
AdafruitAPDS9960 Library Documentation

Release 1.0

Michael McWethy

Mar 02, 2021

Contents

1	Installation and Dependencies	3
1.1	Installing from PyPI	3
2	Usage Example	5
2.1	Hardware Set-up	5
2.2	Basics	5
2.3	Gestures	6
2.4	Color Measurement	6
2.5	Proximity Detection	6
3	Contributing	7
4	Building locally	9
4.1	Sphinx documentation	9
5	Table of Contents	11
5.1	Simple test	11
5.2	APDS9960	13
5.3	colorutility	14
6	Indices and tables	15
	Python Module Index	17
	Index	19

The APDS9960 is a specialized chip that detects hand gestures, proximity and ambient light color over I2C. Its available on [Adafruit as a breakout](#).

Installation and 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](#).

1.1 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-apds9960
```

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

```
sudo pip3 install adafruit-circuitpython-apds9960
```

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-apds9960
```


CHAPTER 2

Usage Example

```
import board
import busio
import digitalio
from adafruit_apds9960.apds9960 import APDS9960

i2c = busio.I2C(board.SCL, board.SDA)
int_pin = digitalio.DigitalInOut(board.D5)
apds = APDS9960(i2c, interrupt_pin=int_pin)

apds.enable_proximity = True
apds.proximity_interrupt_threshold = (0, 175)
apds.enable_proximity_interrupt = True

while True:
    print(apds.proximity)
    apds.clear_interrupt()
```

2.1 Hardware Set-up

Connect Vin to 3.3 V or 5 V power source, GND to ground, SCL and SDA to the appropriate pins.

2.2 Basics

Of course, you must import i2c bus device, board pins, and the library:

```
from board import SCL, SDA, A1
from adafruit_apds9960.apds9960 import APDS9960
import busio
import digitalio
```

To set-up the device to gather data, initialize the I2CDevice using SCL and SDA pins. Then initialize the library. Optionally provide an interrupt pin for proximity detection.

```
int_pin = digitalio.DigitalInOut(A1)
i2c = busio.I2C(SCL, SDA)
apds = APDS9960(i2c, interrupt_pin=int_pin)
```

2.3 Gestures

To get a gesture, see if a gesture is available first, then get the gesture Code

```
gesture = apds.gesture()
if gesture == 1:
    print("up")
if gesture == 2:
    print("down")
if gesture == 3:
    print("left")
if gesture == 4:
    print("right")
```

2.4 Color Measurement

To get a color measure, enable color measures, wait for color data, then get the color data.

```
apds.enable_color = True

while not apds.color_data_ready:
    time.sleep(0.005)

r, g, b, c = apds.color_data
print("r: {}, g: {}, b: {}, c: {}".format(r, g, b, c))
```

2.5 Proximity Detection

To check for a object in proximity, see if a gesture is available first, then get the gesture Code

```
apds.enable_proximity = True

# set the interrupt threshold to fire when proximity reading goes above 175
apds.proximity_interrupt_threshold = (0, 175)

# enable the proximity interrupt
apds.enable_proximity_interrupt = True

while True:
    if not interrupt_pin.value:
        print(apds.proximity)

        # clear the interrupt
        apds.clear_interrupt()
```

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-travis-build-tools` package.

Once installed, make sure you are in the virtual environment:

Then run the build:

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.

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/apds9960_color_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5  import board
6  import busio
7  from adafruit_apds9960.apds9960 import APDS9960
8  from adafruit_apds9960 import colorutility
9
10 i2c = busio.I2C(board.SCL, board.SDA)
11 apds = APDS9960(i2c)
12 apds.enable_color = True
13
14
15 while True:
16     # create some variables to store the color data in
17
18     # wait for color data to be ready
19     while not apds.color_data_ready:
20         time.sleep(0.005)
21
22     # get the data and print the different channels
23     r, g, b, c = apds.color_data
24     print("red: ", r)
25     print("green: ", g)
26     print("blue: ", b)
27     print("clear: ", c)
```

(continues on next page)

(continued from previous page)

```

28
29     print("color temp {}".format(colorutility.calculate_color_temperature(r, g, b)))
30     print("light lux {}".format(colorutility.calculate_lux(r, g, b)))
31     time.sleep(0.5)

```

Listing 2: examples/apds9960_gesture_simpletest.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  from board import SCL, SDA
5  import busio
6  from adafruit_apds9960.apds9960 import APDS9960
7
8  i2c = busio.I2C(SCL, SDA)
9
10 apds = APDS9960(i2c)
11 apds.enable_proximity = True
12 apds.enable_gesture = True
13
14 # Uncomment and set the rotation if depending on how your sensor is mounted.
15 # apds.rotation = 270 # 270 for CLUE
16
17 while True:
18     gesture = apds.gesture()
19
20     if gesture == 0x01:
21         print("up")
22     elif gesture == 0x02:
23         print("down")
24     elif gesture == 0x03:
25         print("left")
26     elif gesture == 0x04:
27         print("right")

```

Listing 3: examples/apds9960_proximity_simpletest.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import board
5  import busio
6  import digitalio
7  from adafruit_apds9960.apds9960 import APDS9960
8
9  i2c = busio.I2C(board.SCL, board.SDA)
10 int_pin = digitalio.DigitalInOut(board.D5)
11 apds = APDS9960(i2c, interrupt_pin=int_pin)
12
13 apds.enable_proximity = True
14 apds.proximity_interrupt_threshold = (0, 175)
15 apds.enable_proximity_interrupt = True
16
17 while True:
18     # print the proximity reading when the interrupt pin goes low
19     if not int_pin.value:

```

(continues on next page)

(continued from previous page)

```

20     print(apds.proximity)
21
22     # clear the interrupt
23     apds.clear_interrupt()

```

5.2 APDS9960

Driver class for the APDS9960 board. Supports gesture, proximity, and color detection.

- Author(s): Michael McWethy

```

class adafruit_apds9960.apds9960.APDS9960 (i2c, *, interrupt_pin=None, address=57, inte-
                                     gration_time=1, gain=1, rotation=0)
    APDS9900 provide basic driver services for the ASDS9960 breakout board

    clear_interrupt ()
        Clear all interrupts

    color_data
        Tuple containing r, g, b, c values

    color_data_ready
        Color data ready flag. zero if not ready, 1 is ready

    color_gain
        Color gain value

    enable
        Board enable. True to enable, False to disable

    enable_color
        Color detection enable flag. True when color detection is enabled, else False

    enable_gesture
        Gesture detection enable flag. True to enable, False to disable. Note that when disabled, gesture mode is
        turned off

    enable_proximity
        Enable of proximity mode

    enable_proximity_interrupt
        Proximity interrupt enable flag. True if enabled, False to disable

    gesture ()
        Returns gesture code if detected. =0 if no gesture detected =1 if an UP, =2 if a DOWN, =3 if an LEFT, =4
        if a RIGHT

    gesture_dimensions
        Gesture dimension value: range 0-3

    gesture_fifo_threshold
        Gesture fifo threshold value: range 0-3

    gesture_gain
        Gesture gain value: range 0-3

    gesture_proximity_threshold
        Proximity threshold value: range 0-255

```

integration_time

Proximity integration time: range 0-255

proximity

Proximity value: range 0-255

proximity_interrupt_threshold

Tuple containing low and high threshold followed by the proximity interrupt persistence. When setting the proximity interrupt threshold values using a tuple of zero to three values: low threshold, high threshold, persistence. persistence defaults to 4 if not provided

rotated_gesture (*original_gesture*)

Applies rotation offset to the given gesture direction and returns the result

rotation

Gesture rotation offset. Acceptable values are 0, 90, 180, 270.

5.3 colorutility

Helper functions for color calculations

- Author(s): Michael McWethy

`adafruit_apds9960.colorutility.calculate_color_temperature(r, g, b)`

Converts the raw R/G/B values to color temperature in degrees Kelvin

`adafruit_apds9960.colorutility.calculate_lux(r, g, b)`

Calculate ambient light values

CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`

a

`adafruit_apds9960.apds9960`, [13](#)

`adafruit_apds9960.colorutility`, [14](#)

A

adafruit_apds9960.apds9960 (module), 13
adafruit_apds9960.colorutility (module), 14
APDS9960 (class in adafruit_apds9960.apds9960), 13

C

calculate_color_temperature() (in module adafruit_apds9960.colorutility), 14
calculate_lux() (in module adafruit_apds9960.colorutility), 14
clear_interrupt() (adafruit_apds9960.apds9960.APDS9960 method), 13
color_data (adafruit_apds9960.apds9960.APDS9960 attribute), 13
color_data_ready (adafruit_apds9960.apds9960.APDS9960 attribute), 13
color_gain (adafruit_apds9960.apds9960.APDS9960 attribute), 13

E

enable (adafruit_apds9960.apds9960.APDS9960 attribute), 13
enable_color (adafruit_apds9960.apds9960.APDS9960 attribute), 13
enable_gesture (adafruit_apds9960.apds9960.APDS9960 attribute), 13
enable_proximity (adafruit_apds9960.apds9960.APDS9960 attribute), 13
enable_proximity_interrupt (adafruit_apds9960.apds9960.APDS9960 attribute), 13

G

gesture() (adafruit_apds9960.apds9960.APDS9960 method), 13
gesture_dimensions (adafruit_apds9960.apds9960.APDS9960 attribute), 13

gesture_fifo_threshold (adafruit_apds9960.apds9960.APDS9960 attribute), 13
gesture_gain (adafruit_apds9960.apds9960.APDS9960 attribute), 13
gesture_proximity_threshold (adafruit_apds9960.apds9960.APDS9960 attribute), 13

I

integration_time (adafruit_apds9960.apds9960.APDS9960 attribute), 13

P

proximity (adafruit_apds9960.apds9960.APDS9960 attribute), 14
proximity_interrupt_threshold (adafruit_apds9960.apds9960.APDS9960 attribute), 14

R

rotated_gesture() (adafruit_apds9960.apds9960.APDS9960 method), 14
rotation (adafruit_apds9960.apds9960.APDS9960 attribute), 14