AdafruitPyBadger Library Documentation

Release 1.0

Kattni Rembor

Apr 10, 2020

Contents

Badge-focused CircuitPython helper library for PyBadge, PyBadge LC, PyGamer and CLUE.

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-pybadger
```

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

```
sudo pip3 install adafruit-circuitpython-pybadger
```

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-pybadger
```

CHAPTER 3

Usage Example

```
from adafruit_pybadger import pybadger

pybadger.show_badge(name_string="Blinka", hello_scale=2, my_name_is_scale=2, name_
↪scale=3)

while True:
    pybadger.auto_dim_display(delay=10)
    if pybadger.button.a:
        pybadger.show_business_card(image_name="Blinka.bmp", name_string="Blinka",␣
↪name_scale=2,
                                    email_string_one="blinka@", email_string_two=
↪"adafruit.com")
    elif pybadger.button.b:
        pybadger.show_qr_code(data="https://circuitpython.org")
    elif pybadger.button.start:
        pybadger.show_badge(name_string="Blinka", hello_scale=2, my_name_is_scale=2,␣
↪name_scale=3)
```

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/pybadger_simpletest.py

```
from adafruit_pybadger import pybadger

pybadger.show_badge(
    name_string="Blinka", hello_scale=2, my_name_is_scale=2, name_scale=3
)

while True:
    pybadger.auto_dim_display(
        delay=10
    )  # Remove or comment out this line if you have the PyBadge LC
    if pybadger.button.a:
        pybadger.show_business_card(
            image_name="Blinka.bmp",
            name_string="Blinka",
            name_scale=2,
            email_string_one="blinka@",
            email_string_two="adafruit.com",
        )
    elif pybadger.button.b:
        pybadger.show_qr_code(data="https://circuitpython.org")
    elif pybadger.button.start:
        pybadger.show_badge(
            name_string="Blinka", hello_scale=2, my_name_is_scale=2, name_scale=3
        )
```

6.2 adafruit_pybadger.pybadger_base

Base class for badge-focused CircuitPython helper library.

- Author(s): Kattni Rembor

6.2.1 Implementation Notes

Hardware:

- Adafruit CLUE
- Adafruit PyBadge
- Adafruit PyBadge LC
- Adafruit PyGamer

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://circuitpython.org/downloads

class adafruit_pybadger.pybadger_base.**PyBadgerBase**
PyBadger base class.

acceleration
Accelerometer data, +/- 2G sensitivity.

auto_dim_display(*delay=5.0*, *movement_threshold=10*)
Auto-dim the display when board is not moving.

Parameters

- **delay** (*int*) – Time in seconds before display auto-dims after movement has ceased.
- **movement_threshold** (*int*) – Threshold required for movement to be considered stopped. Change to increase or decrease sensitivity.

```
from adafruit_pybadger import pybadger

while True:
    pybadger.auto_dim_display(delay=10)
```

badge_background(*background_color=(255, 0, 0)*, *rectangle_color=(255, 255, 255)*, *rectangle_drop=0.4*, *rectangle_height=0.5*)
Create a customisable badge background made up of a background color with a rectangle color block over it. Defaults are for `show_badge`.

Parameters

- **background_color** (*tuple*) – The color to fill the entire screen as a background.
- **rectangle_color** (*tuple*) – The color of a rectangle that displays over the background.
- **rectangle_drop** (*float*) – The distance from the top of the display to begin displaying the rectangle. Float represents a percentage of the display, e.g. 0.4 = 40% of the display. Defaults to `0.4`.
- **rectangle_height** (*float*) – The height of the rectangle. Float represents a percentage of the display, e.g. 0.5 = 50% of the display. Defaults to `0.5`.

```
from adafruit_pybadger import pybadger

pybadger.badge_background(background_color=pybadger.WHITE,
                          rectangle_color=pybadger.PURPLE,
                          rectangle_drop=0.2, rectangle_height=0.6)

while True:
    pybadger.show_custom_badge()
```

badge_line(*text=' '*, *color=(0, 0, 0)*, *scale=1*, *font=<sphinx.ext.autodoc.importer._MockObject object>*, *left_justify=False*, *padding_above=0*)

Add a line of text to the display. Designed to work with `badge_background` for a color-block style badge, or with `image_background` for a badge with a background image.

Parameters

- **text** (*str*) – The text to display. Defaults to displaying a blank line if no text is provided.
- **color** (*tuple*) – The color of the line of text. Defaults to `(0, 0, 0)`.
- **scale** (*int*) – The scale of the text. Must be an integer 1 or higher. Defaults to `1`.
- **font** – The font used for displaying the text. Defaults to `terminalio.FONT`.
- **left_justify** – Left-justify the line of text. Defaults to `False` which centers the font on the display.
- **padding_above** (*int*) – Add padding above the displayed line of text. A `padding_above` of `1` is equivalent to the height of one line of text, `2` is equivalent to the height of two lines of text, etc. Defaults to `0`.

The following example is designed to work on CLUE. To adapt for PyBadge or PyGamer, change the `scale` and `padding_above` values to fit the text to the display. Examples for CLUE, and PyBadge and PyGamer are included in the examples folder in the library repo.

```
from adafruit_pybadger import pybadger

pybadger.badge_background(background_color=pybadger.WHITE,
                          rectangle_color=pybadger.PURPLE,
                          rectangle_drop=0.2, rectangle_height=0.6)

pybadger.badge_line(text="@circuitpython", color=pybadger.BLINKA_PURPLE,␣
↪scale=2,
                    padding_above=2)
pybadger.badge_line(text="Blinka", color=pybadger.WHITE, scale=5,
                    padding_above=3)
pybadger.badge_line(text="CircuitPython", color=pybadger.WHITE, scale=3,
                    padding_above=1)
pybadger.badge_line(text="she/her", color=pybadger.BLINKA_PINK, scale=4,
                    padding_above=4)

while True:
    pybadger.show_custom_badge()
```

static bitmap_qr(*matrix*)

The QR code bitmap.

brightness

Display brightness. Must be a value between `0` and `1`.

image_background(*image_name=None*)
Create a bitmap image background.

Parameters **image_name** (*str*) – The name of the bitmap image as a string including `.bmp`, e.g. `"Blinka.bmp"`. Image file name is required.

```
from adafruit_pybadger import pybadger

pybadger.image_background("Blinka.bmp")

while True:
    pybadger.show_custom_badge()
```

light
Light sensor data.

pixels
Sequence like object representing the NeoPixels on the board.

play_file(*file_name*)
Play a .wav file using the onboard speaker.

Parameters **file_name** – The name of your .wav file in quotation marks including .wav

play_tone(*frequency*, *duration*)
Produce a tone using the speaker. Try changing frequency to change the pitch of the tone.

Parameters

- **frequency** (*int*) – The frequency of the tone in Hz
- **duration** (*float*) – The duration of the tone in seconds

show_badge(*, *background_color=(255, 0, 0)*, *foreground_color=(255, 255, 255)*, *background_text_color=(255, 255, 255)*, *foreground_text_color=(0, 0, 0)*, *hello_font=<sphinx.ext.autodoc.importer._MockObject object>*, *hello_scale=1*, *hello_string='HELLO'*, *my_name_is_font=<sphinx.ext.autodoc.importer._MockObject object>*, *my_name_is_scale=1*, *my_name_is_string='MY NAME IS'*, *name_font=<sphinx.ext.autodoc.importer._MockObject object>*, *name_scale=1*, *name_string='Blinka'*)
Create a "Hello My Name is"-style badge.

Parameters

- **background_color** – The color of the background. Defaults to `(255, 0, 0)`.
- **foreground_color** – The color of the foreground rectangle. Defaults to `(255, 255, 255)`.
- **background_text_color** – The color of the "HELLO MY NAME IS" text. Defaults to `(255, 255, 255)`.
- **foreground_text_color** – The color of the name text. Defaults to `(0, 0, 0)`.
- **hello_font** – The font for the "HELLO" string. Defaults to `terminalio.FONT`.
- **hello_scale** – The size scale of the "HELLO" string. Defaults to 1.
- **hello_string** – The first string of the badge. Defaults to "HELLO".
- **my_name_is_font** – The font for the "MY NAME IS" string. Defaults to `terminalio.FONT`.
- **my_name_is_scale** – The size scale of the "MY NAME IS" string. Defaults to 1.

- **my_name_is_string** – The second string of the badge. Defaults to "MY NAME IS".
- **name_font** – The font for the name string. Defaults to `terminalio.FONT`.
- **name_scale** – The size scale of the name string. Defaults to 1.
- **name_string** – The third string of the badge - change to be your name. Defaults to "Blinka".

```
from adafruit_pybadger import pybadger

while True:
    pybadger.show_badge(name_string="Blinka", hello_scale=2, my_name_is_
↪scale=2,
                        name_scale=3)
```

show_business_card(*, *image_name=None*, *name_string=None*, *name_scale=1*, *name_font=<sphinx.ext.autodoc.importer._MockObject object>*, *email_string_one=None*, *email_scale_one=1*, *email_font_one=<sphinx.ext.autodoc.importer._MockObject object>*, *email_string_two=None*, *email_scale_two=1*, *email_font_two=<sphinx.ext.autodoc.importer._MockObject object>*)

Display a bitmap image and a text string, such as a personal image and email address.

Parameters

- **image_name** (*str*) – REQUIRED. The name of the bitmap image including .bmp, e.g. `"Blinka.bmp"`.
- **name_string** (*str*) – A name string to display along the bottom of the display, e.g. `"Blinka"`.
- **name_scale** (*int*) – The scale of `name_string`. Defaults to 1.
- **name_font** – The font for the name string. Defaults to `terminalio.FONT`.
- **email_string_one** (*str*) – A string to display along the bottom of the display, e.g. `"blinka@adafruit.com"`.
- **email_scale_one** (*int*) – The scale of `email_string_one`. Defaults to 1.
- **email_font_one** – The font for the first email string. Defaults to `terminalio.FONT`.
- **email_string_two** (*str*) – A second string to display along the bottom of the display. Use if your email address is longer than one line or to add more space between the name and email address, e.g. (blinka@) `"adafruit.com"`.
- **email_scale_two** (*int*) – The scale of `email_string_two`. Defaults to 1.
- **email_font_two** – The font for the second email string. Defaults to `terminalio.FONT`.

```
from adafruit_pybadger import pybadger

while True:
    pybadger.show_business_card(image_name="Blinka.bmp", name_string="Blinka",
                                name_scale=2, email_string_one="blinka@",
                                email_string_two="adafruit.com")
```

show_custom_badge()

Call `pybadger.show_custom_badge()` to display the custom badge elements. If `show_custom_badge()` is not called, the custom badge elements will not be displayed.

show_qr_code(*data='https://circuitpython.org'*)
Generate a QR code.

Parameters **data** (*string*) – A string of data for the QR code

```
from adafruit_pybadger import pybadger

while True:
    pybadger.show_qr_code("https://adafruit.com")
```

show_terminal()
Revert to terminalio screen.

start_tone(*frequency*)
Produce a tone using the speaker. Try changing frequency to change the pitch of the tone. Use `stop_tone` to stop the tone.

Parameters **frequency** (*int*) – The frequency of the tone in Hz

stop_tone()
Use with `start_tone` to stop the tone produced.

adafruit_pybadger.pybadger_base.**load_font**(*fontname*, *text*)
Load a font and glyphs in the text string

Parameters

- **fontname** (*str*) – The full path to the font file.
- **text** (*str*) – The text containing the glyphs we want to load.

6.3 adafruit_pybadger.clue

Badge-focused CircuitPython helper library for CLUE.

- Author(s): Kattni Rembor

6.3.1 Implementation Notes

Hardware:

- Adafruit CLUE

Software and Dependencies:

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

class adafruit_pybadger.clue.**Buttons**(*a*, *b*)

a
Alias for field number 0

b
Alias for field number 1

class adafruit_pybadger.clue.**Clue**
Class that represents a single CLUE.

button
The buttons on the board.

Example use:

```
from adafruit_pybadger import pybadger

while True:
    if pybadger.button.a:
        print("Button A")
    elif pybadger.button.b:
        print("Button B")
```

light
This feature is not supported on CLUE.

play_file
This feature is not supported on CLUE.

adafruit_pybadger.clue.**clue = <adafruit_pybadger.clue.Clue object>**
Object that is automatically created on import.

6.4 adafruit_pybadger.pybadge

Badge-focused CircuitPython helper library for PyBadge, PyBadge LC and EdgeBadge. All three boards are included in this module as there is no difference in the CircuitPython builds at this time, and therefore no way to differentiate the boards from within CircuitPython.

- Author(s): Kattni Rembor

6.4.1 Implementation Notes

Hardware:

- Adafruit PyBadge
- Adafruit PyBadge LC
- Adafruit EdgeBadge

Software and Dependencies:

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

class adafruit_pybadger.pybadge.**Buttons**(*b*, *a*, *start*, *select*, *right*, *down*, *up*, *left*)

a
Alias for field number 1

b
Alias for field number 0

down
Alias for field number 5

left
Alias for field number 7

right
Alias for field number 4

select
Alias for field number 3

start
Alias for field number 2

up
Alias for field number 6

class adafruit_pybadger.pybadge.**PyBadge**
Class that represents a single PyBadge, PyBadge LC, or EdgeBadge.

button
The buttons on the board.

Example use:

```
from adafruit_pybadger import pybadger

while True:
    if pybadger.button.a:
        print("Button A")
    elif pybadger.button.b:
        print("Button B")
    elif pybadger.button.start:
        print("Button start")
    elif pybadger.button.select:
        print("Button select")
```

adafruit_pybadger.pybadge.**pybadge = <adafruit_pybadger.pybadge.PyBadge object>**
Object that is automatically created on import.

6.5 adafruit_pybadger.pygamer

Badge-focused CircuitPython helper library for PyGamer.

- Author(s): Kattni Rembor

6.5.1 Implementation Notes

Hardware:

- Adafruit PyGamer

Software and Dependencies:

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

class adafruit_pybadger.pygamer.**Buttons**(*b*, *a*, *start*, *select*, *right*, *down*, *up*, *left*)

a
Alias for field number 1

b
Alias for field number 0

down
Alias for field number 5

left
Alias for field number 7

right
Alias for field number 4

select
Alias for field number 3

start
Alias for field number 2

up
Alias for field number 6

class adafruit_pybadger.pygamer.**PyGamer**
Class that represents a single PyGamer.

button
The buttons on the board.

Example use:

```
from adafruit_pybadger import pybadger

while True:
    if pybadger.button.a:
        print("Button A")
    elif pybadger.button.b:
        print("Button B")
    elif pybadger.button.start:
        print("Button start")
    elif pybadger.button.select:
        print("Button select")
```

joystick
The joystick on the PyGamer.

adafruit_pybadger.pygamer.**pygamer = <adafruit_pybadger.pygamer.PyGamer object>**
Object that is automatically created on import.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

R

S

U