Adafruitfeatherwing Library Documentation

Release 1.0

Scott Shawcroft

Feb 23, 2019

Contents

This library provides FeatherWing specific classes for those that require a significant amount of initialization.

CHAPTER 1

Dependencies

These drivers depends on:

- Adafruit CircuitPython
- INA219
- Seesaw
- HT16K33
- DotStar
- NeoPixel
- DS3231

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle and highly recommended over installing each one.

1.1 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-featherwing
```

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

```
sudo pip3 install adafruit-circuitpython-featherwing
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
```

(continues on next page)

(continued from previous page)

```
source .env/bin/activate
pip3 install adafruit-circuitpython-featherwing
```

CHAPTER 2

Contributing

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

CHAPTER 3

Building locally

To build this library locally you'll need to install the circuitpython-build-tools package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-featherwing --
↪library_location .
```

3.1 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the index.html in your browser to view them. It will also (due to -W) error out on any warning like Travis will. This is a good way to locally verify it will pass.

CHAPTER 4

Table of Contents

4.1 Simple tests

Ensure your device works with this simple test.

Listing 1: examples/featherwing_ina219_simpletest.py

```
""" Example to print out the voltage and current using the INA219 """
import time
from adafruit_featherwing import ina219_featherwing

INA219 = ina219_featherwing.INA219FeatherWing()

while True:
    print("Bus Voltage:   {} V".format(INA219.bus_voltage))
    print("Shunt Voltage: {} V".format(INA219.shunt_voltage))
    print("Voltage:       {} V".format(INA219.voltage))
    print("Current:       {} mA".format(INA219.current))
    print("")
    time.sleep(0.5)
```

Listing 2: examples/featherwing_joy_simpletest.py

```
"""This example zeros the joystick, and prints when the joystick moves
   or the buttons are pressed."""
import time
from adafruit_featherwing import joy_featherwing

wing = joy_featherwing.JoyFeatherWing()
last_x = 0
last_y = 0

while True:
    x, y = wing.joystick
```

(continues on next page)

(continued from previous page)

```
    if (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
        last_x = x
        last_y = y
        print(x, y)
    if wing.button_a:
        print("Button A!")
    if wing.button_b:
        print("Button B!")
    if wing.button_x:
        print("Button X!")
    if wing.button_y:
        print("Button Y!")
    if wing.button_select:
        print("Button SELECT!")
    time.sleep(.01)
```

Listing 3: examples/featherwing_alphanum_simpletest.py

```
"""This example changes the fill, brightness, blink rates,
shows number and text printing, displays a counter
and then shows off the new marquee features."""

from time import sleep
from adafruit_featherwing import alphanum_featherwing

display = alphanum_featherwing.AlphaNumFeatherWing()

#Fill and empty all segments
for count in range(0, 3):
    display.fill(True)
    sleep(0.5)
    display.fill(False)
    sleep(0.5)

#Display a number and text
display.print(1234)
sleep(1)
display.print('Text')

#Change brightness
for brightness in range(0, 16):
    display.brightness = brightness
    sleep(0.1)

#Change blink rate
for blink_rate in range(3, 0, -1):
    display.blink_rate = blink_rate
    sleep(4)
display.blink_rate = 0

#Show a counter using decimals
count = 975.0
while count < 1025:
    count += 1
    display.print(count)
    sleep(0.1)
```

(continues on next page)

(continued from previous page)

```

#Show the Marquee
display.marquee('This is a really long message!!!  ', 0.2)
```

Listing 4: examples/featherwing_dotstar_simpletest.py

```
"""
This plays various animations
and then draws random pixels at random locations
"""

from time import sleep
import random
from adafruit_featherwing import dotstar_featherwing

dotstar = dotstar_featherwing.DotStarFeatherWing()

# HELPERS
# a random color 0 -> 224
def random_color():
    return random.randrange(0, 8) * 32

# Fill screen with random colors at random brightnesses
for i in range(0, 5):
    dotstar.fill((random_color(), random_color(), random_color()))
    dotstar.brightness = random.randrange(2, 10) / 10
    sleep(.2)

# Set display to 30% brightness
dotstar.brightness = 0.3

# Create a gradiant drawing each pixel
for x in range(0, dotstar.columns):
    for y in range(dotstar.rows - 1, -1, -1):
        dotstar[x, y] = (y * 42, 255, y * 42, 1)

#Rotate everything left 36 frames
for i in range(0, 36):
    dotstar.shift_down(True)

# Draw dual gradiant and then update
dotstar.auto_write = False
for y in range(0, dotstar.rows):
    for x in range(0, 6):
        dotstar[x, y] = (y * 84, x * 42, x * 42, 1)
    for x in range(6, 12):
        dotstar[x, y] = (255 - (y * 84), 255 - ((x - 6) * 42), 255 - ((x - 6) * 42),␣
↪1)

# Rotate everything left 36 frames
for i in range(0, 36):
    dotstar.shift_left(True)
    dotstar.shift_up(True)
    dotstar.show()
dotstar.auto_write = True

```

(continues on next page)

(continued from previous page)

```
# Shift pixels without rotating for an animated screen wipe
for i in range(0, 6):
    dotstar.shift_down()

# Show pixels in random locations of random color
# Bottom left corner is (0,0)
while True:
    x = random.randrange(0, dotstar.columns)
    y = random.randrange(0, dotstar.rows)
    dotstar[x, y] = (random_color(), random_color(), random_color())
    sleep(.1)
```

Listing 5: examples/featherwing_neopixel_simpletest.py

```
"""
This example plays various animations
and then draws random pixels at random locations
"""

from time import sleep
import random
from adafruit_featherwing import neopixel_featherwing

neopixel = neopixel_featherwing.NeoPixelFeatherWing()

# HELPERS
# a random color 0 -> 224
def random_color():
    return random.randrange(0, 8) * 32

# Fill screen with random colors at random brightnesses
for i in range(0, 5):
    neopixel.fill((random_color(), random_color(), random_color()))
    neopixel.brightness = random.randrange(2, 10) / 10
    sleep(.2)

# Set display to 30% brightness
neopixel.brightness = 0.3

# Create a gradiant drawing each pixel
for x in range(0, neopixel.columns):
    for y in range(neopixel.rows - 1, -1, -1):
        neopixel[x, y] = (y * 63, 255, y * 63)

#Rotate everything left 36 frames
for i in range(0, 36):
    neopixel.shift_down(True)
    sleep(0.1)

# Draw dual gradiant and then update
#neopixel.auto_write = False
for y in range(0, neopixel.rows):
    for x in range(0, 4):
        neopixel[x, y] = (y * 16 + 32, x * 8, 0)
    for x in range(4, 8):
        neopixel[x, y] = ((4 - y) * 16 + 32, (8 - x) * 8, 0)
```

(continues on next page)

(continued from previous page)

```
neopixel.show()

# Rotate everything left 36 frames
for i in range(0, 36):
    neopixel.shift_left(True)
    neopixel.shift_up(True)
    neopixel.show()
    sleep(0.1)
neopixel.auto_write = True

# Shift pixels without rotating for an animated screen wipe
for i in range(0, neopixel.rows):
    neopixel.shift_down()
    sleep(0.4)

# Show pixels in random locations of random color
# Bottom left corner is (0,0)
while True:
    x = random.randrange(0, neopixel.columns)
    y = random.randrange(0, neopixel.rows)
    neopixel[x, y] = (random_color(), random_color(), random_color())
    sleep(.1)
```

Listing 6: examples/featherwing_sevensegment_simpletest.py

```
"""This example changes the fill, brightness, blink rates,
shows number and text printing, displays a counter
and then shows off the new marquee features."""

from time import sleep
from adafruit_featherwing import sevensegment_featherwing

display = sevensegment_featherwing.SevenSegmentFeatherWing()

#Fill and empty all segments
for count in range(0, 3):
    display.fill(True)
    sleep(0.5)
    display.fill(False)
    sleep(0.5)

#Display a number and text
display.print(1234)
sleep(1)
display.print('FEED')

#Change brightness
for brightness in range(0, 16):
    display.brightness = brightness
    sleep(0.1)

#Change blink rate
for blink_rate in range(3, 0, -1):
    display.blink_rate = blink_rate
    sleep(4)
display.blink_rate = 0
```

(continues on next page)

(continued from previous page)

```

#Show a counter using decimals
count = 975.0
while count < 1025:
    count += 1
    display.print(count)
    sleep(0.1)

#Display a Time
hour = 12
for minute in range(15, 26):
    display.print("{}:{}".format(hour, minute))
    sleep(1)

#Show the Marquee
display.marquee('Deadbeef 192.168.100.102... ', 0.2)
```

Listing 7: examples/featherwing_rtc_simpletest.py

```
"""
This example will allow you to set the date and time
and then loop through and display the current time
"""
import time
from adafruit_featherwing import rtc_featherwing

days = ("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")

# Create the RTC instance:
rtc = rtc_featherwing.RTCFeatherWing()

#pylint: disable-msg=using-constant-test
if True:   # Change this to True to set the date and time
    rtc.set_time(13, 34)        # Set the time (seconds are optional)
    print(rtc.now)
    rtc.set_date(16, 1, 2016)   # Set the date
    print(rtc.now)
    rtc.year = 2019             # Set just the Year
    print(rtc.now)
    rtc.month = 2               # Set Just the Month
    print(rtc.now)
    rtc.hour = 16               # Set just the hour
    print(rtc.now)
    rtc.weekday = 6             # Set just the day of the week (Sunday = 0)
    print(rtc.now)
    rtc.unixtime = 1550335257   # Or set the date and time with a unix timestamp

# Main loop:
while True:
    now = rtc.now
    print("The date is {} {}/{}/{}".format(days[now.weekday], now.day, now.month, now.
↪year))
    print("The time is {}:{:02}:{:02}".format(now.hour, now.minute, now.second))
    print("The UNIX timestamp is {}".format(rtc.unixtime))
    print("The number of days in the current month is {}".format(rtc.get_month_
↪days()))
```

(continues on next page)

(continued from previous page)

```
    if rtc.is_leap_year():
        print("This year is a leap year")
    else:
        print("This year is not a leap year")
    time.sleep(1) # wait a second
```

4.2 Other Tests

Listing 8: examples/featherwing_dotstar_palette_example.py

```
"""
This creates a palette of colors, draws a pattern and
rotates through the palette creating a moving rainbow.
"""

from math import sqrt, cos, sin, radians
from adafruit_featherwing import dotstar_featherwing

dotstar = dotstar_featherwing.DotStarFeatherWing()

# Remap the calculated rotation to 0 - 255
def remap(vector):
    return int(((255 * vector + 85) * 0.75) + 0.5)

# Calculate the Hue rotation starting with Red as 0 degrees
def rotate(degrees):
    cosA = cos(radians(degrees))
    sinA = sin(radians(degrees))
    red = cosA + (1.0 - cosA) / 3.0
    green = 1./3. * (1.0 - cosA) + sqrt(1./3.) * sinA
    blue = 1./3. * (1.0 - cosA) - sqrt(1./3.) * sinA
    return (remap(red), remap(green), remap(blue))

palette = []
pixels = []

# Generate a rainbow palette
for degree in range(0, 360):
    color = rotate(degree)
    palette.append(color[0] << 16 | color[1] << 8 | color[2])

# Create the Pattern
for y in range(0, dotstar.rows):
    for x in range(0, dotstar.columns):
        pixels.append(x * 30 + y * -30)

# Clear the screen
dotstar.fill()

# Start the Animation
dotstar.auto_write = False
while True:
    for color in range(0, 360, 10):
        for index in range(0, dotstar.rows * dotstar.columns):
```

(continues on next page)

(continued from previous page)

```
            palette_index = pixels[index] + color
            if palette_index >= 360:
                palette_index -= 360
            elif palette_index < 0:
                palette_index += 360
            dotstar[index] = palette[palette_index]
        dotstar.show()
```

Listing 9: examples/featherwing_neopixel_palette_example.py

```
"""
This creates a palette of colors, draws a pattern and
rotates through the palette creating a moving rainbow.
"""

from math import sqrt, cos, sin, radians
from adafruit_featherwing import neopixel_featherwing

neopixel = neopixel_featherwing.NeoPixelFeatherWing()

# Remap the calculated rotation to 0 - 255
def remap(vector):
    return int(((255 * vector + 85) * 0.75) + 0.5)

# Calculate the Hue rotation starting with Red as 0 degrees
def rotate(degrees):
    cosA = cos(radians(degrees))
    sinA = sin(radians(degrees))
    red = cosA + (1.0 - cosA) / 3.0
    green = 1./3. * (1.0 - cosA) + sqrt(1./3.) * sinA
    blue = 1./3. * (1.0 - cosA) - sqrt(1./3.) * sinA
    return (remap(red), remap(green), remap(blue))

palette = []
pixels = []

# Generate a rainbow palette
for degree in range(0, 360):
    color = rotate(degree)
    palette.append(color[0] << 16 | color[1] << 8 | color[2])

# Create the Pattern
for y in range(0, neopixel.rows):
    for x in range(0, neopixel.columns):
        pixels.append(x * 30 + y * -30)

# Clear the screen
neopixel.fill()

# Start the Animation
neopixel.auto_write = False
while True:
    for color in range(0, 360, 10):
        for index in range(0, neopixel.rows * neopixel.columns):
            palette_index = pixels[index] + color
            if palette_index >= 360:
```

(continues on next page)

(continued from previous page)

```
                palette_index -= 360
            elif palette_index < 0:
                palette_index += 360
            neopixel[index] = palette[palette_index]
        neopixel.show()
```

4.3 adafruit_featherwing.ina219_featherwing

Helper for using the INA219 FeatherWing.

- Author(s): Kattni Rembor

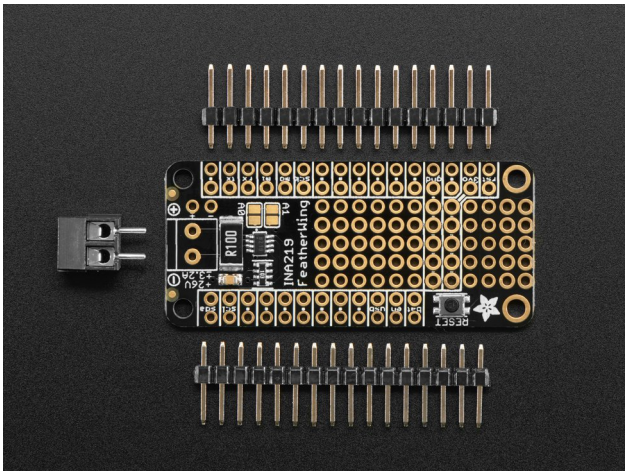
class adafruit_featherwing.ina219_featherwing.**INA219FeatherWing**

Class representing an Adafruit INA219 FeatherWing.

Automatically uses the feather's I2C bus.

bus_voltage

Bus voltage returns volts.



This example prints the bus voltage with the appropriate units.

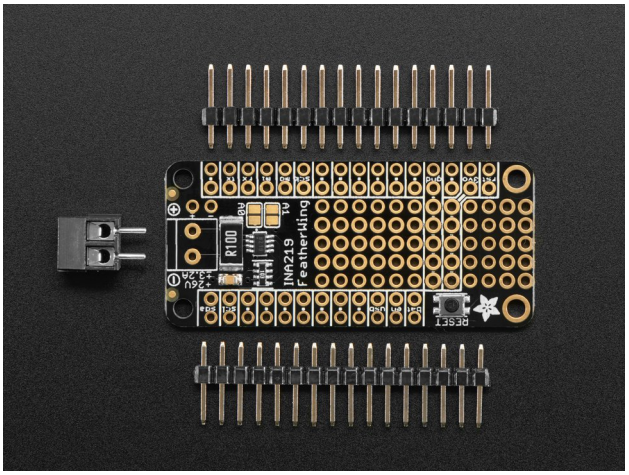
```
from adafruit_featherwing import ina219_featherwing
import time

ina219 = ina219_featherwing.INA219FeatherWing()

while True:
    print("Bus Voltage: {} V".format(ina219.bus_voltage))
    time.sleep(0.5)
```

current

Current returns mA.



This example prints the current with the appropriate units.

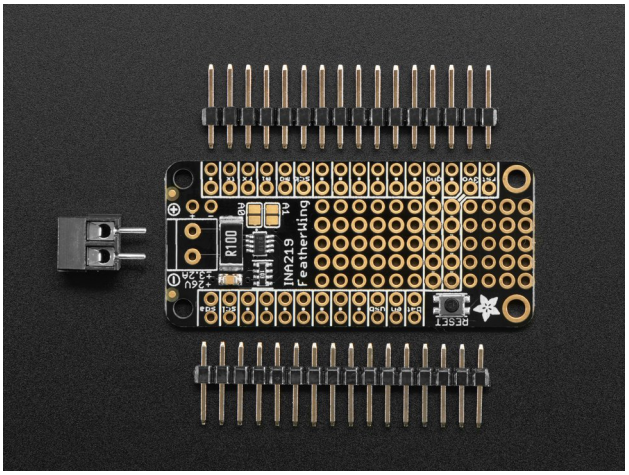
```
from adafruit_featherwing import ina219_featherwing
import time

ina219 = ina219_featherwing.INA219FeatherWing()

while True:
    print("Current: {} mA".format(ina219.current))
    time.sleep(0.5)
```

shunt_voltage

Shunt voltage returns volts.



This example prints the shunt voltage with the appropriate units.

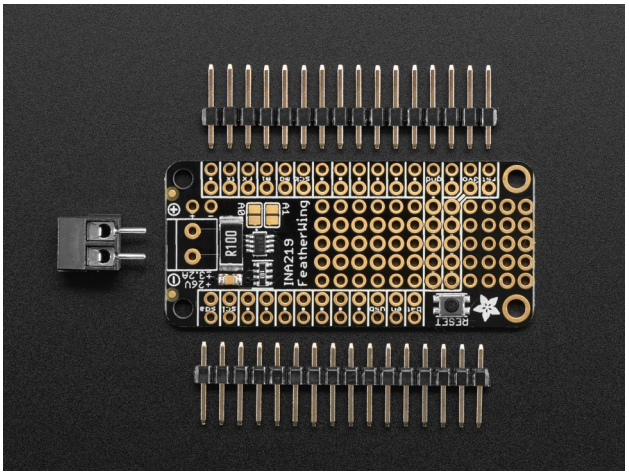
```
from adafruit_featherwing import ina219_featherwing
import time

ina219 = ina219_featherwing.INA219FeatherWing()

while True:
    print("Shunt Voltage: {} V".format(ina219.shunt_voltage))
    time.sleep(0.5)
```

voltage

Voltage, known as load voltage, is bus voltage plus shunt voltage. Returns volts.



This example prints the voltage with the appropriate units.

```
from adafruit_featherwing import ina219_featherwing
import time

ina219 = ina219_featherwing.INA219FeatherWing()

while True:
    print("Voltage: {} V".format(ina219.voltage))
    time.sleep(0.5)
```

4.4 adafruit_featherwing.joy_featherwing

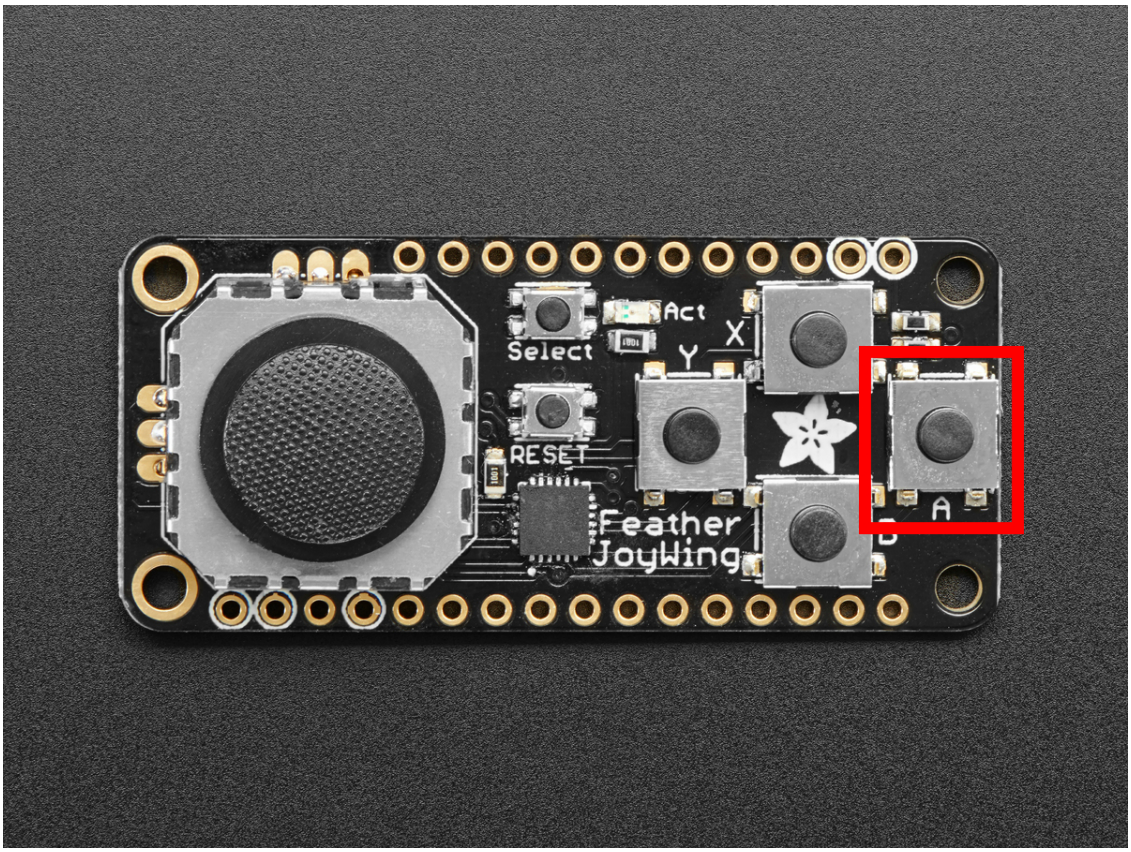
Helper for using the Joy FeatherWing.

- Author(s): Kattni Rembor

class adafruit_featherwing.joy_featherwing.**JoyFeatherWing**
Class representing an Adafruit Joy FeatherWing.

Automatically uses the feather's I2C bus.

button_a
Joy featherwing button A.



This example prints when button A is pressed.

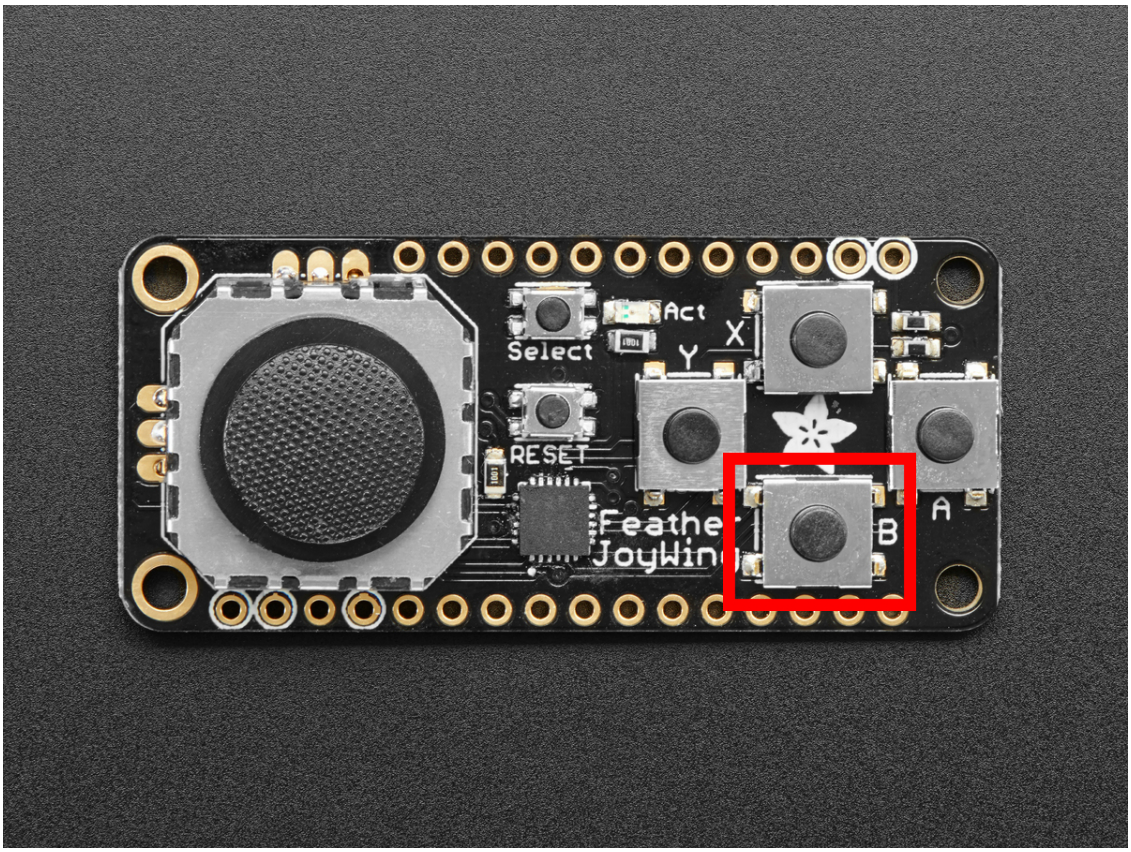
```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()

while True:
    if wing.button_a:
    print("Button A pressed!")
```

button_b

Joy featherwing button B.



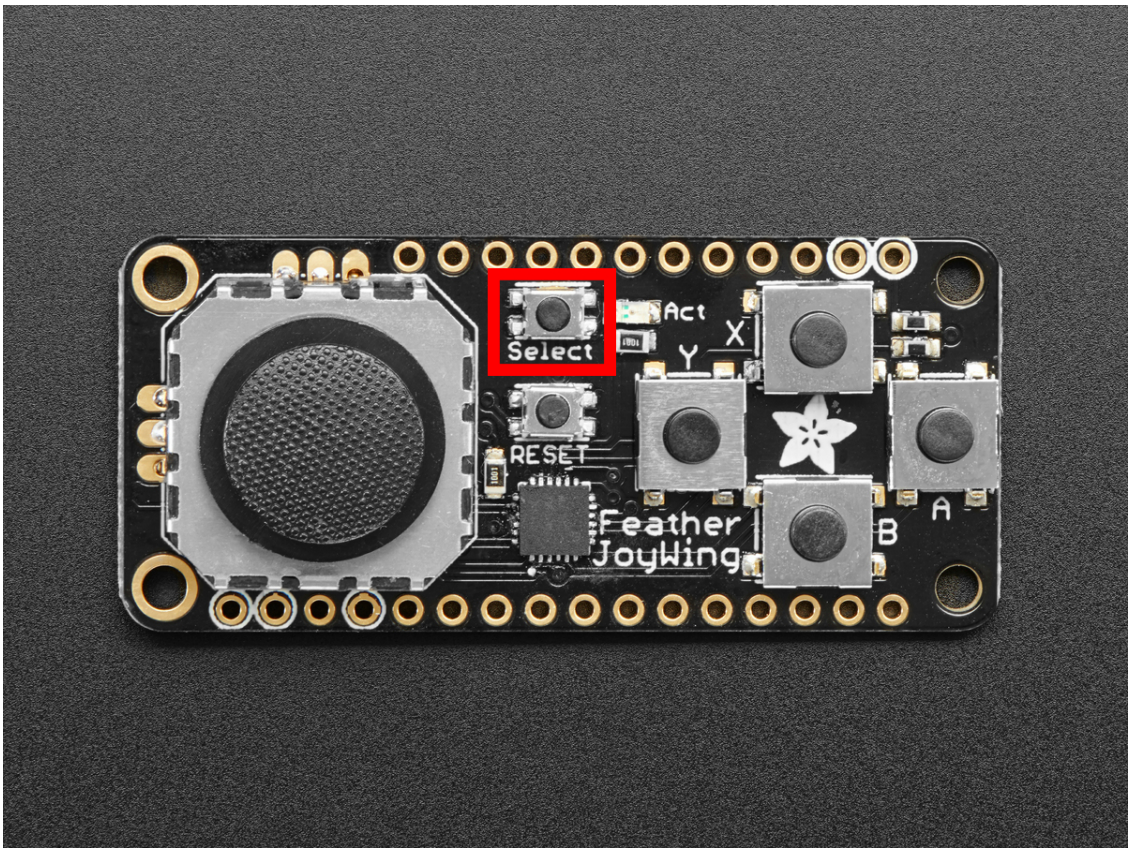
This example prints when button B is pressed.

```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()

while True:
    if wing.button_b:
    print("Button B pressed!")
```

button_select
Joy featherwing button SELECT.



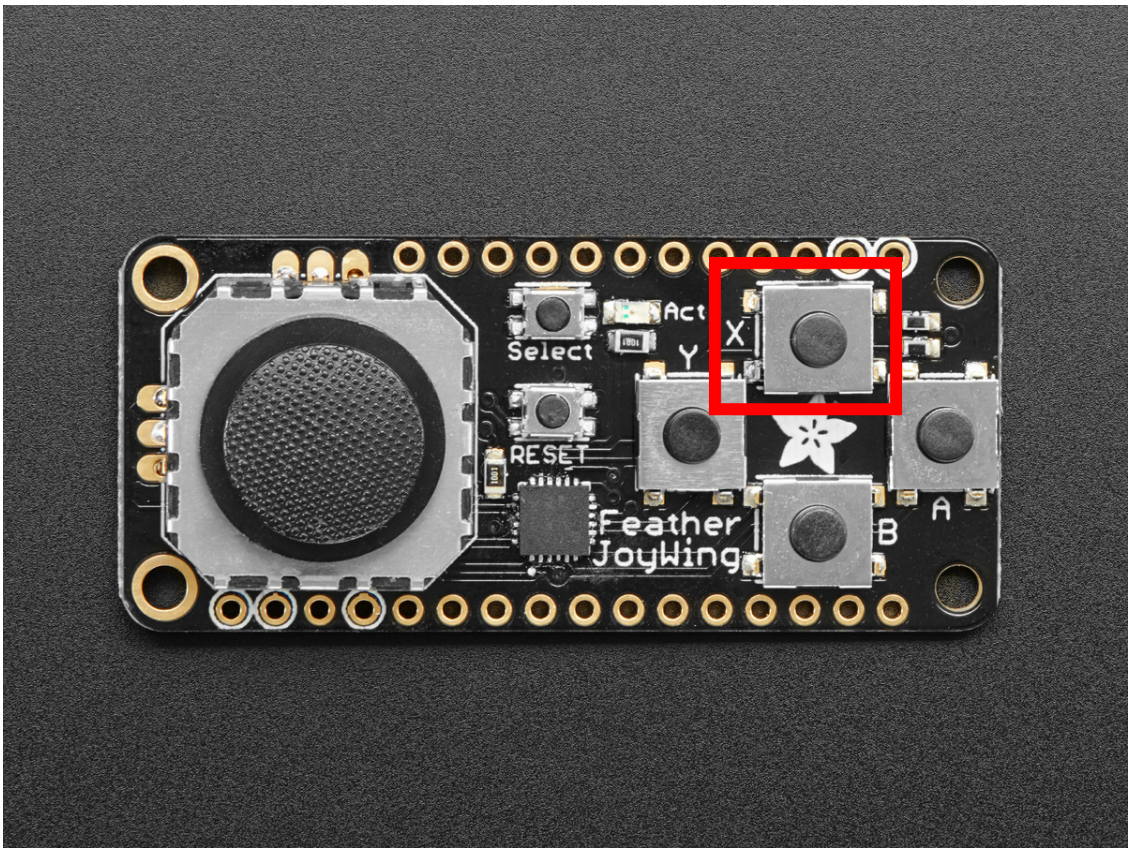
This example prints when button SELECT is pressed.

```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()

while True:
    if wing.button_select:
    print("Button SELECT pressed!")
```

button_x
Joy featherwing button X.



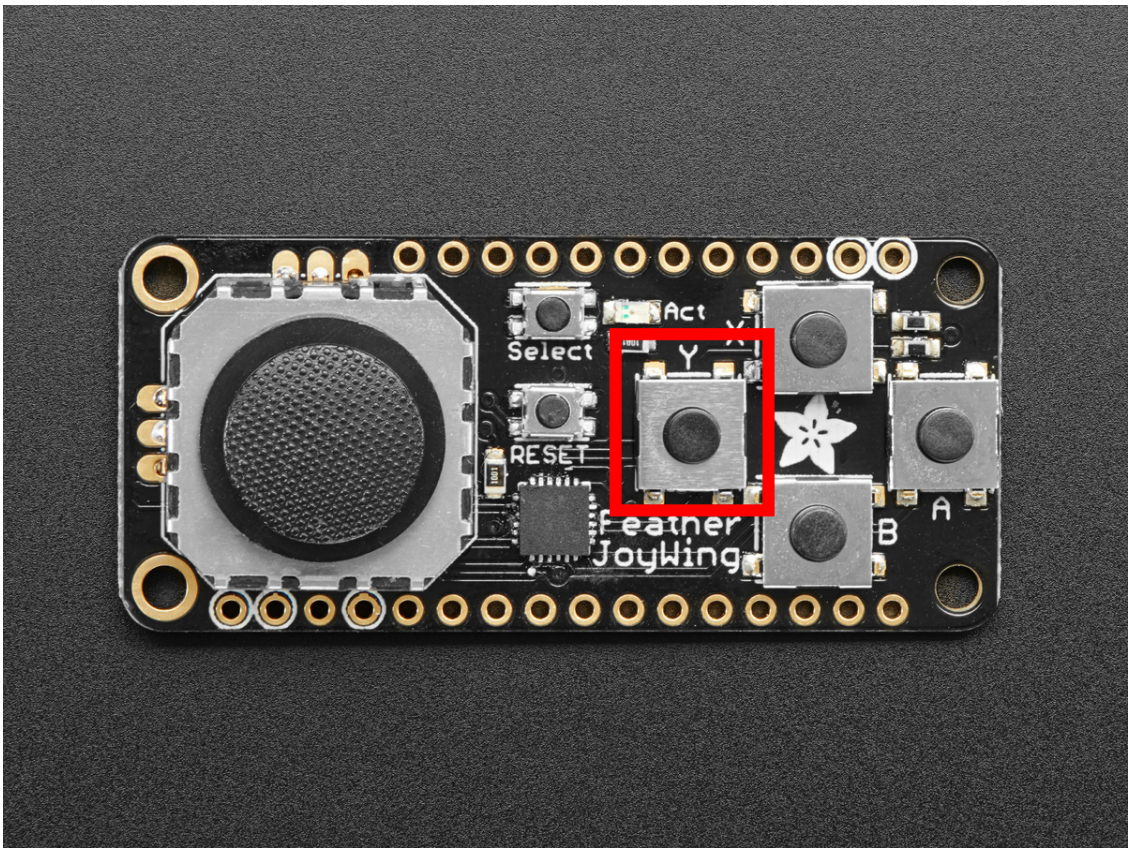
This example prints when button X is pressed.

```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()

while True:
    if wing.button_x:
    print("Button X pressed!")
```

button_y
Joy featherwing button Y.



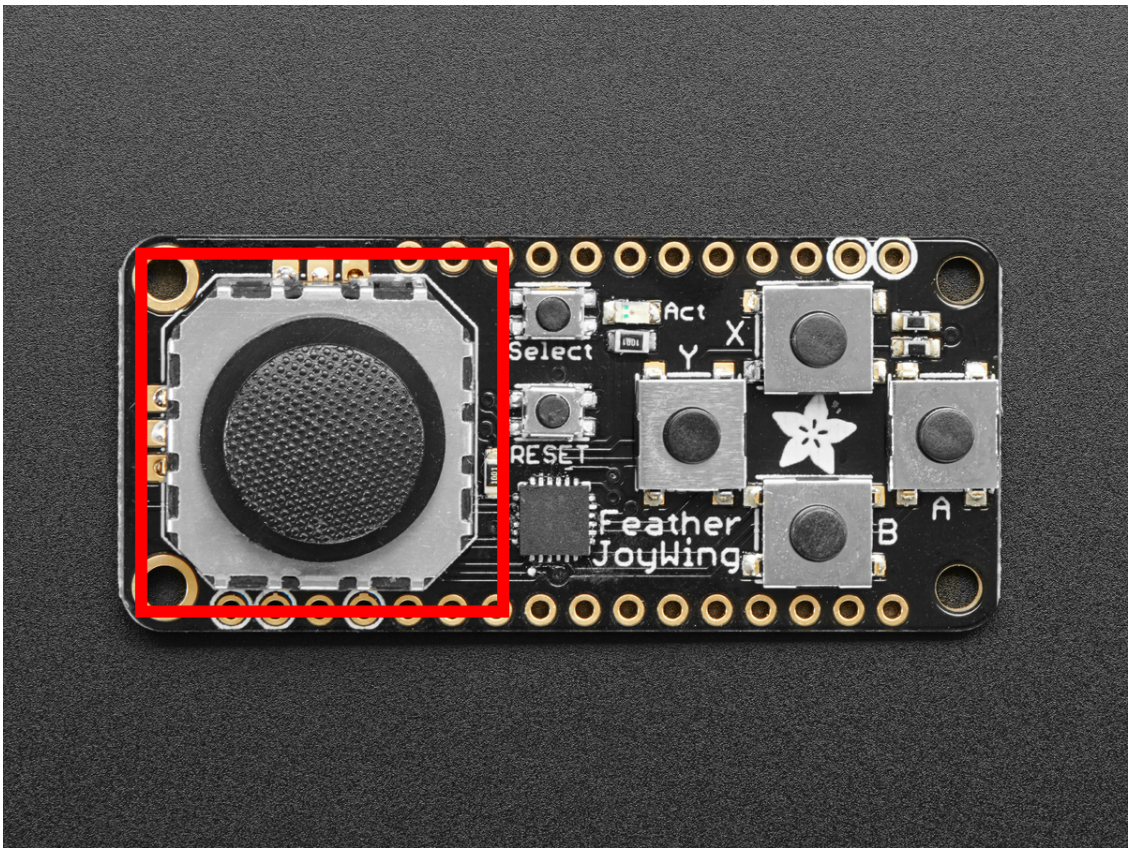
This example prints when button Y is pressed.

```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()

while True:
    if wing.button_y:
    print("Button Y pressed!")
```

joystick
Joy FeatherWing joystick.



This example zeros the joystick, and prints the coordinates of joystick when it is moved.

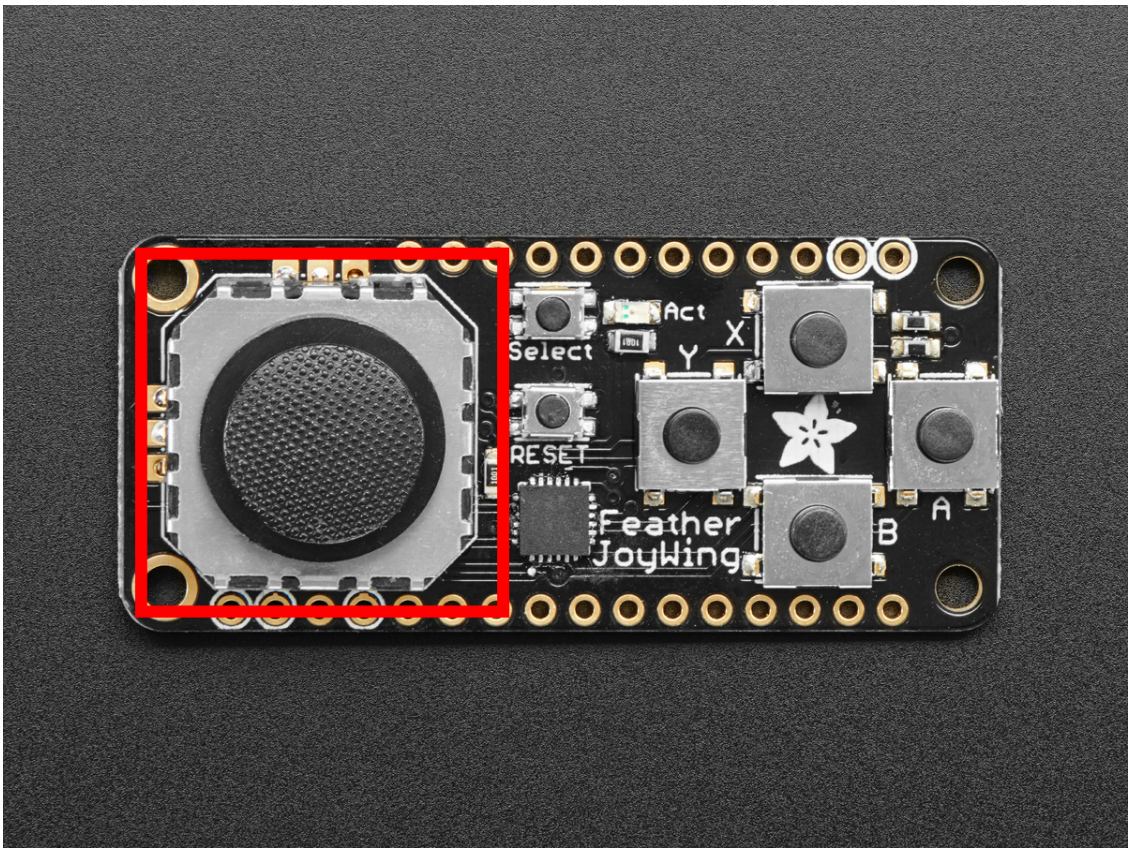
```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()
last_x = 0
last_y = 0
wing.zero_joystick()

while True:
    x, y = wing.joystick
    if (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
        last_x = x
        last_y = y
        print(x, y)
    time.sleep(0.01)
```

joystick_offset

Offset used to correctly report (0, 0) when the joystick is centered.



Provide a tuple of (x, y) to set your joystick center to (0, 0). The offset you provide is subtracted from the current reading. For example, if your joystick reads as (-4, 0), you would enter (-4, 0) as the offset. The code will subtract -4 from -4, and 0 from 0, returning (0, 0).

This example supplies an offset for zeroing, and prints the coordinates of the joystick when it is moved.

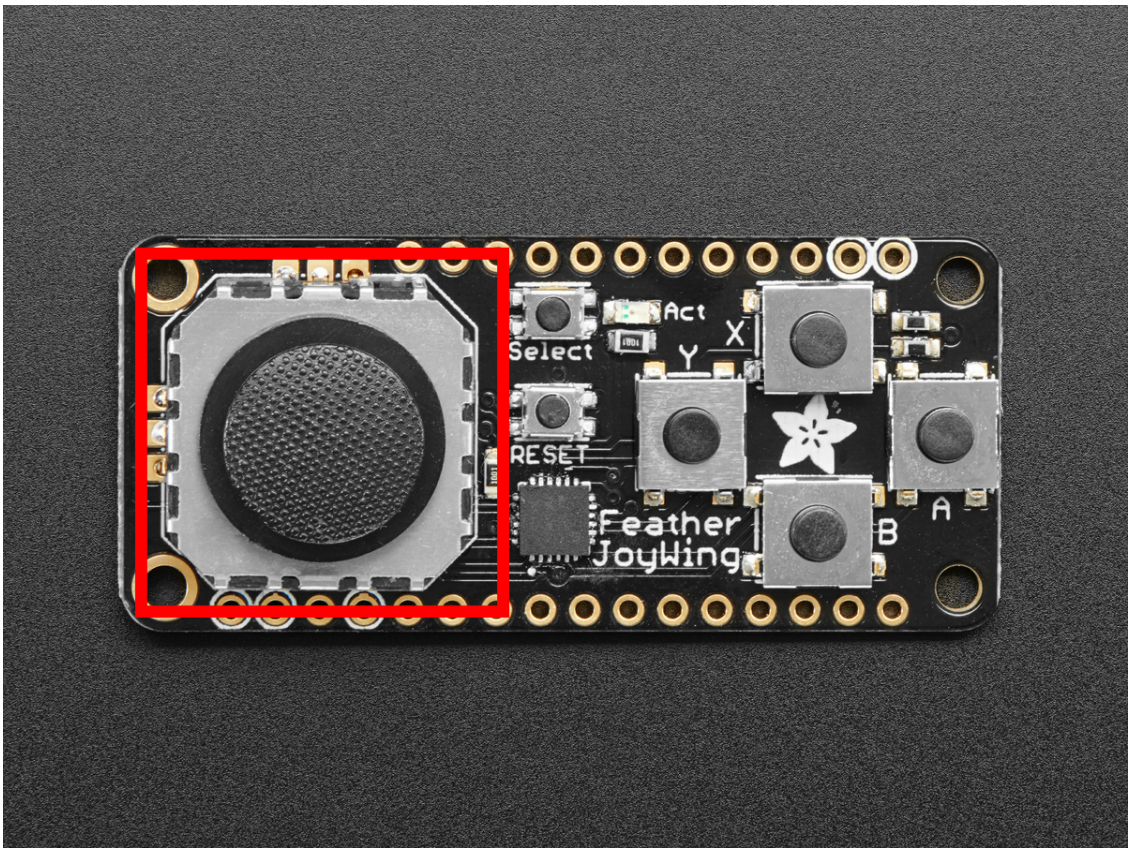
```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()
last_x = 0
last_y = 0

while True:
    wing.joystick_offset = (-4, 0)
    x, y = wing.joystick
    if (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
        last_x = x
        last_y = y
        print(x, y)
    time.sleep(0.01)
```

zero_joystick()

Zeros the joystick by using current reading as (0, 0). Note: You must not be touching the joystick at the time of zeroing for it to be accurate.



This example zeros the joystick, and prints the coordinates of joystick when it is moved.

```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()
last_x = 0
last_y = 0
wing.zero_joystick()

while True:
    x, y = wing.joystick
    if (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
        last_x = x
        last_y = y
        print(x, y)
    time.sleep(0.01)
```

4.5 adafruit_featherwing.alphanum_featherwing

Helper for using the 14-Segment AlphaNumeric FeatherWing.

- Author(s): Melissa LeBlanc-Williams

class adafruit_featherwing.alphanum_featherwing.**AlphaNumFeatherWing**(*address=112*)
Class representing an Adafruit 14-segment AlphaNumeric FeatherWing.

Automatically uses the feather's I2C bus.

4.6 adafruit_featherwing.dotstar_featherwing

Helper for using the DotStar FeatherWing.

- Author(s): Melissa LeBlanc-Williams

class adafruit_featherwing.dotstar_featherwing.**DotStarFeatherWing**(*clock=<sphinx.ext.autodoc.importer object>*, *data=<sphinx.ext.autodoc.importer object>*, *brightness=0.2*)

Class representing a DotStar FeatherWing.

The feather uses pins D13 and D11

4.7 adafruit_featherwing.neopixel_featherwing

Helper for using the NeoPixel FeatherWing.

- Author(s): Melissa LeBlanc-Williams

class adafruit_featherwing.neopixel_featherwing.**NeoPixelFeatherWing**(*pixel_pin=<sphinx.ext.autodoc.importer object>*, *brightness=0.1*)

Class representing a NeoPixel FeatherWing.

The feather uses pins D6 by default

shift_down(*rotate=False*)

Shift all pixels down.

Parameters **rotate** – (Optional) Rotate the shifted pixels to top (default=False)

This example shifts 2 pixels down

```
import time
from adafruit_featherwing import neopixel_featherwing

neopixel = neopixel_featherwing.NeoPixelFeatherWing()

# Draw Red and Green Pixels
neopixel[4, 1] = (255, 0, 0)
neopixel[5, 1] = (0, 255, 0)

# Rotate it off the screen
for i in range(0, neopixel.rows - 1):
    neopixel.shift_down(True)
    time.sleep(.1)

time.sleep(1)
# Shift it off the screen
for i in range(0, neopixel.rows - 1):
    neopixel.shift_down()
    time.sleep(.1)
```

shift_up(*rotate=False*)
Shift all pixels up

Parameters **rotate** – (Optional) Rotate the shifted pixels to bottom (default=False)

This example shifts 2 pixels up

```
import time
from adafruit_featherwing import neopixel_featherwing

neopixel = neopixel_featherwing.NeoPixelFeatherWing()

# Draw Red and Green Pixels
neopixel[4, 1] = (255, 0, 0)
neopixel[5, 1] = (0, 255, 0)

# Rotate it off the screen
for i in range(0, neopixel.rows - 1):
    neopixel.shift_up(True)
    time.sleep(.1)

time.sleep(1)
# Shift it off the screen
for i in range(0, neopixel.rows - 1):
    neopixel.shift_up()
    time.sleep(.1)
```

4.8 adafruit_featherwing.rtc_featherwing

Helper for using the DS3231 Precision RTC FeatherWing.

- Author(s): Melissa LeBlanc-Williams

class adafruit_featherwing.rtc_featherwing.**RTCFeatherWing**
Class representing an DS3231 Precision RTC FeatherWing.

Automatically uses the feather's I2C bus.

datetime
Passthru property to the ds3231 library for compatibility

day
The Current Day

get_month_days(*month=None*, *year=None*)
Return the number of days for the month of the given year

Parameters

- **month** (*int*) – (Optional) The month to use. If none is provided, current month is used.
- **year** (*int*) – (Optional) The year to check. If none is provided, current year is used.

hour
The Current Hour

is_leap_year(*year=None*)
Check if the year is a leap year

Parameters **year** (*int*) – (Optional) The year to check. If none is provided, current year is used.

minute
The Current Minute

month
The Current Month

now
The Current Date and Time in Named Tuple Style (Read Only)

second
The Current Second

set_date(*day*, *month*, *year*)
Set the date only

Parameters

- **day** (*int*) – The day we want to set the date to
- **month** (*int*) – The month we want to set the date to
- **year** (*int*) – The year we want to set the date to

set_time(*hour*, *minute*, *second=0*)
Set the time only

Parameters

- **hour** (*int*) – The hour we want to set the time to
- **minute** (*int*) – The minute we want to set the time to
- **second** (*int*) – (Optional) The second we want to set the time to (default=0)

unixtime
The Current Date and Time in Unix Time

weekday
The Current Day of the Week Value (0-6) where Sunday is 0

year
The Current Year

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

S

U

V

W

Y

Z