AdafruitAdafruit$_{I}OLibraryDocumentation$

Release 1.0

Brent Rubell

Jan 09, 2020

Contents

CircuitPython wrapper library for communicating with Adafruit IO.

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-adafruitio
```

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

```
sudo pip3 install adafruit-circuitpython-adafruitio
```

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-adafruitio
```

CHAPTER 3

Usage Example

Usage examples for the Adafruit IO HTTP API are within the examples/http folder.

Usage examples for the Adafruit IO MQTT API are within the examples/mqtt folder.

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/http/adafruit_io_simpletest.py

```
"""
Sending data to Adafruit IO and receiving it.
"""
from random import randint
import board
import busio
from digitalio import DigitalInOut

# ESP32 SPI
from adafruit_esp32spi import adafruit_esp32spi, adafruit_esp32spi_wifimanager

# Import NeoPixel Library
import neopixel

# Import Adafruit IO HTTP Client
from adafruit_io.adafruit_io import IO_HTTP, AdafruitIO_RequestError

# Get wifi details and more from a secrets.py file
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# ESP32 Setup
try:
    esp32_cs = DigitalInOut(board.ESP_CS)
```

(continues on next page)

(continued from previous page)

```
    esp32_ready = DigitalInOut(board.ESP_BUSY)
    esp32_reset = DigitalInOut(board.ESP_RESET)
except AttributeError:
    esp32_cs = DigitalInOut(board.D9)
    esp32_ready = DigitalInOut(board.D10)
    esp32_reset = DigitalInOut(board.D5)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)
status_light = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.2) # Uncomment for␣
↪Most Boards
"""Uncomment below for ItsyBitsy M4"""
#status_light = dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1, brightness=0.
↪2)
wifi = adafruit_esp32spi_wifimanager.ESPSPI_WiFiManager(esp, secrets, status_light)

# Set your Adafruit IO Username and Key in secrets.py
# (visit io.adafruit.com if you need to create an account,
# or if you need your Adafruit IO key.)
aio_username = secrets['aio_username']
aio_key = secrets['aio_key']

# Create an instance of the Adafruit IO HTTP client
io = IO_HTTP(aio_username, aio_key, wifi)

try:
    # Get the 'temperature' feed from Adafruit IO
    temperature_feed = io.get_feed('temperature')
except AdafruitIO_RequestError:
    # If no 'temperature' feed exists, create one
    temperature_feed = io.create_new_feed('temperature')

# Send random integer values to the feed
random_value = randint(0, 50)
print('Sending {0} to temperature feed...'.format(random_value))
io.send_data(temperature_feed['key'], random_value)
print('Data sent!')

# Retrieve data value from the feed
print('Retrieving data from temperature feed...')
received_data = io.receive_data(temperature_feed['key'])
print('Data from temperature feed: ', received_data['value'])
```

6.2 API

6.2.1 adafruit_io

A CircuitPython library for communicating with Adafruit IO.

- Author(s): Brent Rubell for Adafruit Industries

6.2.1.1 Implementation Notes

Software and Dependencies:

- **Adafruit CircuitPython firmware for the supported boards:** https://github.com/adafruit/circuitpython/releases

class adafruit_io.adafruit_io.**IO_HTTP**(*adafruit_io_username*, *adafruit_io_key*, *wifi_manager*)

Client for interacting with the Adafruit IO HTTP API. https://io.adafruit.com/api/docs/#adafruit-io-http-api

param str adafruit_io_username Adafruit IO Username

param str adafruit_io_key Adafruit IO Key

param wifi_manager WiFiManager object from ESPSPI_WiFiManager or ESPAT_WiFiManager

add_feed_to_group(*group_key*, *feed_key*)

Adds an existing feed to a group :param str group_key: Group :param str feed_key: Feed to add to the group

create_new_feed(*feed_key*, *feed_desc=None*, *feed_license=None*)

Creates a new Adafruit IO feed. :param str feed_key: Adafruit IO Feed Key :param str feed_desc: Optional description of feed :param str feed_license: Optional feed license

create_new_group(*group_key*, *group_description*)

Creates a new Adafruit IO Group. :param str group_key: Adafruit IO Group Key :param str group_description: Brief summary about the group

delete_data(*feed_key*, *data_id*)

Deletes an existing Data point from a feed. :param string feed: Adafruit IO feed key :param string data_id: Data point to delete from the feed

delete_feed(*feed_key*)

Deletes an existing feed. :param str feed_key: Valid feed key

delete_group(*group_key*)

Deletes an existing group. :param str group_key: Adafruit IO Group Key

get_feed(*feed_key*, *detailed=False*)

Returns an Adafruit IO feed based on the feed key :param str feed_key: Adafruit IO Feed Key :param bool detailed: Returns a more verbose feed record

get_group(*group_key*)

Returns Group based on Group Key :param str group_key: Adafruit IO Group Key

receive_data(*feed_key*)

Return the most recent value for the specified feed. :param string feed_key: Adafruit IO feed key

receive_random_data(*generator_id*)

Get data from the Adafruit IO Random Data Stream Service :param int generator_id: Specified randomizer record

receive_time()

Returns a struct_time from the Adafruit IO Server based on the device's IP address. https://circuitpython.readthedocs.io/en/latest/shared-bindings/time/__init__.html#time.struct_time

receive_weather(*weather_id*)

Get data from the Adafruit IO Weather Forecast Service NOTE: This service is avaliable to Adafruit IO Plus subscribers only. :param int weather_id: ID for retrieving a specified weather record.

send_data(*feed_key*, *data*, *metadata=None*, *precision=None*)

Sends value data to a specified Adafruit IO feed. :param str feed_key: Adafruit IO feed key :param str data: Data to send to the Adafruit IO feed :param dict metadata: Optional metadata associated with the data :param int precision: Optional amount of precision points to send with floating point data

class adafruit_io.adafruit_io.**IO_MQTT**(*mqtt_client*)

Client for interacting with the Adafruit IO MQTT API. https://io.adafruit.com/api/docs/mqtt.html#adafruit-io-mqtt-api

Parameters **mqtt_client** (*MiniMQTT*) – MiniMQTT Client object.

connect()

Connects to the Adafruit IO MQTT Broker. Must be called before any other API methods are called.

disconnect()

Disconnects from Adafruit IO MQTT Broker.

get(*feed_key*)

Calling this method will make Adafruit IO publish the most recent value on feed_key. https://io.adafruit.com/api/docs/mqtt.html#retained-values :param str feed_key: Adafruit IO Feed key.

Example of obtaining a recently published value on a feed: ..code-block:: python

io.get('temperature')

is_connected

Returns if connected to Adafruit IO MQTT Broker.

loop()

Manually process messages from Adafruit IO. Call this method to check incoming subscription messages.

Example usage of polling the message queue using loop.

..code-block:: python

while True: io.loop()

loop_blocking()

Starts a blocking loop and to processes messages from Adafruit IO. Code below this call will not run.

publish(*feed_key*, *data*, *metadata=None*, *shared_user=None*, *is_group=False*)

Publishes to an An Adafruit IO Feed.

Parameters

- **feed_key** (*str*) – Adafruit IO Feed key.
- **data** (*float*) – Data to publish to the feed or group.
- **data** – Data to publish to the feed or group.
- **data** – Data to publish to the feed or group.
- **metadata** (*str*) – Optional metadata associated with the data.
- **shared_user** (*str*) – Owner of the Adafruit IO feed, required for feed sharing.
- **is_group** (*bool*) – Set True if publishing to an Adafruit IO Group.

Example of publishing an integer to Adafruit IO on feed 'temperature'. ..code-block:: python

client.publish('temperature', 30)

Example of publishing a floating point value to feed 'temperature'. ..code-block:: python

client.publish('temperature', 3.14)

Example of publishing a string to feed 'temperature'. ..code-block:: python

client.publish('temperature, 'thirty degrees')

Example of publishing an integer to group 'weatherstation'. ..code-block:: python

client.publish('weatherstation', 12, is_group=True)

Example of publishing to a shared feed. ..code-block:: python

client.publish('temperature', shared_user='myfriend')

Example of publishing a value along with locational metadata to a feed. ..code-block:: python

data = 42 # format: "lat, lon, ele" metadata = "40.726190, -74.005334, -6" io.publish("location-feed", data, metadata)

publish_multiple(*feeds_and_data*, *timeout=3*, *is_group=False*)

Publishes multiple data points to multiple feeds or groups with a variable timeout.

Parameters

- **feeds_and_data** (*str*) – List of tuples containing topic strings and data values.
- **timeout** (*int*) – Delay between publishing data points to Adafruit IO, in seconds.
- **is_group** (*bool*) – Set to True if you're publishing to a group.

Example of publishing multiple data points on different feeds to Adafruit IO: ..code-block:: python

client.publish_multiple([('humidity', 24.5), ('temperature', 54)])

subscribe(*feed_key=None*, *group_key=None*, *shared_user=None*)

Subscribes to your Adafruit IO feed or group. Can also subscribe to someone else's feed. :param str feed_key: Adafruit IO Feed key. :param str group_key: Adafruit IO Group key. :param str shared_user: Owner of the Adafruit IO feed, required for shared feeds.

Example of subscribing to an Adafruit IO Feed named 'temperature'.

```
client.subscribe('temperature')
```

Example of subscribing to two Adafruit IO feeds: 'temperature' and 'humidity'.

```
client.subscribe([('temperature'), ('humidity')])
```

subscribe_to_errors()

Subscribes to your personal Adafruit IO /errors topic. Notifies you of errors relating to publish/subscribe calls.

subscribe_to_randomizer(*randomizer_id*)

Subscribes to a random data stream created by the Adafruit IO Words service. :param int randomizer_id: Random word record you want data for.

subscribe_to_throttling()

Subscribes to your personal Adafruit IO /throttle topic. https://io.adafruit.com/api/docs/mqtt.html#mqtt-api-rate-limiting

subscribe_to_time(*time_type*)

Adafruit IO provides some built-in MQTT topics for getting the current server time. :param str time_type: Current Adafruit IO server time. Can be 'seconds', 'millis', or 'iso'. Information about these topics can be found on the Adafruit IO MQTT API Docs.: https://io.adafruit.com/api/docs/mqtt.html#time-topics

subscribe_to_weather(*weather_record*, *forecast*)

Subscribes to a weather forecast using the Adafruit IO PLUS weather service. This feature is only avaliable to Adafruit IO PLUS subscribers. :param int weather_record: Weather record you want data for. :param str forecast: Forecast data you'd like to recieve.

unsubscribe(*feed_key=None*, *group_key=None*, *shared_user=None*)

Unsubscribes from an Adafruit IO feed or group. Can also subscribe to someone else's feed. :param str

feed_key: Adafruit IO Feed key. :param str group_key: Adafruit IO Group key. :param str shared_user: Owner of the Adafruit IO feed, required for shared feeds.

Example of unsubscribing from a feed.

```
client.unsubscribe('temperature')
```

Example of unsubscribing from two feeds: 'temperature' and 'humidity'

```
client.unsubscribe([('temperature'), ('humidity')])
```

Example of unsubscribing from a shared feed.

```
client.unsubscribe('temperature', shared_user='adabot')
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

U