AdafruitRequests Library Documentation

Release 1.0

ladyada

Jun 07, 2021

Contents

A requests-like library for HTTP commands.

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-requests
```

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

```
sudo pip3 install adafruit-circuitpython-requests
```

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-requests
```

CHAPTER 3

Usage Example

Usage examples are within the *Simple test* subfolder of this library.

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/requests_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

# adafruit_requests usage with an esp32spi_socket
import board
import busio
from digitalio import DigitalInOut
import adafruit_esp32spi.adafruit_esp32spi_socket as socket
from adafruit_esp32spi import adafruit_esp32spi
import adafruit_requests as requests

# Add a secrets.py to your filesystem that has a dictionary called secrets with "ssid
↪" and
# "password" keys with your WiFi credentials. DO NOT share that file or commit it␣
↪into Git or other
# source control.
# pylint: disable=no-name-in-module,wrong-import-order
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# If you are using a board with pre-defined ESP32 Pins:
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)
```

(continues on next page)

(continued from previous page)

```
26
27  # If you have an externally connected ESP32:
28  # esp32_cs = DigitalInOut(board.D9)
29  # esp32_ready = DigitalInOut(board.D10)
30  # esp32_reset = DigitalInOut(board.D5)
31
32  # If you have an AirLift Featherwing or ItsyBitsy Airlift:
33  # esp32_cs = DigitalInOut(board.D13)
34  # esp32_ready = DigitalInOut(board.D11)
35  # esp32_reset = DigitalInOut(board.D12)
36
37  spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
38  esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)
39
40  print("Connecting to AP...")
41  while not esp.is_connected:
42      try:
43          esp.connect_AP(secrets["ssid"], secrets["password"])
44      except RuntimeError as e:
45          print("could not connect to AP, retrying: ", e)
46          continue
47  print("Connected to", str(esp.ssid, "utf-8"), "\tRSSI:", esp.rssi)
48
49  # Initialize a requests object with a socket and esp32spi interface
50  socket.set_interface(esp)
51  requests.set_socket(socket, esp)
52
53  TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
54  JSON_GET_URL = "https://httpbin.org/get"
55  JSON_POST_URL = "https://httpbin.org/post"
56
57  print("Fetching text from %s" % TEXT_URL)
58  response = requests.get(TEXT_URL)
59  print("-" * 40)
60
61  print("Text Response: ", response.text)
62  print("-" * 40)
63  response.close()
64
65  print("Fetching JSON data from %s" % JSON_GET_URL)
66  response = requests.get(JSON_GET_URL)
67  print("-" * 40)
68
69  print("JSON Response: ", response.json())
70  print("-" * 40)
71  response.close()
72
73  data = "31F"
74  print("POSTing data to {0}: {1}".format(JSON_POST_URL, data))
75  response = requests.post(JSON_POST_URL, data=data)
76  print("-" * 40)
77
78  json_resp = response.json()
79  # Parse out the 'data' key from json_resp dict.
80  print("Data received from server:", json_resp["data"])
81  print("-" * 40)
82  response.close()
```

(continues on next page)

(continued from previous page)

```
83
84 json_data = {"Date": "July 25, 2019"}
85 print("POSTing data to {0}: {1}".format(JSON_POST_URL, json_data))
86 response = requests.post(JSON_POST_URL, json=json_data)
87 print("-" * 40)
88
89 json_resp = response.json()
90 # Parse out the 'json' key from json_resp dict.
91 print("JSON Data received from server:", json_resp["json"])
92 print("-" * 40)
93 response.close()
```

6.2 adafruit_requests

A requests-like library for web interfacing

- Author(s): ladyada, Paul Sokolovsky, Scott Shawcroft

6.2.1 Implementation Notes

Adapted from https://github.com/micropython/micropython-lib/tree/master/urequests

micropython-lib consists of multiple modules from different sources and authors. Each module comes under its own licensing terms. Short name of a license can be found in a file within a module directory (usually metadata.txt or setup.py). Complete text of each license used is provided at https://github.com/micropython/micropython-lib/blob/master/LICENSE

author='Paul Sokolovsky' license='MIT'

Software and Dependencies:

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

exception adafruit_requests.**OutOfRetries**
Raised when requests has retried to make a request unsuccessfully.

class adafruit_requests.**Response**(*sock*, *session=None*)
The response from a request, contains all the headers/content

close()
Drain the remaining ESP socket buffers. We assume we already got what we wanted.

content
The HTTP content direct from the socket, as bytes

headers
The response headers. Does not include headers from the trailer until the content has been read.

iter_content(*chunk_size=1*, *decode_unicode=False*)
An iterator that will stream data by only reading 'chunk_size' bytes and yielding them, when we can't buffer the whole datastream

json()
The HTTP content, parsed into a json dictionary

text
The HTTP content, encoded into a string according to the HTTP header encoding

class adafruit_requests.**Session**(*socket_pool*, *ssl_context=None*)
HTTP session that shares sockets and ssl context.

delete(*url*, ***kw*)
Send HTTP DELETE request

get(*url*, ***kw*)
Send HTTP GET request

head(*url*, ***kw*)
Send HTTP HEAD request

patch(*url*, ***kw*)
Send HTTP PATCH request

post(*url*, ***kw*)
Send HTTP POST request

put(*url*, ***kw*)
Send HTTP PUT request

request(*method*, *url*, *data=None*, *json=None*, *headers=None*, *stream=False*, *timeout=60*)
Perform an HTTP request to the given url which we will parse to determine whether to use SSL ('https://') or not. We can also send some provided 'data' or a json dictionary which we will stringify. 'headers' is optional HTTP headers sent along. 'stream' will determine if we buffer everything, or whether to only read only when requested

adafruit_requests.**delete**(*url*, ***kw*)
Send HTTP DELETE request

adafruit_requests.**get**(*url*, ***kw*)
Send HTTP GET request

adafruit_requests.**head**(*url*, ***kw*)
Send HTTP HEAD request

adafruit_requests.**patch**(*url*, ***kw*)
Send HTTP PATCH request

adafruit_requests.**post**(*url*, ***kw*)
Send HTTP POST request

adafruit_requests.**put**(*url*, ***kw*)
Send HTTP PUT request

adafruit_requests.**request**(*method*, *url*, *data=None*, *json=None*, *headers=None*, *stream=False*, *timeout=1*)
Send HTTP request

adafruit_requests.**set_socket**(*sock*, *iface=None*)
Legacy API for setting the socket and network interface. Use a *Session* instead.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index