Py)

New to qwiic? Take a look at the entire [SparkFun qwiic ecosystem](https://www.sparkfun.com/qwiic).

class qwiic_ccs811.**QwiicCcs811**(*address=None*, *i2c_driver=None*)
QwiicCccs811

param address The I2C address to use for the device. If not provided, the default address is used.

param i2c_driver An existing i2c driver object. If not provided a driver object is created.

return The Ccs811 device object.

rtype Object

CO2
Return the current CO2 value.

Returns The CO2 Value

Return type float

TVOC
Return the current TVOC value.

Returns The TVOC Value

Return type float

app_valid()
Returns True if if the sensor APP_VALID bit is set in the status register

Returns True if APP_VALID is set

Return type bool

baseline
Returns the baseline value Used for telling sensor what 'clean' air is You must put the sensor in clean air and record this value

Returns Baseline value for the sensor

Return type integer

begin()
Initialize the operation of the Ccs811 module

Returns Returns SENSOR_SUCCESS on success, SENSOR_ID_ERROR on bad chip ID or SENSOR_INTERNAL_ERROR.

Return type integer

check_status_error()
Returns if the Error bit on the sensor is set.

Returns value of Error bit

Return type integer

connected
Determine if a CCS811 device is conntected to the system..

Returns True if the device is connected, otherwise False.

Return type bool

data_available()
Returns True if data is available on the sensor

Returns True if data is available.

Return type bool

disable_interrupts()
Clear the Interrupt bit in the sensor and disable Interrupts on the sensor

Returns SENSOR_SUCCESS

Return type integer

enable_interrupts()
Set the Interrupt bit in the sensor and enable Interrupts on the sensor

Returns SENSOR_SUCCESS

Return type integer

error_register
Returns the value of the sensors error Register

Returns Error register

Return type int

get_baseline()
Returns the baseline value Used for telling sensor what 'clean' air is You must put the sensor in clean air and record this value

Returns Baseline value for the sensor

Return type integer

get_co2()
Return the current CO2 value.

Returns The CO2 Value

Return type float

get_error_register()
Returns the value of the sensors error Register

Returns Error register

Return type int

get_reference_resistance()
Get the sensors referance resistance

Returns The current reference resistance

Return type integer

get_resistance()
Return the current resistance value.

Returns The resistance value

Return type float

get_temperature()
Return the current temperature value.

Returns The temperature Value

Return type float

get_tvoc()
Return the current TVOC value.

Returns The TVOC Value

Return type float

is_connected()
Determine if a CCS811 device is conntected to the system..

Returns True if the device is connected, otherwise False.

Return type bool

read_algorithm_results()
Reads the resutls from the sensor and stores internally

Returns SENSOR_SUCCESS

Return type integer

read_ntc()
Read the NTC values from the sensor and store for future calications.

NOTE: The qwiic CCS811 doesn't support this function, but other CCS811 sparkfun boards do.

Returns A **SENSOR_** status code

Return type integer

referance_resistance
Get the sensors referance resistance

Returns The current reference resistance

Return type integer

resistance
Return the current resistance value.

Returns The resistance value

Return type float

set_baseline(*input_val*)
Set the baseline value for the sensor

Returns SENSOR_SUCCESS

Return type integer

set_drive_mode(*mode*)
Set the Drive mode for the sensor

Parameters **mode** – Valid values are: 0 = Idle, 1 = read every 1s, 2 = every 10s, 3 = every 60s, 4 = RAW mode

Returns SENSOR_SUCCESS

Return type integer

set_environmental_data(*relativeHumidity*, *temperature*)
Given a temp and humidity, write this data to the CSS811 for better compensation This function expects the humidity and temp to come in as floats

Parameters

- **relativeHumidity** – The relativity Humity for the sensor to use
- **temperature** – The temperature for the sensor to use

Returns one of the **SENSOR_** return codes.

Return type integer

set_reference_resistance(*input_val*)
Set the sensors referance resistance

Parameters **input** – The referance resistance to set in the sensor

Returns No return value

temperature
Return the current temperature value.

Returns The temperature Value

Return type float

8.2 Basic Operation

Listing 1: examples/qwiic_ccs811_ex1.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# qwiic_ccs811_ex1.py
#
# Simple Example for the Qwiic CCS811 Device
#-----------------------------------------------------------------------
#
# Written by  SparkFun Electronics, May 2019
#
#
# More information on qwiic is at https://www.sparkfun.com/qwiic
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 1
#

from __future__ import print_function
import qwiic_ccs811
import time
import sys

def runExample():

        print("\nSparkFun CCS811 Sensor Basic Example \n")
        mySensor = qwiic_ccs811.QwiicCcs811()

        if mySensor.connected == False:
                print("The Qwiic CCS811 device isn't connected to the system. Please␣
↪check your connection", \
                        file=sys.stderr)
                return
```

(continues on next page)

(continued from previous page)

```

        mySensor.begin()

        while True:

                mySensor.read_algorithm_results()

                print("CO2:\t%.3f" % mySensor.CO2)

                print("tVOC:\t%.3f\n" % mySensor.TVOC)


                time.sleep(1)


if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("\nEnding Basic Example")
                sys.exit(0)


```

8.3 Adjust Sensor Settings

Listing 2: examples/qwiic_ccs811_ex3.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# qwiic_ccs811_ex3.py
#
# Simple Example for the Qwiic CCS811 Device
#-----------------------------------------------------------------------------
#
# Written by  SparkFun Electronics, May 2019
#
#
# More information on qwiic is at https://www.sparkfun.com/qwiic
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==============================================================================
# Example 3
#

from __future__ import print_function
import qwiic_ccs811
import time
import sys

# Define some error messages
_deviceErrors = { \
        1 << 5 : "HeaterSupply",  \
    1 << 4 : "HeaterFault", \
    1 << 3 : "MaxResistance", \
    1 << 2 : "MeasModeInvalid",  \
    1 << 1 : "ReadRegInvalid", \
    1 << 0 : "MsgInvalid" \
}

def runExample():

        print("\nSparkFun CCS811 Sensor Example 3 - NTC data to CCS811 for␣
↪compensation. \n")
        mySensor = qwiic_ccs811.QwiicCcs811()

        if mySensor.connected == False:
                print("The Qwiic CCS811 device isn't connected to the system. Please␣
↪check your connection", \
                        file=sys.stderr)
                return

        mySensor.begin()

        mySensor.referance_resistance = 9950

        while True:

                if mySensor.data_available():

                        mySensor.read_algorithm_results()

                        print("CO2:\t%.3f ppm" % mySensor.CO2)

                        print("tVOC:\t%.3f ppb" % mySensor.TVOC)

                        mySensor.read_ntc()
                        print("Measured Resistance: %.3f ohms" % mySensor.resistance)

                        readTemperature = mySensor.temperature
```

(continues on next page)

(continued from previous page)

```
                        print("Converted Temperature: %.2f deg C" % readTemperature)

                        mySensor.set_environmental_data( 50, readTemperature)

                elif mySensor.check_status_error():

                        error = mySensor.get_error_register();
                        if error == 0xFF:
                                # communication error
                                print("Failed to get Error ID register from sensor")
                        else:
                                strErr = "Unknown Error"
                                for code in _deviceErrors.keys():
                                        if error & code:
                                                strErr = _deviceErrors[code]
                                                break
                                print("Device Error: %s" % strErr)

                time.sleep(1)


if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("\nEnding Example")
                sys.exit(0)


```

8.4 Set Enviromental Values

Listing 3: examples/qwiic_ccs811_ex7.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# qwiic_ccs811_ex7.py
#
# Simple Example for the Qwiic CCS811 Device
#-----------------------------------------------------------------------------
#
# Written by  SparkFun Electronics, May 2019
#
#
# More information on qwiic is at https://www.sparkfun.com/qwiic
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
```

(continues on next page)

(continued from previous page)

```
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 3
#

from __future__ import print_function
import qwiic_ccs811
import time
import sys
import random

# Define some error messages
_deviceErrors = { \
        1 << 5 : "HeaterSupply",  \
    1 << 4 : "HeaterFault", \
    1 << 3 : "MaxResistance", \
    1 << 2 : "MeasModeInvalid",  \
    1 << 1 : "ReadRegInvalid", \
    1 << 0 : "MsgInvalid" \
}

def runExample():

        print("\nSparkFun CCS811 Sensor Example 3 - NTC data to CCS811 for␣
↪compensation. \n")
        mySensor = qwiic_ccs811.QwiicCcs811()

        if mySensor.connected == False:
                print("The Qwiic CCS811 device isn't connected to the system. Please␣
↪check your connection", \
                        file=sys.stderr)
                return

        mySensor.begin()


        while True:

                humidityVariable = random.randrange(0, 10000)/100   # 0 to 100%
                temperatureVariable = random.randrange(500, 7000) / 100 #5C to 70C

                print("New humidity and temperature:")
                print("  Humidity:    %.2f percent relative" % humidityVariable)
                print("  Temperature: %.2f degrees C" % temperatureVariable)
```

(continues on next page)

(continued from previous page)

```
76
77                  mySensor.set_environmental_data(humidityVariable, temperatureVariable)
78                  if mySensor.data_available():
79
80                          mySensor.read_algorithm_results()
81
82                          print("  CO2:\t%.3f ppm" % mySensor.CO2)
83                          print("  tVOC:\t%.3f ppb\n" % mySensor.TVOC)
84
85                  elif mySensor.check_status_error():
86
87                          error = mySensor.get_error_register();
88                          if error == 0xFF:
89                                  # communication error
90                                  print("Failed to get Error ID register from sensor")
91                          else:
92                                  strErr = "Unknown Error"
93                                  for code in _deviceErrors.keys():
94                                          if error & code:
95                                                  strErr = _deviceErrors[code]
96                                                  break
97                                  print("Device Error: %s" % strErr)
98
99                  time.sleep(1)
100
101
102 if __name__ == '__main__':
103         try:
104                 runExample()
105         except (KeyboardInterrupt, SystemExit) as exErr:
106                 print("\nEnding Example")
107                 sys.exit(0)
108
109
```

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

q

qwiic_ccs811, 17

Index

A

app_valid() (*qwiic_ccs811.QwiicCcs811 method*), 18

B

baseline (*qwiic_ccs811.QwiicCcs811 attribute*), 18
begin() (*qwiic_ccs811.QwiicCcs811 method*), 18

C

check_status_error() (*qwiic_ccs811.QwiicCcs811 method*), 18
CO2 (*qwiic_ccs811.QwiicCcs811 attribute*), 17
connected (*qwiic_ccs811.QwiicCcs811 attribute*), 18

D

data_available() (*qwiic_ccs811.QwiicCcs811 method*), 18
disable_interrupts() (*qwiic_ccs811.QwiicCcs811 method*), 18

E

enable_interrupts() (*qwiic_ccs811.QwiicCcs811 method*), 18
error_register (*qwiic_ccs811.QwiicCcs811 attribute*), 18

G

get_baseline() (*qwiic_ccs811.QwiicCcs811 method*), 18
get_co2() (*qwiic_ccs811.QwiicCcs811 method*), 19
get_error_register() (*qwiic_ccs811.QwiicCcs811 method*), 19
get_reference_resistance() (*qwiic_ccs811.QwiicCcs811 method*), 19
get_resistance() (*qwiic_ccs811.QwiicCcs811 method*), 19
get_temperature() (*qwiic_ccs811.QwiicCcs811 method*), 19
get_tvoc() (*qwiic_ccs811.QwiicCcs811 method*), 19

I

is_connected() (*qwiic_ccs811.QwiicCcs811 method*), 19

Q

qwiic_ccs811 (*module*), 17
QwiicCcs811 (*class in qwiic_ccs811*), 17

R

read_algorithm_results() (*qwiic_ccs811.QwiicCcs811 method*), 19
read_ntc() (*qwiic_ccs811.QwiicCcs811 method*), 19
referance_resistance (*qwiic_ccs811.QwiicCcs811 attribute*), 20
resistance (*qwiic_ccs811.QwiicCcs811 attribute*), 20

S

set_baseline() (*qwiic_ccs811.QwiicCcs811 method*), 20
set_drive_mode() (*qwiic_ccs811.QwiicCcs811 method*), 20
set_environmental_data() (*qwiic_ccs811.QwiicCcs811 method*), 20
set_reference_resistance() (*qwiic_ccs811.QwiicCcs811 method*), 20

T

temperature (*qwiic_ccs811.QwiicCcs811 attribute*), 20
TVOC (*qwiic_ccs811.QwiicCcs811 attribute*), 17