# Contents

1 Dependencies .................................... 3
2 Installing from PyPI .......................... 5
3 Usage Example ............................... 7
4 Contributing ................................... 9
5 Documentation ................................ 11
6 Table of Contents ............................. 13
   6.1 Simple test .................................. 13
   6.2 adafruit_requests .......................... 15
      6.2.1 Implementation Notes ................. 15
7 Indices and tables ........................... 17
Python Module Index .......................... 19
Index ........................................... 21
A requests-like library for HTTP commands.
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.
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:

```bash
pip3 install adafruit-circuitpython-requests
```

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

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

To install in a virtual environment in your current project:

```bash
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.
Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.
For information on building library documentation, please check out this guide.
6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/requests_simpletest.py

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

import board
import busio
from digitalio import DigitalInOut
import adafruit_esp32spi.adafruit_esp32spi_socket as socket
import adafruit_requests as requests

# Add a secrets.py to your filesystem that has a dictionary called secrets with "ssid"
# "password" keys with your WiFi credentials. DO NOT share that file or commit it
# into Git or other
# source control.
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
else:
    from secrets import secrets

# 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)
# If you have an externally connected ESP32:
# esp32_cs = DigitalInOut(board.D9)
# esp32_ready = DigitalInOut(board.D10)
# esp32_reset = DigitalInOut(board.D5)

# If you have an AirLift Featherwing or ItsyBitsy Airlift:
# esp32_cs = DigitalInOut(board.D13)
# esp32_ready = DigitalInOut(board.D11)
# esp32_reset = DigitalInOut(board.D12)

spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)

print("Connecting to AP...")
while not esp.is_connected:
    try:
        esp.connect_AP(secrets["ssid"], secrets["password"])
    except RuntimeError as e:
        print("could not connect to AP, retrying: ", e)
        continue
    print("Connected to", str(esp.ssid, "utf-8"), "	RSSI: ", esp.rssi)

# Initialize a requests object with a socket and esp32spi interface
socket.set_interface(esp)
requests.set_socket(socket, esp)

TEXT_URL = "http://wifitest.adafruit.com/testwifi/index.html"
JSON_GET_URL = "https://httpbin.org/get"
JSON_POST_URL = "https://httpbin.org/post"

print("Fetching text from %s" % TEXT_URL)
response = requests.get(TEXT_URL)
print("-
 response.close()

print("Text Response: ", response.text)
print("-
 response.close()

print("Fetching JSON data from %s" % JSON_GET_URL)
response = requests.get(JSON_GET_URL)
print("-
 response.close()

print("JSON Response: ", response.json())
print("-
 response.close()

data = "31F"
print("POSTing data to {0}: (1)".format(JSON_POST_URL, data))
response = requests.post(JSON_POST_URL, data=data)
print("-
 response.close()

json_resp = response.json()
# Parse out the 'data' key from json_resp dict.
print("Data received from server:"; json_resp["data")
print("-
 response.close()

(continues on next page)
json_data = \{"Date": "July 25, 2019"\}
print("POSTing data to {0}: {1}".format(JSON_POST_URL, json_data))
response = requests.post(JSON_POST_URL, json=json_data)
print("-" * 40)

json_resp = response.json()
# Parse out the 'json' key from json_resp dict.
print("JSON Data received from server:", json_resp["json"])
print("-" * 40)
response.close()
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

- adafruit_requests, 15
Index

A
adafruit_requests (module), 15

C
close() (adafruit_requests.Response method), 15
content (adafruit_requests.Response attribute), 15

D
delete() (adafruit_requests.Session method), 16
delete() (in module adafruit_requests), 16

G
get() (adafruit_requests.Session method), 16
get() (in module adafruit_requests), 16

H
head() (adafruit_requests.Session method), 16
head() (in module adafruit_requests), 16
headers (adafruit_requests.Response attribute), 15

I
iter_content() (adafruit_requests.Response method), 15

J
json() (adafruit_requests.Response method), 15

O
OutOfRetries, 15

P
patch() (adafruit_requests.Session method), 16
patch() (in module adafruit_requests), 16
post() (adafruit_requests.Session method), 16
post() (in module adafruit_requests), 16
put() (adafruit_requests.Session method), 16
put() (in module adafruit_requests), 16

R
request() (adafruit_requests.Session method), 16
request() (in module adafruit_requests), 16
Response (class in adafruit_requests), 15

S
Session (class in adafruit_requests), 15
set_socket() (in module adafruit_requests), 16

T
text (adafruit_requests.Response attribute), 15