AdafruitPortalBase Library Documentation

Release 1.0

Melissa LeBlanc-Williams

Jun 05, 2021

CONTENTS

Base Library for the Portal-style libraries. This library only contains base classes and is not intended to be run on its own.

CHAPTER

ONE

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

TWO

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-portalbase
```

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

```
sudo pip3 install adafruit-circuitpython-portalbase
```

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-portalbase
```

CHAPTER

THREE

CONTRIBUTING

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER

FOUR

DOCUMENTATION

For information on building library documentation, please check out this guide.

CHAPTER

FIVE

TABLE OF CONTENTS

5.1 adafruit_portalbase

Base Library for the Portal-style libraries.

- Author(s): Melissa LeBlanc-Williams

5.1.1 Implementation Notes

Software and Dependencies:

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

class adafruit_portalbase.**PortalBase**(*network*, *graphics*, *, *url=None*, *headers=None*, *json_path=None*, *regexp_path=None*, *json_transform=None*, *success_callback=None*, *debug=False*)

Class representing the Adafruit MagTag.

Parameters

- **network** – An initialized network class instance.
- **graphics** – An initialized graphics class instance.
- **url** – The URL of your data source. Defaults to `None`.
- **headers** – The headers for authentication, typically used by Azure API's.
- **json_path** – The list of json traversal to get data out of. Can be list of lists for multiple data points. Defaults to `None` to not use json.
- **regexp_path** – The list of regexp strings to get data out (use a single regexp group). Can be list of regexps for multiple data points. Defaults to `None` to not use regexp.
- **default_bg** – The path to your default background image file or a hex color. Defaults to 0x000000.
- **status_neopixel** – The pin for the status NeoPixel. Use `board.NEOPIXEL` for the on-board NeoPixel. Defaults to `None`, to not use the status LED
- **json_transform** – A function or a list of functions to call with the parsed JSON. Changes and additions are permitted for the `dict` object.
- **success_callback** – A function we'll call if you like, when we fetch data successfully. Defaults to `None`.
- **debug** – Turn on debug print outs. Defaults to False.

add_text(*text_position=(0, 0)*, *text_font=<fontio.BuiltinFont object>*, *text_color=0*, *text_wrap=0*, *text_maxlen=0*, *text_transform=None*, *text_scale=1*, *line_spacing=1.25*, *text_anchor_point=(0, 0.5)*, *is_data=True*, *text=None*)

Add text labels with settings

Parameters

- **text_font** (*str*) – The path to your font file for your data text display.
- **text_position** – The position of your extracted text on the display in an (x, y) tuple. Can be a list of tuples for when there's a list of json_paths, for example.
- **text_color** – The color of the text, in 0xRRGGBB format. Can be a list of colors for when there's multiple texts. Defaults to `None`.
- **text_wrap** – When non-zero, the maximum number of characters on each line before text is wrapped. (for long text data chunks). Defaults to 0, no wrapping.
- **text_maxlen** – The max length of the text. If non-zero, it will be truncated to this length. Defaults to 0.
- **text_transform** – A function that will be called on the text before display
- **text_scale** (*int*) – The factor to scale the default size of the text by
- **line_spacing** (*float*) – The factor to space the lines apart
- **text_anchor_point** ((*float*,*float*)) – Values between 0 and 1 to indicate where the text position is relative to the label
- **is_data** (*bool*) – If True, fetch will attempt to update the label
- **text** (*str*) – If this is provided, it will set the initial text of the label.

enter_light_sleep(*sleep_time*)

Enter light sleep and resume the program after a certain period of time.

See https://circuitpython.readthedocs.io/en/latest/shared-bindings/alarm/index.html for more details.

Parameters **sleep_time** (*float*) – The amount of time to sleep in seconds

exit_and_deep_sleep(*sleep_time*)

Stops the current program and enters deep sleep. The program is restarted from the beginning after a certain period of time.

See https://circuitpython.readthedocs.io/en/latest/shared-bindings/alarm/index.html for more details.

Parameters **sleep_time** (*float*) – The amount of time to sleep in seconds

fetch(*refresh_url=None*, *timeout=10*)

Fetch data from the url we initialized with, perfom any parsing, and display text or graphics. This function does pretty much everything Optionally update the URL

Parameters

- **refresh_url** (*str*) – The overriding URL to fetch from. Defaults to `None`.
- **timeout** (*int*) – The timeout period in seconds.

get_io_data(*feed_key*)

Return all values from the Adafruit IO Feed Data that matches the feed key

Parameters **feed_key** (*str*) – Name of feed key to receive data from.

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

Return the Adafruit IO Feed that matches the feed key

Parameters

- **feed_key** (*str*) – Name of feed key to match.
- **detailed** (*bool*) – Whether to return additional detailed information

get_io_group(*group_key*)
Return the Adafruit IO Group that matches the group key

Parameters **group_key** (*str*) – Name of group key to match.

get_local_time(*location=None*)
Accessor function for get_local_time()

static html_color_convert(*color*)
Convert an HTML color code to an integer

Parameters **color** – The color value to be converted

property json_path
Get or set the list of json traversal to get data out of. Can be list of lists for multiple data points.

preload_font(*glyphs=None*, *index=0*)
Preload font.

Parameters **glyphs** – The font glyphs to load. Defaults to `None`, uses alphanumeric glyphs if None.

push_to_io(*feed_key*, *data*)
Push data to an adafruit.io feed

Parameters

- **feed_key** (*str*) – Name of feed key to push data to.
- **data** – data to send to feed

set_background(*file_or_color*, *position=None*)
The background image to a bitmap file.

Parameters **file_or_color** – The filename of the chosen background image, or a hex color.

set_headers(*headers*)
Set the headers used by fetch().

Parameters **headers** – The new header dictionary

set_text(*val*, *index=0*)
Display text, with indexing into our list of text boxes.

Parameters

- **val** (*str*) – The text to be displayed
- **index** – Defaults to 0.

set_text_color(*color*, *index=0*)
Update the text color, with indexing into our list of text boxes.

Parameters

- **color** (*int*) – The color value to be used
- **index** – Defaults to 0.

static wrap_nicely(*string*, *max_chars*)
A helper that will return a list of lines with word-break wrapping.

Parameters

- **string** (*str*) – The text to be wrapped.
- **max_chars** (*int*) – The maximum number of characters on a line before wrapping.

5.2 adafruit_portalbase.graphics

Base Library for the Portal-style libraries.

- Author(s): Melissa LeBlanc-Williams

5.2.1 Implementation Notes

Software and Dependencies:

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

class adafruit_portalbase.graphics.**GraphicsBase**(*display*, *, *default_bg=0*, *scale=1*, *debug=False*)
Graphics Base Class for the Portal-style libraries.

Parameters

- **display** – An initialized display.
- **default_bg** – The path to your default background image file or a hex color. Defaults to 0x000000.
- **auto_refresh** (*bool*) – Automatically refresh the eInk after writing to displayio. Defaults to True.
- **debug** – Turn on debug print outs. Defaults to False.

qrcode(*qr_data*, *, *qr_size=1*, *x=0*, *y=0*, *qr_color=0*)
Display a QR code

Parameters

- **qr_data** – The data for the QR code.
- **qr_size** (*int*) – The scale of the QR code.
- **x** – The x position of upper left corner of the QR code on the display.
- **y** – The y position of upper left corner of the QR code on the display.

set_background(*file_or_color*, *position=None*)
The background image to a bitmap file.

Parameters

- **file_or_color** – The filename of the chosen background image, or a hex color.
- **position** (*tuple*) – Optional x and y coordinates to place the background at.

5.3 adafruit_portalbase.network

Base Library for the Portal-style libraries.

- Author(s): Melissa LeBlanc-Williams

5.3.1 Implementation Notes

Software and Dependencies:

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

exception adafruit_portalbase.network.**HttpError**
HTTP Specific Error

class adafruit_portalbase.network.**NetworkBase**(*wifi_module*, *, *extract_values=True*, *debug=False*, *secrets_data=None*)
Network Base Class for the Portal-style libraries.

Parameters

- **wifi_module** – An initialized WiFi Module that encapsulates the WiFi communications
- **extract_values** (*bool*) – If true, single-length fetched values are automatically extracted from lists and tuples. Defaults to `True`.
- **debug** – Turn on debug print outs. Defaults to False.
- **secrets_data** (*list*) – An optional list in place of the data contained in the secrets.py file

add_json_content_type(*content_type*)
Add a JSON content type

Parameters **type** (*str*) – The content JSON type like 'application/json'

add_json_transform(*json_transform*)
Add a function that is applied to JSON data when data is fetched

Parameters **json_transform** – A function or a list of functions to call with the parsed JSON. Changes and additions are permitted for the `dict` object.

check_response(*response*)
Check the response object status code, change the lights, and return content type

Parameters **response** – The response object from a network call

connect()
Connect to WiFi using the settings found in secrets.py

fetch(*url*, *, *headers=None*, *timeout=10*)
Fetch data from the specified url and return a response object

Parameters

- **url** (*str*) – The URL to fetch from.
- **headers** (*dict*) – Extra headers to include in the request.
- **timeout** (*int*) – The timeout period in seconds.

fetch_data(*url*, *, *headers=None*, *json_path=None*, *regexp_path=None*, *timeout=10*)
Fetch data from the specified url and perfom any parsing

Parameters

- **url** (*str*) – The URL to fetch from.
- **headers** (*dict*) – Extra headers to include in the request.
- **json_path** – The path to drill down into the JSON data.
- **regexp_path** – The path formatted as a regular expression to search the text data.
- **timeout** (*int*) – The timeout period in seconds.

get_io_data(*feed_key*)
Return all values from Adafruit IO Feed Data that matches the feed key

Parameters **feed_key** (*str*) – Name of feed key to receive data from.

get_io_feed(*feed_key*, *detailed=False*)
Return the Adafruit IO Feed that matches the feed key

Parameters

- **feed_key** (*str*) – Name of feed key to match.
- **detailed** (*bool*) – Whether to return additional detailed information

get_io_group(*group_key*)
Return the Adafruit IO Group that matches the group key

Parameters **group_key** (*str*) – Name of group key to match.

get_local_time(*location=None*)
Fetch and "set" the local time of this microcontroller to the local time at the location, using an internet time API.

Parameters **location** (*str*) – Your city and country, e.g. `"America/New_York"`.

get_strftime(*time_format*, *location=None*)
Fetch a custom strftime relative to your location.

Parameters **location** (*str*) – Your city and country, e.g. `"America/New_York"`.

static json_traverse(*json*, *path*)
Traverse down the specified JSON path and return the value or values

Parameters

- **json** – JSON data to traverse
- **path** (*list*) – The path that we want to follow

neo_status(*value*)
The status NeoPixel.

Parameters **value** – The color to change the NeoPixel.

process_json(*json_data*, *json_path*)
Process JSON content

Parameters

- **json_data** (*dict*) – The JSON data as a dict
- **json_path** – The path to drill down into the JSON data.

static process_text(*text*, *regexp_path*)
Process text content

Parameters

- **text** (*str*) – The entire text content
- **regexp_path** – The path formatted as a regular expression to search the text data.

push_to_io(*feed_key*, *data*)
Push data to an adafruit.io feed

Parameters

- **feed_key** (*str*) – Name of feed key to push data to.
- **data** – data to send to feed

static url_encode(*url*)
A function to perform minimal URL encoding

wget(*url*, *filename*, *, *chunk_size=12000*)
Download a url and save to filename location, like the command wget.

Parameters

- **url** – The URL from which to obtain the data.
- **filename** – The name of the file to save the data to.
- **chunk_size** – how much data to read/write at a time.

5.4 adafruit_portalbase.wifi_esp32s2

WiFi Helper module for the ESP32-S2 based boards.

- Author(s): Melissa LeBlanc-Williams

5.4.1 Implementation Notes

Software and Dependencies:

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

class adafruit_portalbase.wifi_esp32s2.**WiFi**(*, *status_led=None*)
Class representing the WiFi portion of the ESP32-S2.

Parameters **status_led** – The initialized object for status DotStar, NeoPixel, or RGB LED. Defaults to `None`, to not use the status LED

connect(*ssid*, *password*)
Connect to the WiFi Network using the information provided

Parameters

- **ssid** – The WiFi name
- **password** – The WiFi password

property enabled
Return whether the WiFi Radio is enabled

property ip_address
Return the IP Version 4 Address

property is_connected
Return whether we have already connected since reconnections are handled automatically.

neo_status(*value*)
The status DotStar.

Parameters **value** – The color to change the DotStar.

5.5 adafruit_portalbase.wifi_coprocessor

WiFi Helper module for the board using the WiFi CoProcessor.

- Author(s): Melissa LeBlanc-Williams

5.5.1 Implementation Notes

Software and Dependencies:

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

class adafruit_portalbase.wifi_coprocessor.**WiFi**(*, *status_led=None*, *esp=None*, *external_spi=None*)
Class representing the ESP.

Parameters

- **status_led** – The initialized object for status DotStar, NeoPixel, or RGB LED. Defaults to `None`, to not use the status LED
- **esp** – A passed ESP32 object, Can be used in cases where the ESP32 chip needs to be used before calling the pyportal class. Defaults to `None`.
- **external_spi** (*busio.SPI*) – A previously declared spi object. Defaults to `None`.

connect(*ssid*, *password*)
Connect to WiFi using the settings found in secrets.py

property enabled
Not currently disablable on the ESP32 Coprocessor

property is_connected
Return whether we are connected.

manager(*secrets*)
Initialize the WiFi Manager if it hasn't been cached and return it

neo_status(*value*)
The status NeoPixel.

Parameters **value** – The color to change the NeoPixel.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX

J

M

N

P

Q

S

U

W