py-opc Documentation

David H Hagan

Oct 11, 2018

Contents

py-opc is a python module that makes it easy to operate the Alphasense OPC-N2 optical particle counter using a Raspberry Pi over the SPI bus using either a SPI-USB converter or directly using the GPIO pins. It was originally designed using a Rapsberry Pi 2 Model B and Python3.5; however, it should work on all variants.

There are a variety of OPC Models and firmware versions from Alphasense; a table documenting which ones are supported can be found in the complete documentation. If you own an OPC-N2 with a firmware version that has not been tested, please do so and submit as an issue on the GitHub repository.

CHAPTER 1

Installation

There are several ways to install py-opc. The recommended method is through the python package manager, `pip`:

```
>>> pip install py-opc [--upgrade]
```

If interested in testing a development version, clone (or download) the repository, navigate to the folder where the files are located, and install as follows:

```
>>> python setup.py develop
```

If interested in developing, please visit the "how to contribute" page.

CHAPTER 2

Requirements

One of the following packages is required:

- py-spidev
- pyusbiss (0.2.0 or greater)

Use the spidev library if you are planning to connect to OPC to the microcontroller via the GPIO pins. Use the pyusbiss library if connecting via a SPI-USB adapter.

CHAPTER 3

Setting Up the Raspberry Pi

There are now two simple ways to connect your Alphasense OPC to a Raspberry Pi (or similar device): directly via the GPIO pins, or using a SPI-USB converter.

3.1 Connecting via GPIO

If you are not familiar with setting up a Raspberry Pi to be used as a SPI device, here are a couple of great tutorials: RPi, Drogon, and Hensley. A few important things to note:

- The Alphsense OPC-N2 is a 3v3 logic SPI Mode 1 device
- The OPC requires at least 250 mA, so powering directly through the RPi is not an option

To connect the RPi to the OPC-N2, there are four connections that must be made, plus ground and power. The power source must be 5 VDC, and should power both the OPC and the RPi to avoid ground issues. The connections are stated below:

Pin	Function	OPC	RPi
1	5 VDC	VCC	•
2	Serial Clock	SCK	CLK
3	Master In Slave Out	SDO	MISO
4	Master Out Slave In	SDI	MOSI
5	Chip Select	/SS	CE0 or CE1
6	Ground	GND	•

3.2 Connecting via a USB-SPI Converter

To connect your OPC to the Raspberry Pi directly, you can use a SPI-USB converter. They can be found fairly inexpensively online or directly from Alphasense.

You will then be able to directly connect the OPC to your Raspberry Pi's USB port.

CHAPTER 4

Getting Help

Still running into problems?

- To report a problem with this documentation, contact the author.
- Report an issue with the py-opc library on GitHub
- Submit a feature request for py-opc on GitHub.

CHAPTER 5

Current Supported Firmware

There are several versions of Alphasense OPC-N2 firmware's that are currently deployed. If you have a version that is not listed in the table or is not tested, please send a pull request with your test results! The following versions have been tested:

OPC-N2 Firmware Version	Python2.7	Python3.5	Date Tested	Tested By
v14				
v15				
v16				
v17				
v18.2	Yes	Yes	2016-04-02	D. Hagan

CHAPTER 6

Examples

6.1 Setting up the SPI Connection

6.1.1 Using the SpiDev Library via GPIO Pins

```
import spidev
import opc

# Open a SPI connection on CE0
spi = spidev.SpiDev()
spi.open(0, 0)

# Set the SPI mode and clock speed
spi.mode = 1
spi.max_speed_hz = 500000
```

6.1.2 Using the pyusbiss Library via USB Port

```
from usbiss.spi import SPI
import opc

# Open a SPI connection
spi = SPI("/dev/ttyACM0")

# Set the SPI mode and clock speed
spi.mode = 1
spi.max_speed_hz = 500000
```

6.2 Initiating the OPCN2

```
try:
    alpha = opc.OPCN2(spi)
except Exception as e:
    print ("Startup Error: {}".format(e))
```

6.3 Reading a Histogram

```
# Turn on the OPC
alpha.on()

# Read the histogram and print to console
for key, value in alpha.histogram().items():
    print ("Key: {}\tValue: {}".format(key, value))

# Shut down the opc
alpha.off()
```

CHAPTER 7

API Reference

class opc.**_OPC**(*spi_connection*, *firmware=None*, *max_cnxn_retries=5*, *retry_interval_ms=1000*, ***kwargs*)

Generic class for any Alphasense OPC. Provides the common methods and calculations for each OPC. This class is designed to be the base class, and should not be used alone unless during development.

Parameters

- **spi_connection** (*spidev.SpiDev or usbiss.spi.SPI*) – spidev.SpiDev or usbiss.spi.SPI connection
- **debug** (*boolean*) – Set true to print data to console while running
- **model** (*string*) – Model number of the OPC ('N1' or 'N2') set by the parent class
- **firmware** – You can manually set the firmware version as a tuple. Ex. (18,2)
- **max_cnxn_retries** (*int*) – Maximum number of times a connection will try to be made.
- **retry_interval_ms** (*int*) – The sleep interval for the device between retrying to connect to the OPC. Units are in ms.

Raises opc.exceptions.SpiConnectionError

Return type *opc._OPC*

_16bit_unsigned(*LSB*, *MSB*)

Returns the combined LSB and MSB

Parameters

- **LSB** (*byte*) – Least Significant Byte
- **MSB** (*byte*) – Most Significant Byte

Return type 16-bit unsigned int

_calculate_float(*byte_array*)

Returns an IEEE 754 float from an array of 4 bytes

Parameters **byte_array** (*array*) – Expects an array of 4 bytes

Return type float

_calculate_mtof(*mtof*)
Returns the average amount of time that particles in a bin took to cross the path of the laser [units -> microseconds]

Parameters **mtof** (*float*) – mass time-of-flight

Return type float

_calculate_period(*vals*)
calculate the sampling period in seconds

_calculate_pressure(*vals*)
Calculates the pressure in pascals

Parameters **vals** (*array*) – array of bytes

Return type float

_calculate_temp(*vals*)
Calculates the temperature in degrees celcius

Parameters **vals** (*array*) – array of bytes

Return type float

calculate_bin_boundary(*bb*)
Calculate the adc value that corresponds to a specific bin boundary diameter in microns.

Parameters **bb** (*float*) – Bin Boundary in microns

Return type int

lookup_bin_boundary(*adc_value*)
Looks up the bin boundary value in microns based on the lookup table provided by Alphasense.

Parameters **adc_value** (*int*) – ADC Value (0 - 4095)

Return type float

ping()
Checks the connection between the Raspberry Pi and the OPC

Return type Boolean

ping()
Checks the connection between the Raspberry Pi and the OPC

Return type Boolean

read_info_string()
Reads the information string for the OPC

Return type string

Example

```
>>> alpha.read_info_string()
'OPC-N2 FirmwareVer=OPC-018.2....................BD'
```

class opc.**OPCN1**(*spi_connection*, *firmware=None*, *max_cnxn_retries=5*, *retry_interval_ms=1000*, ***kwargs*)
Create an instance of the Alphasene OPC-N1. opc.OPCN1 inherits from the opc.OPC parent class.

Parameters **spi_connection** (*spidev.SpiDev*) – The spidev instance for the SPI connection.

Return type *opc.OPCN1*

Raises FirmwareVersionError

off()
Turn OFF the OPC (fan and laser)

Returns boolean success state

on()
Turn ON the OPC (fan and laser)

Returns boolean success state

read_bin_boundaries()
Return the bin boundaries.

Returns dictionary with 17 bin boundaries.

read_bin_particle_density()
Read the bin particle density

Returns float

read_gsc_sfr()
Read the gain-scaling-coefficient and sample flow rate.

Returns dictionary containing GSC and SFR

read_histogram()
Read and reset the histogram. The expected return is a dictionary containing the counts per bin, MToF for bins 1, 3, 5, and 7, temperature, pressure, the sampling period, the checksum, PM1, PM2.5, and PM10.

NOTE: The sampling period for the OPCN1 seems to be incorrect.

Returns dictionary

write_bin_particle_density()
Write the bin particle density values to memory. This method is currently a placeholder.

Returns None

write_gsc_sfr()
Write the gsc and sfr values

NOTE: This method is currently a placeholder.

class opc.**OPCN2**(*spi_connection*, ***kwargs*)
Create an instance of the Alphasene OPC-N2. Currently supported by firmware versions 14-18. opc.OPCN2 inherits from the opc.OPC parent class.

Parameters **spi_connection** (*spidev.SpiDev*) – The spidev instance for the SPI connection.

Return type *opc.OPCN2*

Raises opc.exceptions.FirmwareVersionError

Example

```
>>> alpha = opc.OPCN2(spi)
>>> alpha
Alphasense OPC-N2v18.2
```

_enter_bootloader_mode()
Enter bootloader mode. Must be issued prior to writing configuration variables to non-volatile memory.

Return type boolean

Example

```
>>> alpha._enter_bootloader_mode()
True
```

config()
Read the configuration variables and returns them as a dictionary

Return type dictionary

Example

```
>>> alpha.config()
{
    'BPD 13': 1.6499,
    'BPD 12': 1.6499,
    'BPD 11': 1.6499,
    'BPD 10': 1.6499,
    'BPD 15': 1.6499,
    'BPD 14': 1.6499,
    'BSVW 15': 1.0,
    ...
}
```

config2(***kwargs*)
Read the second set of configuration variables and return as a dictionary.

NOTE: This method is supported by firmware v18+.

Return type dictionary

Example

```
>>> a.config2()
{
    'AMFanOnIdle': 0,
    'AMIdleIntervalCount': 0,
    'AMMaxDataArraysInFile': 61798,
    'AMSamplingInterval': 1,
    'AMOnlySavePMData': 0,
    'AMLaserOnIdle': 0
}
```

histogram(*number_concentration=True*)
Read and reset the histogram. As of v1.3.0, histogram values are reported in particle number concentration (#/cc) by default.

Parameters **number_concentration** (*boolean*) – If true, histogram bins are reported in number concentration vs. raw values.

Return type dictionary

Example

```
>>> alpha.histogram()
{
```

(continues on next page)

(continued from previous page)

```
    'Temperature': None,
    'Pressure': None,
    'Bin 0': 0,
    'Bin 1': 0,
    'Bin 2': 0,
    ...
    'Bin 15': 0,
    'SFR': 3.700,
    'Bin1MToF': 0,
    'Bin3MToF': 0,
    'Bin5MToF': 0,
    'Bin7MToF': 0,
    'PM1': 0.0,
    'PM2.5': 0.0,
    'PM10': 0.0,
    'Sampling Period': 2.345,
    'Checksum': 0
}
```

off()

Turn OFF the OPC (fan and laser)

Return type boolean

Example

```
>>> alpha.off()
True
```

on()

Turn ON the OPC (fan and laser)

Return type boolean

Example

```
>>> alpha.on()
True
```

pm(***kwargs*)

Read the PM data and reset the histogram

NOTE: This method is supported by firmware v18+.

Return type dictionary

Example

```
>>> alpha.pm()
{
    'PM1': 0.12,
    'PM2.5': 0.24,
    'PM10': 1.42
}
```

read_firmware(***kwargs*)

Read the firmware version of the OPC-N2. Firmware v18+ only.

Return type dict

Example

```
>>> alpha.read_firmware()
{
    'major': 18,
    'minor': 2,
    'version': 18.2
}
```

read_pot_status(***kwargs*)

Read the status of the digital pot. Firmware v18+ only. The return value is a dictionary containing the following as unsigned 8-bit integers: FanON, LaserON, FanDACVal, LaserDACVal.

Return type dict

Example

```
>>> alpha.read_pot_status()
{
    'LaserDACVal': 230,
    'FanDACVal': 255,
    'FanON': 0,
    'LaserON': 0
}
```

save_config_variables()

Save the configuration variables in non-volatile memory. This method should be used in conjuction with *write_config_variables*.

Return type boolean

Example

```
>>> alpha.save_config_variables()
True
```

set_fan_power(*power*)

Set only the Fan power.

Parameters **power** (*int*) – Fan power value as an integer between 0-255.

Return type boolean

Example

```
>>> alpha.set_fan_power(255)
True
```

set_laser_power(*power*)

Set the laser power only.

Parameters **power** (*int*) – Laser power as a value between 0-255.

Return type boolean

Example

```
>>> alpha.set_laser_power(230)
True
```

sn(***kwargs*)

Read the Serial Number string. This method is only available on OPC-N2 firmware versions 18+.

Return type string

Example

```
>>> alpha.sn()
'OPC-N2 123456789'
```

toggle_fan(*state*)
Toggle the power state of the fan.

Parameters **state** (*boolean*) – Boolean state of the fan

Return type boolean

Example

```
>>> alpha.toggle_fan(False)
True
```

toggle_laser(*state*)
Toggle the power state of the laser.

Parameters **state** (*boolean*) – Boolean state of the laser

Return type boolean

Example

```
>>> alpha.toggle_laser(True)
True
```

write_config_variables(*config_vars*)
Write configuration variables to non-volatile memory.

NOTE: This method is currently a placeholder and is not implemented.

Parameters **config_vars** (*dictionary*) – dictionary containing the configuration variables

write_config_variables2(***kwargs*)
Write configuration variables 2 to non-volatile memory.

NOTE: This method is currently a placeholder and is not implemented. NOTE: This method is supported by firmware v18+.

Parameters **config_vars** (*dictionary*) – dictionary containing the configuration variables

write_sn(***kwargs*)
Write the Serial Number string. This method is available for Firmware versions 18+.

NOTE: This method is currently a placeholder and is not implemented.

Parameters **sn** (*string*) – string containing the serial number to write

7.1 Exceptions

exception opc.exceptions.**FirmwareVersionError**
Raised if the firmware version of your OPC is not supported with this version of the py-opc module. Please check the GitHub repository for updates.

This is usually raised under two circumstances:

1. Your firmware version is not supported
2. Your firmware version cannot be detected (usually due to a bad wiring)

exception opc.exceptions.**SpiConnectionError**
Raised when the argument sent to opc.OPCN2() is not a valid spidev.SpiDev instance.

Python Module Index

Index

Symbols

C

F

H

L

O

P

R

S

T

W