CircuitPython

DisplayIO $_{Dial Library Documentation}$

Release 1.0

Tim C

Aug 17, 2021

CONTENTS

1 Dependencies **3**

2 Installing from PyPI **5**

3 Installing to a Connected CircuitPython Device with Circup **7**

4 Usage Example **9**

5 Contributing **11**

6 Documentation **13**

7 Table of Contents **15**
- 7.1 Simple test . . . 15
- 7.2 `displayio_dial` . . . 16
  - 7.2.1 Implementation Notes . . . 16

8 Indices and tables **25**

Python Module Index **27**

Index **29**

A dial gauge widget for displaying graphical information.

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 or individual libraries can be installed using circup.

CHAPTER

TWO

INSTALLING FROM PYPI

Note: This library is not available on PyPI yet. Install documentation is included as a standard element. Stay tuned for PyPI availability!

CHAPTER

THREE

INSTALLING TO A CONNECTED CIRCUITPYTHON DEVICE WITH CIRCUP

Make sure that you have `circup` installed in your Python environment. Install it with the following command if necessary:

```
pip3 install circup
```

With `circup` installed and your CircuitPython device connected use the following command to install:

```
circup install displayio_dial
```

Or the following command to update an existing version:

```
circup update
```

CHAPTER

FOUR

USAGE EXAMPLE

See scripts in the examples directory of this repository.

CHAPTER

FIVE

CONTRIBUTING

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

CHAPTER

SIX

DOCUMENTATION

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

CHAPTER

SEVEN

TABLE OF CONTENTS

7.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/displayio_dial_simpletest.py

```
# SPDX-FileCopyrightText: 2021 Kevin Matocha
#
# SPDX-License-Identifier: MIT
##############################
"""
This is a basic demonstration of a Dial widget.
"""

import time
import board
import displayio
import terminalio
from displayio_dial import Dial

# Fonts used for the Dial tick labels
tick_font = terminalio.FONT

display = board.DISPLAY  # create the display on the PyPortal or Clue (for example)
# otherwise change this to setup the display
# for display chip driver and pinout you have (e.g. ILI9341)


# Define the minimum and maximum values for the dial
minimum_value = 0
maximum_value = 100

# Create a Dial widget
my_dial = Dial(
    x=20,  # set x-position of the dial inside of my_group
    y=20,  # set y-position of the dial inside of my_group
    width=180,  # requested width of the dial
    height=180,  # requested height of the dial
    padding=25,  # add 25 pixels around the dial to make room for labels
    start_angle=-120,  # left angle position at -120 degrees
```

(continues on next page)

(continued from previous page)

```
    sweep_angle=240,  # total sweep angle of 240 degrees
    min_value=minimum_value,  # set the minimum value shown on the dial
    max_value=maximum_value,  # set the maximum value shown on the dial
    tick_label_font=tick_font,  # the font used for the tick labels
    tick_label_scale=2.0,  # the scale factor for the tick label font
)

my_group = displayio.Group()
my_group.append(my_dial)

display.show(my_group)  # add high level Group to the display

step_size = 1

while True:

    # run the dial from minimum to maximum
    for this_value in range(minimum_value, maximum_value + 1, step_size):
        my_dial.value = this_value
        display.refresh()  # force the display to refresh
    time.sleep(0.5)

    # run the dial from maximum to minimum
    for this_value in range(maximum_value, minimum_value - 1, -step_size):
        my_dial.value = this_value
        display.refresh()  # force the display to refresh
    time.sleep(0.5)
```

7.2 displayio_dial

A dial gauge widget for displaying graphical information.

- Author(s): Kevin Matocha

7.2.1 Implementation Notes

Hardware:

Software and Dependencies:

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

class displayio_dial.**Dial**(*width=100*, *height=100*, *padding=12*, *sweep_angle=90*, *start_angle=None*, *clip_needle=False*, *needle_width=7*, *needle_color=8912896*, *limit_rotation=True*, *value=None*, *value_font=None*, *display_value=False*, *value_color=16711680*, *value_format_string=':0.0f'*, *min_value=0.0*, *max_value=100.0*, *anchor_point=None*, *anchored_position=None*, *tick_color=16777215*, *major_ticks=5*, *major_tick_stroke=3*, *major_tick_length=10*, *major_tick_labels=('0', '25', '50', '75', '100')*, *minor_ticks=5*, *minor_tick_stroke=1*, *minor_tick_length=5*, *tick_label_font=None*, *tick_label_color=16777215*, *rotate_tick_labels=True*, *tick_label_scale=1.0*, *background_color=None*, *value_label_anchor_point=(0.5, - 1.0)*, *value_label_anchor_on_widget=(0.5, 0.5)*, ***kwargs*)

A dial widget. The origin is set using `x` and `y`.

Parameters

- **x** (*int*) – pixel position
- **y** (*int*) – pixel position
- **width** (*int*) – requested width, in pixels
- **height** (*int*) – requested height, in pixels
- **padding** (*int*) – keep out padding amount around the border, in pixels, default is 12
- **sweep_angle** (*float*) – dial rotation, in degrees, maximum value is 360 degrees, default is 90 degrees
- **start_angle** (*float*) – starting angle, in degrees. Set to `None` for symmetry along vertical axis. Vertical is defined as 0 degrees. Negative values are counter-clockwise degrees; positive values are clockwise degrees. Defaults to `None`.
- **min_value** (*float*) – the minimum value displayed on the dial, default is 0.0
- **max_value** (*float*) – the maximum value displayed the dial, default is 100.0
- **value** (*float*) – the value to display (if None, defaults to `min_value`)
- **display_value** (*Boolean*) – set `True` to display a value label on the dial
- **value_font** (*Font*) – the font for the value label, defaults to `terminalio.FONT`
- **value_color** (*int*) – the color for the value label, defaults to 0xFF0000
- **value_format_string** (*str*) – the format string for displaying the value label (defaults to ':0.0f' to show the value rounded to the nearest whole number)
- **value_label_anchor_point** (*(float,float)*) – anchor point on the label, default value is (0.5, -1.0) where the y-value of -1.0 signifies the text baseline
- **value_label_anchor_point_on_widget** (*(float,float)*) – anchor point on the widget where the label will be placed, default value is (0.5, 0.5)
- **needle_width** (*int*) – requested pixel width of the triangular needle, default = 7
- **needle_color** (*int*) – color value for the needle, defaults to red (0xFF0000)
- **limit_rotation** (*Boolean*) – Set True to limit needle rotation to between the `min_value` and `max_value`, set to False for unlimited rotation, default is True
- **tick_color** (*int*) – tick line color (24-bit hex value), defaults to 0xFFFFFF
- **major_ticks** (*int*) – number of major ticks, default = 5
- **major_tick_stroke** (*int*) – major tick line stroke width, in pixels, default = 3

- **major_tick_length** (*int*) – major tick length, in pixels, default = 10
- **major_tick_labels** (*str*) – array of strings for the major tick labels, default is ("0", "25", "50", "75", "100")
- **tick_label_scale** (*float*) – the scaling of the tick labels, default = 1.0
- **tick_label_font** (*Font*) – font to be used for major tick labels, default is `terminalio.FONT`
- **tick_label_color** (*int*) – color for the major tick labels, default is 0xFFFFFF
- **angle_tick_labels** (*Boolean*) – set True to rotate the major tick labels to match the tick angle, default is True
- **minor_ticks** (*int*) – number of minor ticks (per major tick), default = 5
- **minor_tick_stroke** (*int*) – minor tick line stroke width, in pixels, default = 1
- **minor_tick_length** (*int*) – minor tick length, in pixels, default = 5
- **background_color** (*int*) – background color (RGB tuple or 24-bit hex value), set `None` for transparent, default is `None`
- **anchor_point** (*(float, float)*) – (X,Y) values from 0.0 to 1.0 to define the dial's anchor point relative to the dial's bounding box
- **anchored_position** (*(int, int)*) – (x,y) pixel value for the location of the *anchor_point*

Simple example of dial and moving needle

See file: `examples/displayio_layout_dial_simpletest.py`

Fig. 1: This is a diagram of a dial widget with the needle moving from its minimum to maximum positions.

Create a Group of a given size and scale. Scale is in one dimension. For example, scale=2 leads to a layer's pixel being 2x2 pixels when in the group.

resize(*new_width*, *new_height*)

Resizes the dial dimensions to the maximum size that will fit within the requested bounding box size (`new_width`, `new_height`)

Parameters

- **new_width** (*int*) – requested width, in pixels
- **new_height** (*int*) – requested height, in pixels

property value

The dial's value.

property value_font

The font used for the value's label.

property value_color

The font color used for the value's label.

property dial_center

The (x,y) pixel location of the dial's center of rotation.

property dial_radius

The length of the dial's radius, in pixels.

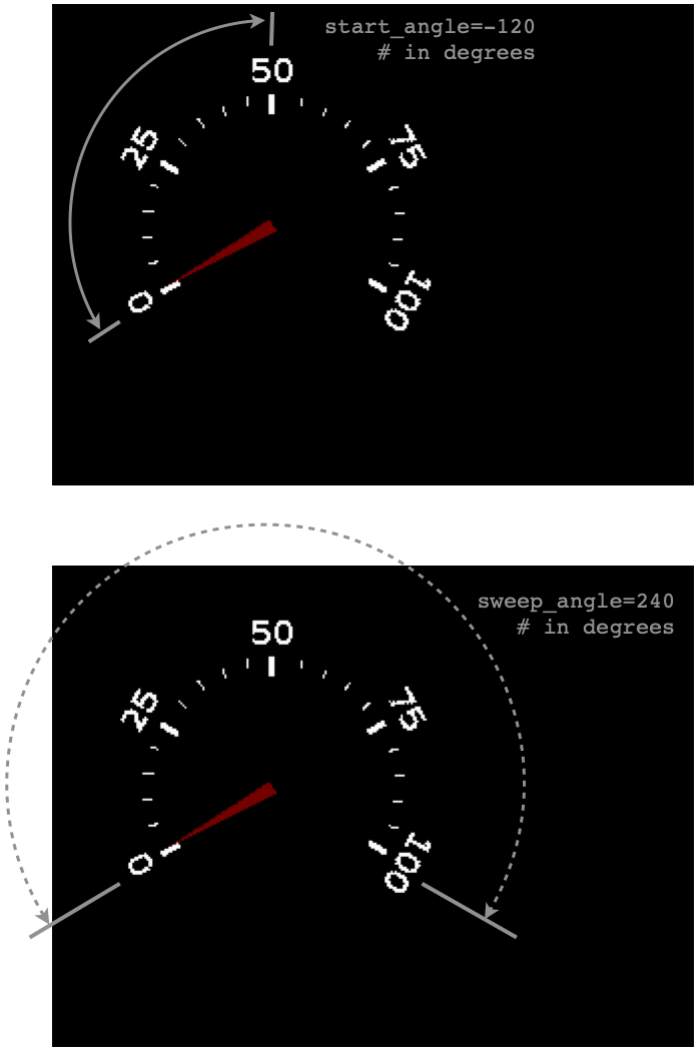


Fig. 2: Diagram showing the definition of `start_angle` and `sweep_angle`, both are in units of degrees.

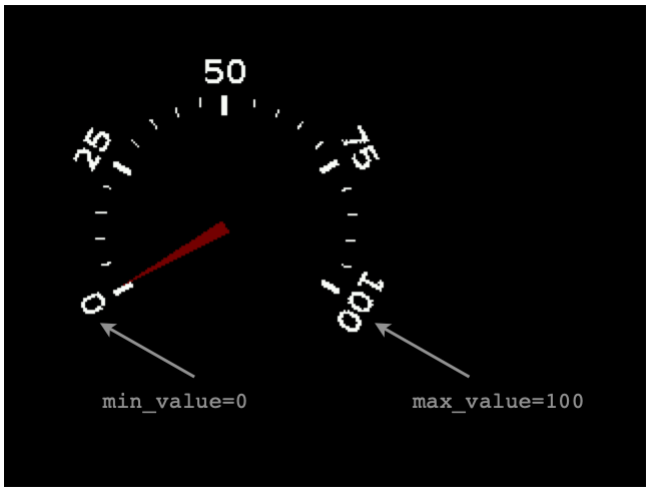


Fig. 3: Diagram showing the defintion of `min_value` and `max_value`.

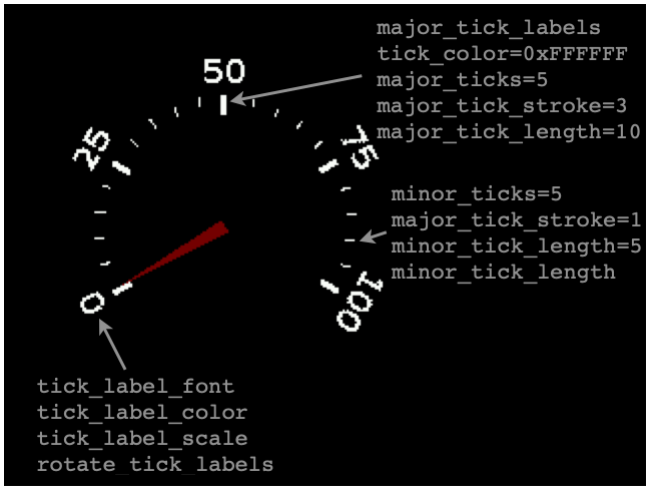


Fig. 4: Diagram showing the various parameters for setting the dial labels and major and minor tick marks.

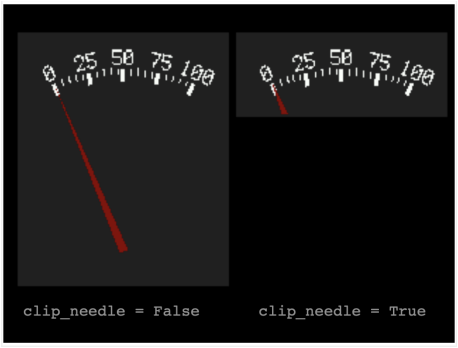


Fig. 5: Diagram showing the impact of the `clip_needle` input parameter, with the dial's boundary shown. For `sweep_angle` values less than 180 degrees, the needle can protrude a long way from the dial ticks. By setting `clip_needle = True`, the needle graphic will be clipped at the edge of the dial boundary (see comparison in the graphic above). The left dial is created with `clip_needle = False`, meaning that the dial is not clipped. The right dial is created with `clip_needle = True` and the needle is clipped at the edge of the dial. Use additional `padding` to expose more length of needle, even when clipped.

property start_angle
The starting angle of the dial, in degrees.

property sweep_angle
The sweep angle of the dial, in degrees.

property anchor_point
The anchor point for positioning the widget, works in concert with *anchored_position* The relative (X,Y) position of the widget where the anchored_position is placed. For example (0.0, 0.0) is the Widget's upper left corner, (0.5, 0.5) is the Widget's center point, and (1.0, 1.0) is the Widget's lower right corner.

Parameters **anchor_point** (*Tuple[float, float]*) – In relative units of the Widget size.

property anchored_position
The anchored position (in pixels) for positioning the widget, works in concert with *anchor_point*. The *anchored_position* is the x,y pixel position for the placement of the Widget's *anchor_point*.

Parameters **anchored_position** (*Tuple[int, int]*) – The (x,y) pixel position for the anchored_position (in pixels).

append(*layer*)
Append a layer to the group. It will be drawn above other layers.

property bounding_box
The boundary of the widget. [x, y, width, height] in Widget's local coordinates (in pixels). (getter only)

Returns Tuple[int, int, int, int]

property height
The widget height, in pixels. (getter only)

Returns int

property hidden
True when the Group and all of it's layers are not visible. When False, the Group's layers are visible if they haven't been hidden.

index(*layer*)
Returns the index of the first copy of layer. Raises ValueError if not found.

insert(*index*, *layer*)
Insert a layer into the group.

pop(*index=- 1*)
Remove the ith item and return it.

remove(*layer*)
Remove the first copy of layer. Raises ValueError if it is not present.

property scale
Scales each pixel within the Group in both directions. For example, when scale=2 each pixel will be represented by 2x2 pixels.

update_transform(*parent_transform*)
Update the parent transform and child transforms

property width
The widget width, in pixels. (getter only)

Returns int

property x
X position of the Group in the parent.

property y
Y position of the Group in the parent.

displayio_dial.**draw_ticks**(*target_bitmap*, ***, *dial_center*, *dial_radius*, *tick_count*, *tick_stroke*, *tick_length*, *start_angle*, *sweep_angle*, *tick_color_index=2*)
Helper function for drawing ticks on the dial widget. Can be used to customize the dial face.

Parameters

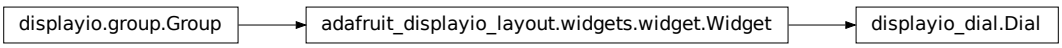
- **target_bitmap** (*displayio.Bitmap*) – Bitmap where ticks will be drawn into
- **dial_center** ((*int*,*int*)) – the (x,y) pixel location in the bitmap of the dial's center of rotation
- **dial_radius** (*int*) – the radius of the dial (not including padding), in pixels
- **tick_count** (*int*) – number of ticks to be drawn
- **tick_stroke** (*int*) – the pixel width of the line used to draw the tick
- **start_angle** (*float*) – starting angle of the dial, in degrees
- **sweep_angle** (*float*) – total sweep angle of the dial, in degrees
- **tick_color_index** (*int*) – the bitmap's color index that should be used for drawing the tick marks

displayio_dial.**draw_labels**(*target_bitmap*, ***, *font*, *font_height*, *tick_labels*, *dial_center*, *dial_radius*, *start_angle*, *sweep_angle*, *rotate_labels=True*, *tick_label_scale=1.0*)
Helper function for drawing text labels on the dial widget. Can be used to customize the dial face.

Parameters

- **target_bitmap** (*displayio.Bitmap*) – Bitmap where ticks will be drawn into
- **font** (*Font*) – the font to be used to draw the tick mark text labels
- **font_height** (*int*) – the height of the font, used for text placement
- **tick_labels** (*List[str]*) – a list of strings for the tick text labels

- **dial_center** ((*int*, *int*)) – the (x,y) pixel location in the bitmap of the dial's center of rotation
- **dial_radius** (*int*) – the radius of the dial (not including padding), in pixels
- **tick_count** (*int*) – number of ticks to be drawn
- **tick_stroke** (*int*) – the pixel width of the line used to draw the tick
- **start_angle** (*float*) – starting angle of the dial, in degrees
- **sweep_angle** (*float*) – total sweep angle of the dial, in degrees
- **rotate_labels** (*bool*) – set to True if you want the label text to be rotated to align with the tick marks
- **tick_label_scale** (*float*) – scale factor for the tick text labels, default is 1.0



CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

displayio_dial, 16

INDEX

A

anchor_point (*displayio_dial.Dial property*), 21
anchored_position (*displayio_dial.Dial property*), 21
append() (*displayio_dial.Dial method*), 21

B

bounding_box (*displayio_dial.Dial property*), 21

D

Dial (*class in displayio_dial*), 16
dial_center (*displayio_dial.Dial property*), 18
dial_radius (*displayio_dial.Dial property*), 18
displayio_dial
    module, 16
draw_labels() (*in module displayio_dial*), 22
draw_ticks() (*in module displayio_dial*), 22

H

height (*displayio_dial.Dial property*), 21
hidden (*displayio_dial.Dial property*), 21

I

index() (*displayio_dial.Dial method*), 21
insert() (*displayio_dial.Dial method*), 21

M

module
    displayio_dial, 16

P

pop() (*displayio_dial.Dial method*), 22

R

remove() (*displayio_dial.Dial method*), 22
resize() (*displayio_dial.Dial method*), 18

S

scale (*displayio_dial.Dial property*), 22
start_angle (*displayio_dial.Dial property*), 18
sweep_angle (*displayio_dial.Dial property*), 21

U

update_transform() (*displayio_dial.Dial method*), 22

V

value (*displayio_dial.Dial property*), 18
value_color (*displayio_dial.Dial property*), 18
value_font (*displayio_dial.Dial property*), 18

W

width (*displayio_dial.Dial property*), 22

X

x (*displayio_dial.Dial property*), 22

Y

y (*displayio_dial.Dial property*), 22