AdafruitAS7341 Library Documentation

Release 1.0

Bryan Siepert

May 24, 2021

Contents

CircuitPython library for use with the Adafruit AS7341 breakout. **NOTE**: Due to the size of this library, it may not work on M0 (ex: Trinket M0) and other low memory boards.

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython
- Bus Device
- Register

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-as7341
```

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

```
sudo pip3 install adafruit-circuitpython-as7341
```

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-as7341
```

CHAPTER 3

Usage Example

```
from time import sleep
import board
from adafruit_as7341 import AS7341

i2c = board.I2C()  # uses board.SCL and board.SDA
sensor = AS7341(i2c)


def bar_graph(read_value):
    scaled = int(read_value / 1000)
    return "[%5d] " % read_value + (scaled * "*")


while True:

    print("F1 - 415nm/Violet  %s" % bar_graph(sensor.channel_415nm))
    print("F2 - 445nm//Indigo %s" % bar_graph(sensor.channel_445nm))
    print("F3 - 480nm//Blue   %s" % bar_graph(sensor.channel_480nm))
    print("F4 - 515nm//Cyan   %s" % bar_graph(sensor.channel_515nm))
    print("F5 - 555nm/Green   %s" % bar_graph(sensor.channel_555nm))
    print("F6 - 590nm/Yellow  %s" % bar_graph(sensor.channel_590nm))
    print("F7 - 630nm/Orange  %s" % bar_graph(sensor.channel_630nm))
    print("F8 - 680nm/Red     %s" % bar_graph(sensor.channel_680nm))
    print("\n----------------------------------------------")
    sleep(1)
```

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/as7341_simpletest.py

```
# SPDX-FileCopyrightText: 2020 Bryan Siepert, written for Adafruit Industries
# SPDX-License-Identifier: MIT
from time import sleep
import board
from adafruit_as7341 import AS7341

i2c = board.I2C()  # uses board.SCL and board.SDA
sensor = AS7341(i2c)


def bar_graph(read_value):
    scaled = int(read_value / 1000)
    return "[%5d] " % read_value + (scaled * "*")


while True:

    print("F1 - 415nm/Violet  %s" % bar_graph(sensor.channel_415nm))
    print("F2 - 445nm//Indigo %s" % bar_graph(sensor.channel_445nm))
    print("F3 - 480nm//Blue   %s" % bar_graph(sensor.channel_480nm))
    print("F4 - 515nm//Cyan   %s" % bar_graph(sensor.channel_515nm))
    print("F5 - 555nm/Green   %s" % bar_graph(sensor.channel_555nm))
    print("F6 - 590nm/Yellow  %s" % bar_graph(sensor.channel_590nm))
    print("F7 - 630nm/Orange  %s" % bar_graph(sensor.channel_630nm))
    print("F8 - 680nm/Red     %s" % bar_graph(sensor.channel_680nm))
    print("\n-------------------------------------------------")
    sleep(1)
```

6.2 LED test

Testing the LED

Listing 2: examples/as7341_led_test.py

```
# SPDX-FileCopyrightText: 2020 Bryan Siepert, written for Adafruit Industries
#
# SPDX-License-Identifier: MIT
from time import sleep
import board
import adafruit_as7341

i2c = board.I2C()  # uses board.SCL and board.SDA
sensor = adafruit_as7341.AS7341(i2c)

print("out of init!")
print("Current current is")
print(sensor.led_current)
print("Setting current")
sensor.led_current = 50
print("enabling led")
sensor.led = True
sleep(0.5)
print("disabling LED")
sensor.led = False

print("led status:", sensor.led)
```

6.3 Flicker Detection

Showing how to use flicker detection

Listing 3: examples/as7341_flicker_detection.py

```
# SPDX-FileCopyrightText: 2020 Bryan Siepert, written for Adafruit Industries
# SPDX-License-Identifier: MIT
from time import sleep
import board
from adafruit_as7341 import AS7341

i2c = board.I2C()  # uses board.SCL and board.SDA
sensor = AS7341(i2c)
sensor.flicker_detection_enabled = True

while True:

    flicker_detected = sensor.flicker_detected
    if flicker_detected:
        print("Detected a %d Hz flicker" % flicker_detected)

    sleep(0.1)
```

6.4 Batched Readings Example

Example in how to get all the channel readings at the same time

Listing 4: examples/as7341_batched_readings.py

```
# SPDX-FileCopyrightText: 2020 Bryan Siepert, written for Adafruit Industries
# SPDX-License-Identifier: MIT
from time import sleep
import board
from adafruit_as7341 import AS7341

i2c = board.I2C()  # uses board.SCL and board.SDA
sensor = AS7341(i2c)


def bar_graph(read_value):
    scaled = int(read_value / 1000)
    return "[%5d] " % read_value + (scaled * "*")


while True:
    sensor_channels = sensor.all_channels
    print("F1 - 415nm/Violet  %s" % bar_graph(sensor_channels[0]))
    print("F2 - 445nm//Indigo %s" % bar_graph(sensor_channels[1]))
    print("F3 - 480nm//Blue   %s" % bar_graph(sensor_channels[2]))
    print("F4 - 515nm//Cyan   %s" % bar_graph(sensor_channels[3]))
    print("F5 - 555nm/Green   %s" % bar_graph(sensor_channels[4]))
    print("F6 - 590nm/Yellow  %s" % bar_graph(sensor_channels[5]))
    print("F7 - 630nm/Orange  %s" % bar_graph(sensor_channels[6]))
    print("F8 - 680nm/Red     %s" % bar_graph(sensor_channels[7]))
    print("\n----------------------------------------------")

    sleep(1)
```

6.5 adafruit_as7341

CircuitPython library for use with the Adafruit AS7341 breakout

- Author(s): Bryan Siepert

6.5.1 Implementation Notes

Hardware:

- Adafruit AS7341 Breakout (Product ID: 4698)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://circuitpython.org/downloads
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice
- Adafruit's Register library: https://github.com/adafruit/Adafruit_CircuitPython_Register

class adafruit_as7341.**AS7341**(*i2c_bus*, *address=57*)
Library for the AS7341 Sensor

Parameters

- **i2c_bus** (*I2C*) – The I2C bus the device is connected to
- **address** (*int*) – The I2C device address. Defaults to `0x39`

Quickstart: Importing and using the device

Here is an example of using the *AS7341*. First you will need to import the libraries to use the sensor

```
import board
from adafruit_as7341 import AS7341
```

Once this is done you can define your `board.I2C` object and define your sensor object

```
i2c = board.I2C()  # uses board.SCL and board.SDA
sensor = AS7341(i2c)
```

Now you have access to the different channels

```
channel_415nm = channel_415nm
channel_445nm = channel_445nm
channel_480nm = channel_480nm
channel_515nm = channel_515nm
channel_555nm = channel_555nm
channel_590nm = channel_590nm
channel_630nm = channel_630nm
channel_680nm = channel_680nm
```

all_channels
The current readings for all six ADC channels

astep
The integration time step size in 2.78 microsecond increments

atime
The integration time step count. Total integration time will be `(ATIME + 1) * (ASTEP + 1) * 2.78µS`

channel_415nm
The current reading for the 415nm band

channel_445nm
The current reading for the 445nm band

channel_480nm
The current reading for the 480nm band

channel_515nm
The current reading for the 515nm band

channel_555nm
The current reading for the 555nm band

channel_590nm
The current reading for the 590nm band

channel_630nm
The current reading for the 630nm band

channel_680nm
The current reading for the 680nm band

flicker_detected
The flicker frequency detected in Hertz

flicker_detection_enabled
The flicker detection status of the sensor. True if the sensor is configured to detect flickers. Currently only 1000Hz and 1200Hz flicker detection is supported

gain
The ADC gain multiplier. Must be a valid `adafruit_as7341.Gain()`

initialize()
Configure the sensors with the default settings

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index