Adafruitfeatherwing Library Documentation

Release 1.0

Scott Shawcroft

Feb 26, 2020

Contents

1 Dependencies **3**
- 1.1 Installing from PyPI . . . 3

2 Contributing **5**

3 Documentation **7**

4 Table of Contents **9**
- 4.1 Simple tests . . . 9
- 4.2 Other Examples . . . 18
- 4.3 adafruit_featherwing.ina219_featherwing . . . 20
- 4.4 adafruit_featherwing.joy_featherwing . . . 22
- 4.5 adafruit_featherwing.alphanum_featherwing . . . 30
- 4.6 adafruit_featherwing.dotstar_featherwing . . . 31
- 4.7 adafruit_featherwing.neopixel_featherwing . . . 31
- 4.8 adafruit_featherwing.rtc_featherwing . . . 32
- 4.9 adafruit_featherwing.gps_featherwing . . . 33
- 4.10 adafruit_featherwing.matrix_featherwing . . . 34
- 4.11 adafruit_featherwing.minitft_featherwing . . . 35
- 4.12 adafruit_featherwing.tempmotion_featherwing . . . 36

5 Indices and tables **39**

Python Module Index **41**

Index **43**

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
- ST7735R
- ADXL34x
- ADT7410

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
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

Documentation

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

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
```

Listing 8: examples/featherwing_gps_simpletest.py

```
"""
This example will connect to the GPS at the default 9600 baudrate and
update once per second. Initialization is automatically handled and there
are some additional features such as MPH and KPH calculations.
"""
import time
from adafruit_featherwing import gps_featherwing

# Create a GPS featherwing instance.
gps = gps_featherwing.GPSFeatherWing()

# Main loop runs forever printing the location, etc. every second.
last_print = time.monotonic()
while True:
    # Make sure to call gps.update() every loop iteration and at least twice
    # as fast as data comes from the GPS unit (usually every second).
    # This returns a bool that's true if it parsed new data (you can ignore it
    # though if you don't care and instead look at the has_fix property).
    gps.update()
    # Every second print out current location details if there's a fix.
    current = time.monotonic()
    if current - last_print >= 1.0:
        last_print = current
        if not gps.has_fix:
            # Try again if we don't have a fix yet.
            print('Waiting for fix...')
            continue
        # Print out details about the fix like location, date, etc.
        print('=' * 40)  # Print a separator line.
        print('Fix timestamp: {}/{}/{} {:02}:{:02}:{:02}'.format(
            gps.timestamp.tm_mon,   # Grab parts of the time from the
            gps.timestamp.tm_mday,  # struct_time object that holds
            gps.timestamp.tm_year,  # the fix time.  Note you might
            gps.timestamp.tm_hour,  # not get all data like year, day,
            gps.timestamp.tm_min,   # month!
            gps.timestamp.tm_sec))
        print('Latitude: {0:.6f} degrees'.format(gps.latitude))
        print('Longitude: {0:.6f} degrees'.format(gps.longitude))
        print('Fix quality: {}'.format(gps.fix_quality))
        # Some attributes beyond latitude, longitude and timestamp are optional
        # and might not be present.  Check if they're None before trying to use!
        if gps.satellites is not None:
            print('# satellites: {}'.format(gps.satellites))
        if gps.altitude is not None:
            print('Altitude: {} meters'.format(gps.altitude))
        if gps.speed_knots is not None:
            print('Speed (Knots): {} knots'.format(gps.speed_knots))
        if gps.speed_mph is not None:
```

(continues on next page)

(continued from previous page)

```
            print('Speed (Miles Per Hour): {} MPH'.format(gps.speed_mph))
        if gps.speed_kph is not None:
            print('Speed (KM Per Hour): {} KPH'.format(gps.speed_kph))
        if gps.track_angle is not None:
            print('Track angle: {} degrees'.format(gps.track_angle))
        if gps.horizontal_dilution is not None:
            print('Horizontal dilution: {}'.format(gps.horizontal_dilution))
        if gps.height_geoid is not None:
            print('Height geo ID: {} meters'.format(gps.height_geoid))
```

Listing 9: examples/featherwing_matrix_simpletest.py

```
"""
This example will demonstrate some graphic effects and then
draw a smiley face and shift it around the display
"""
import time
from adafruit_featherwing import matrix_featherwing

matrix = matrix_featherwing.MatrixFeatherWing()

# Create a Fade-in Effect
matrix.brightness = 0
matrix.fill(True)
for level in range(0, 16):
    matrix.brightness = level
    time.sleep(0.1)

# Show the different Blink Rates
for level in range(3, -1, -1):
    matrix.blink_rate = level
    time.sleep(4)

# Create a Fade-out Effect
for level in range(15, -1, -1):
    matrix.brightness = level
    time.sleep(0.1)
matrix.fill(False)

# Reset the brightness to full
matrix.brightness = 15

# Clear the Screen
matrix.fill(False)

# Draw a Smiley Face
for row in range(2, 6):
    matrix[row, 0] = 1
    matrix[row, 7] = 1

for column in range(2, 6):
    matrix[0, column] = 1
    matrix[7, column] = 1

matrix[1, 1] = 1
matrix[1, 6] = 1
```

(continues on next page)

(continued from previous page)

```
matrix[6, 1] = 1
matrix[6, 6] = 1
matrix[2, 5] = 1
matrix[5, 5] = 1
matrix[2, 3] = 1
matrix[5, 3] = 1
matrix[3, 2] = 1
matrix[4, 2] = 1

# Move the Smiley Face Around
while True:
    for frame in range(0, 8):
        matrix.shift_right()
    for frame in range(0, 8):
        matrix.shift_down(True)
    for frame in range(0, 8):
        matrix.shift_left()
    for frame in range(0, 8):
        matrix.shift_up(True)
```

Listing 10: examples/featherwing_minitft_simpletest.py

```
"""
This example display a CircuitPython console and
print which button that is being pressed if any
"""
import time
from adafruit_featherwing import minitft_featherwing

minitft = minitft_featherwing.MiniTFTFeatherWing()

while True:
    buttons = minitft.buttons

    if buttons.right:
        print("Button RIGHT!")

    if buttons.down:
        print("Button DOWN!")

    if buttons.left:
        print("Button LEFT!")

    if buttons.up:
        print("Button UP!")

    if buttons.select:
        print("Button SELECT!")

    if buttons.a:
        print("Button A!")

    if buttons.b:
        print("Button B!")

    time.sleep(.001)
```

Listing 11: examples/featherwing_tempmotion_simpletest.py

```
"""
This example will show the current temperature in the Serial Console
whenever the FeatherWing senses that it has been tapped
"""

import time
from adafruit_featherwing import tempmotion_featherwing
temp_motion = tempmotion_featherwing.TempMotionFeatherWing()
temp_motion.enable_tap_detection()
while True:
    if temp_motion.events['tap']:
        print("The temperature is %f"%temp_motion.temperature)
    time.sleep(1)
```

4.2 Other Examples

Listing 12: 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):
```

(continues on next page)

(continued from previous page)

```
        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):
            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 13: 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)

```

(continues on next page)

(continued from previous page)

```
# 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:
                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

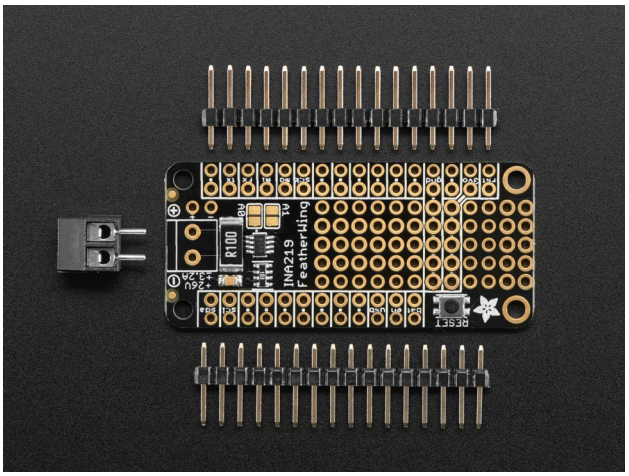
class adafruit_featherwing.ina219_featherwing.**INA219FeatherWing**(*i2c=None*)

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:
```

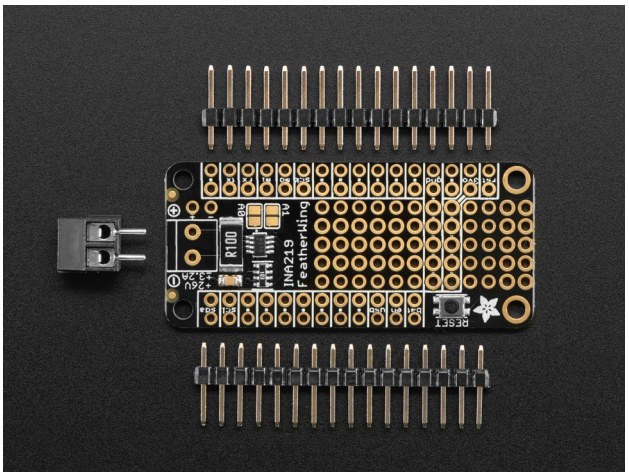
(continues on next page)

(continued from previous page)

```
    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.

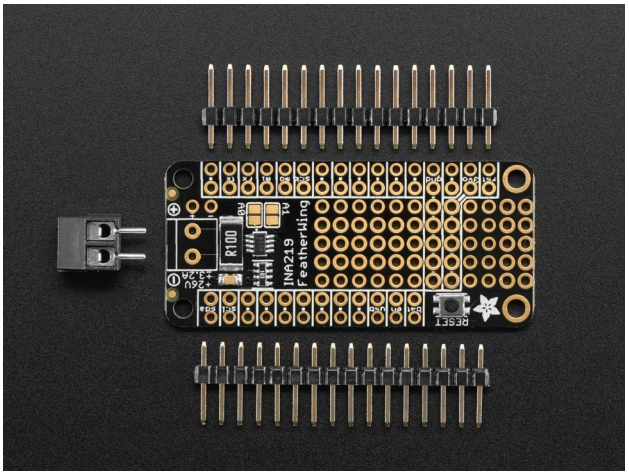
```
from adafruit_featherwing import ina219_featherwing
import time

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
```

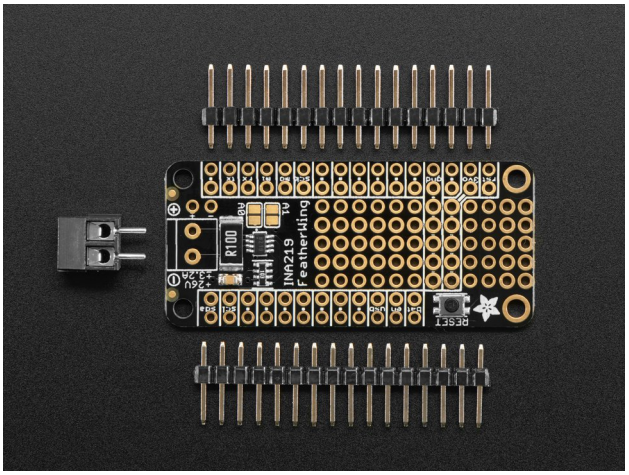
(continues on next page)

(continued from previous page)

```
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

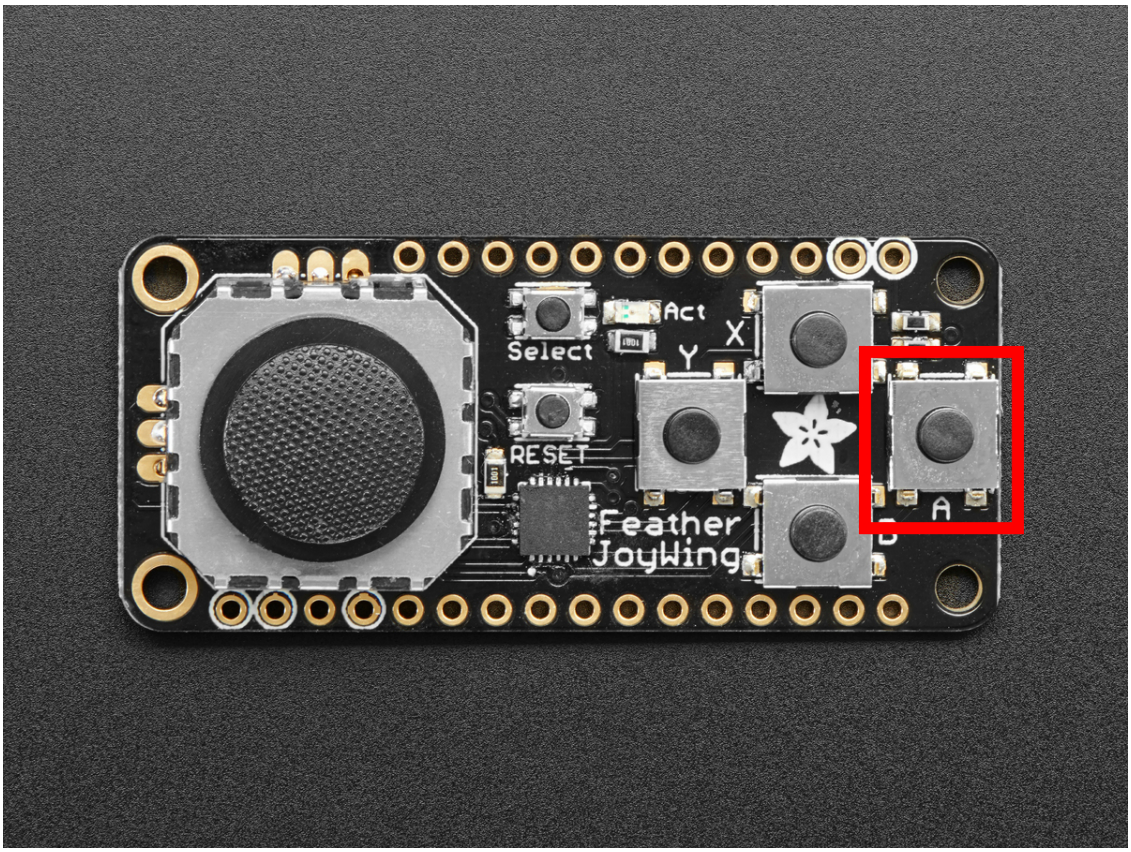
Helper for using the Joy FeatherWing.

- Author(s): Kattni Rembor

class adafruit_featherwing.joy_featherwing.**JoyFeatherWing**(*i2c=None*)
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.

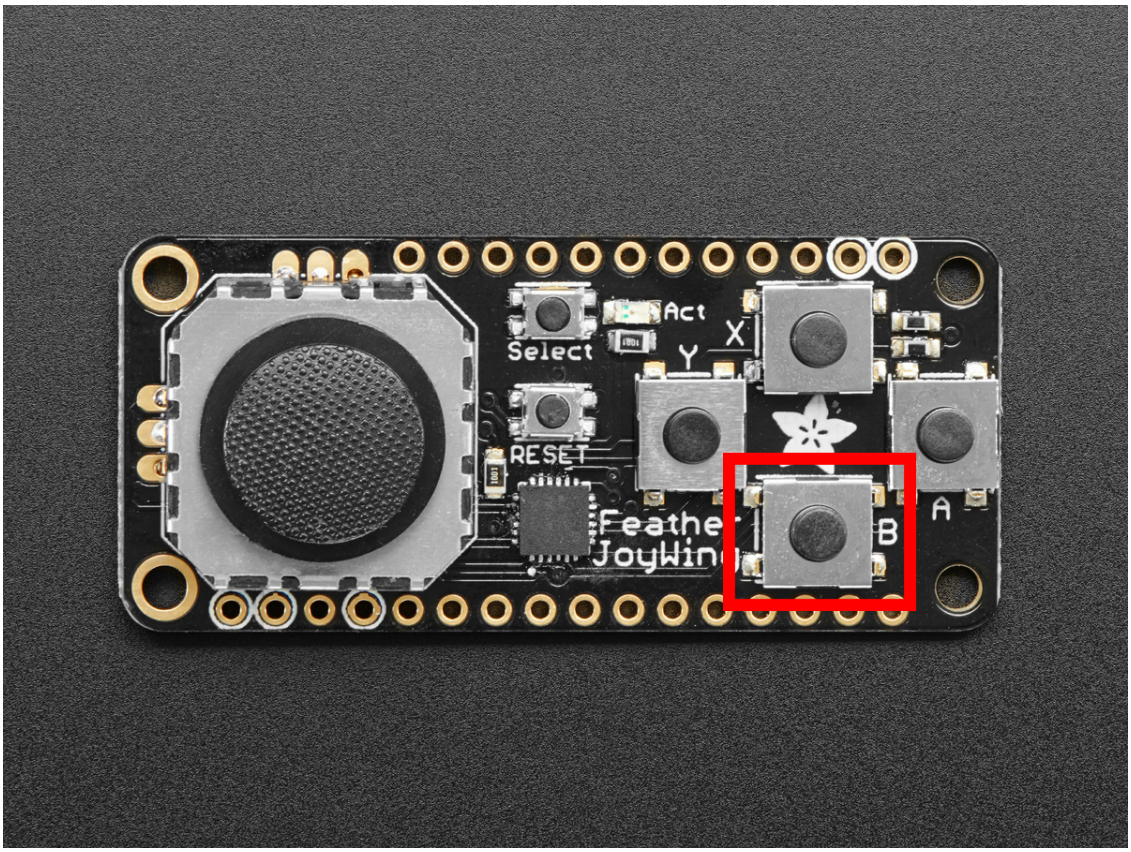
```
from adafruit_featherwing import joy_featherwing
import time

wing = joy_featherwing.JoyFeatherWing()

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

button_b

Joy featherwing button B.



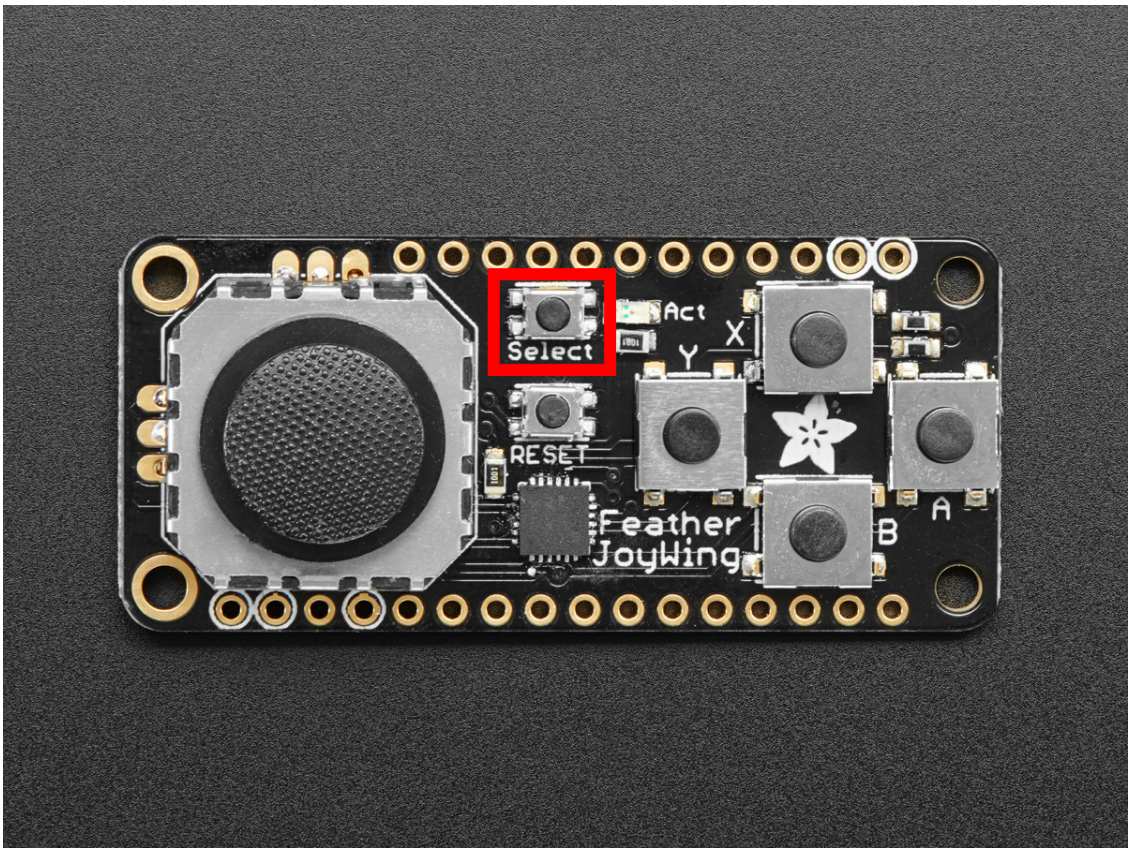
This example prints when button B is pressed.

```
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.



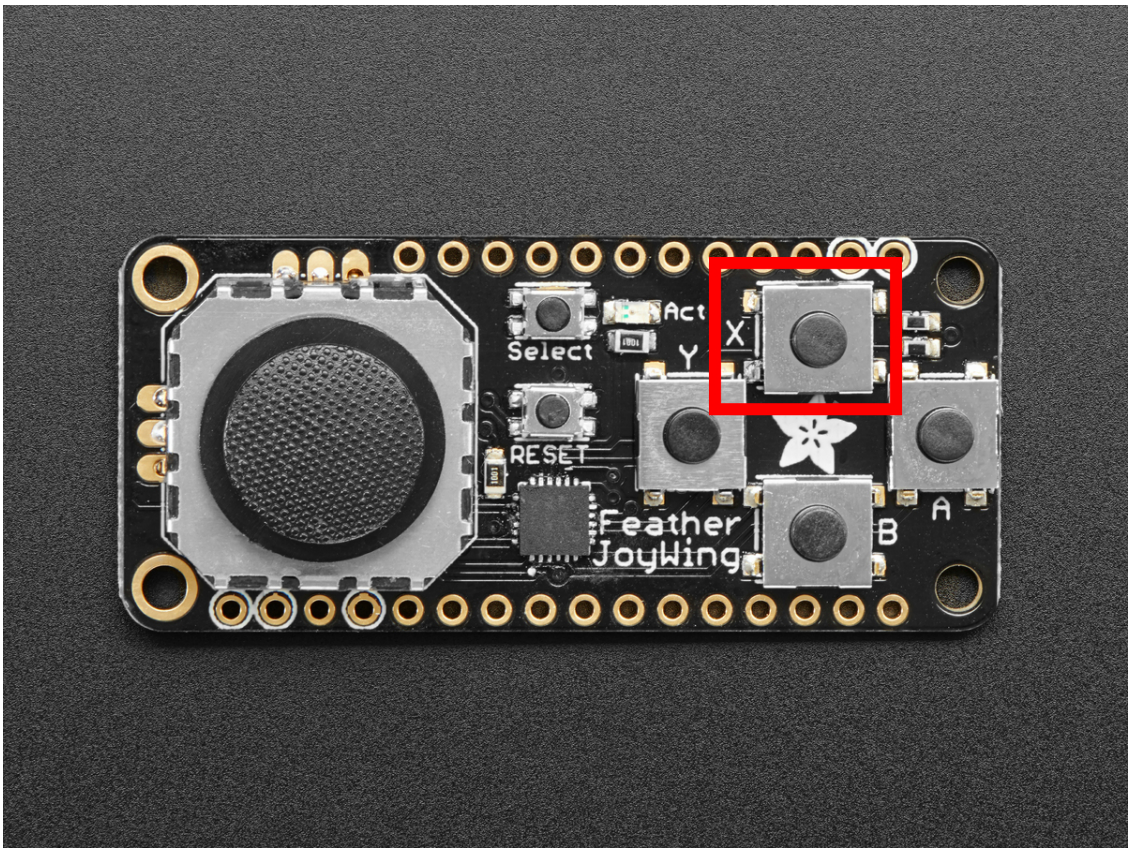
This example prints when button SELECT is pressed.

```
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.



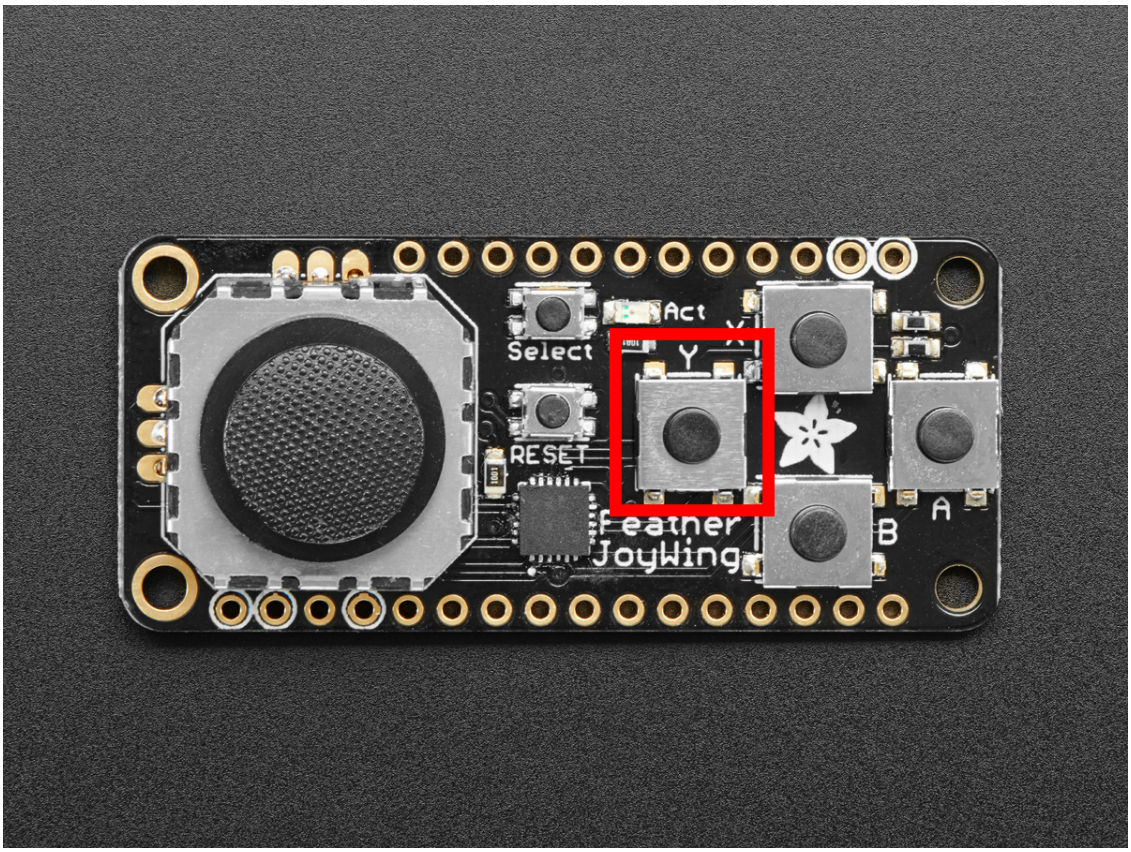
This example prints when button X is pressed.

```
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.



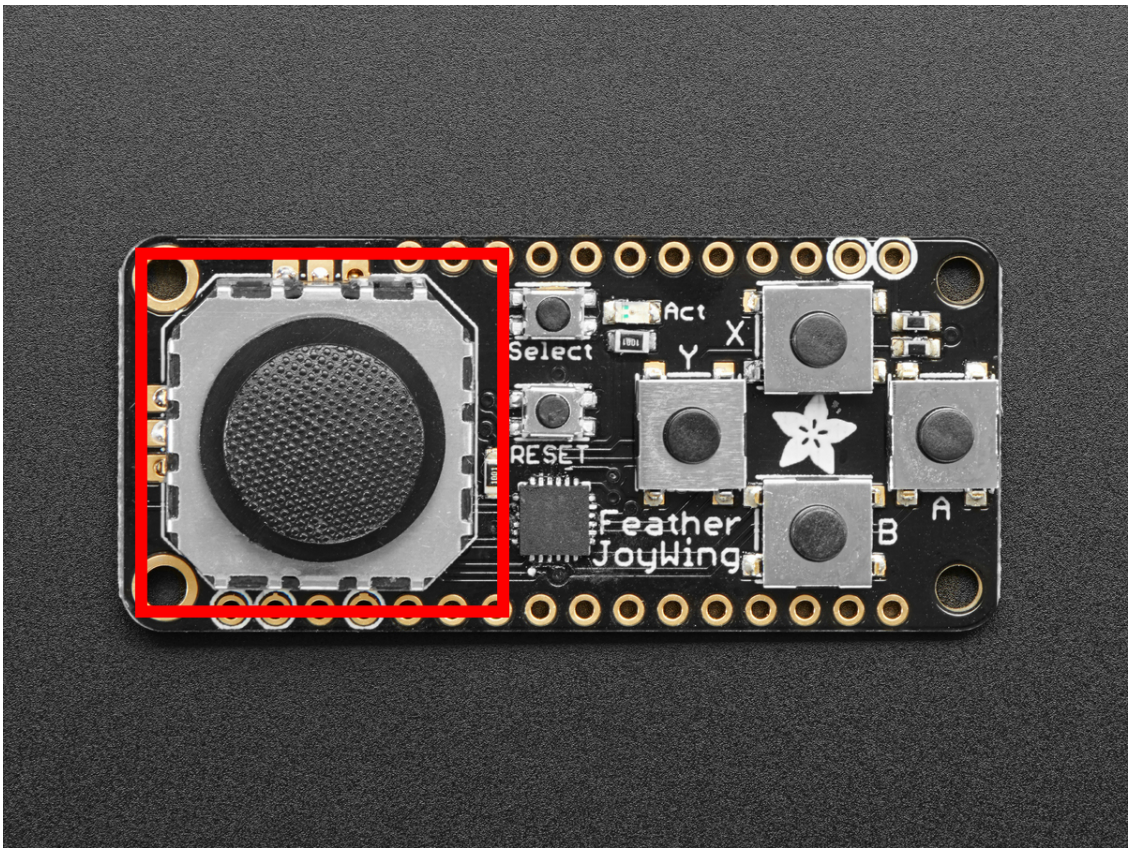
This example prints when button Y is pressed.

```
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.

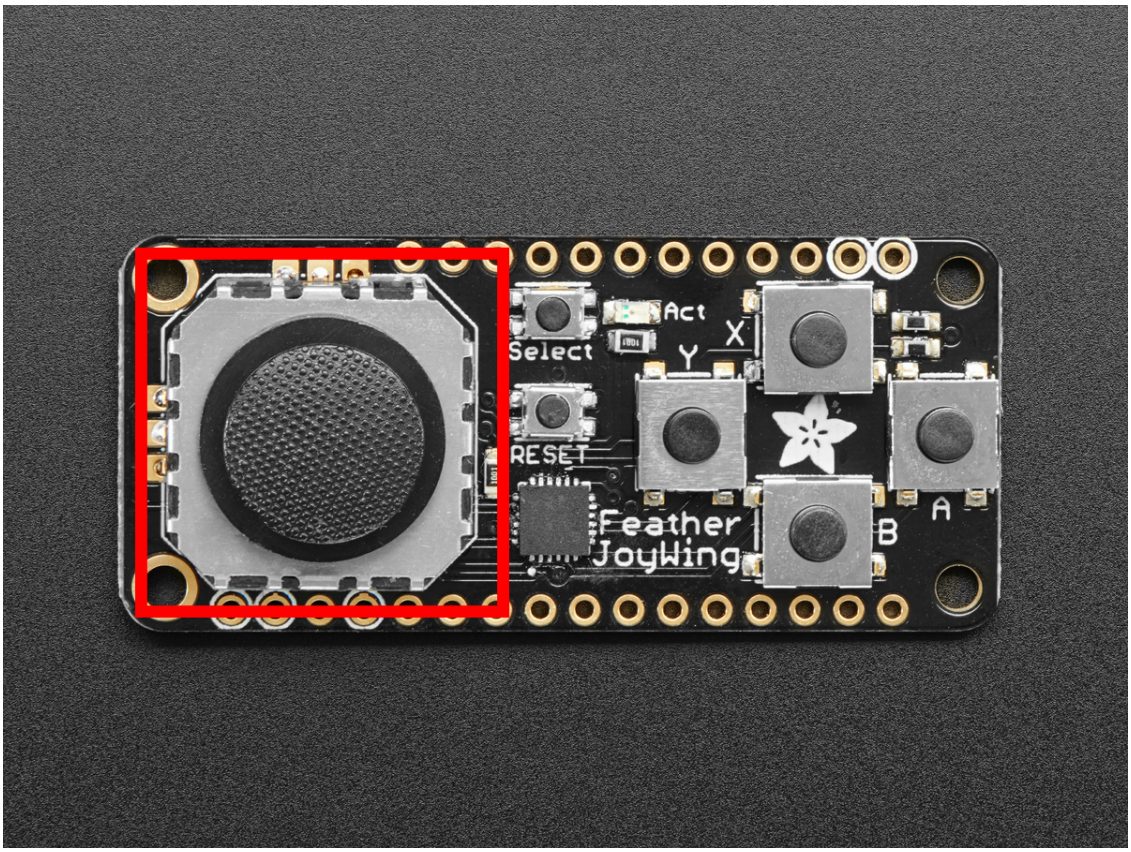
```
from adafruit_featherwing import joy_featherwing
import time

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.

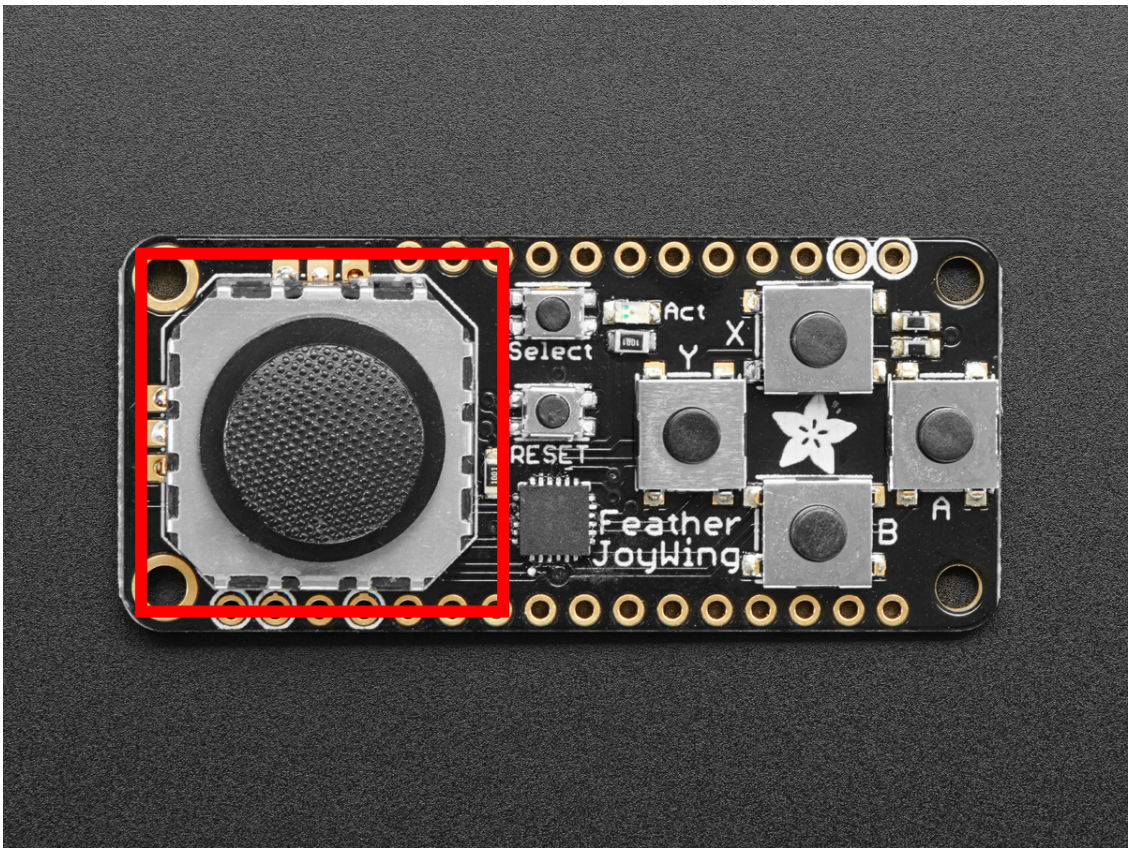
```
from adafruit_featherwing import joy_featherwing
import time

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*, *i2c=None*)

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.im 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):
```

(continues on next page)

(continued from previous page)

```
    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**(*i2c=None*)
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

4.9 adafruit_featherwing.gps_featherwing

Helper for using the Ultimate GPS FeatherWing.

- Author(s): Melissa LeBlanc-Williams

class adafruit_featherwing.gps_featherwing.**GPSFeatherWing**(*update_period=1000*, *baudrate=9600*)
Class representing an Ultimate GPS FeatherWing.

Automatically uses the feather's UART bus.

altitude
Return the Altitude in meters

fix_quality
Return the Fix Quality from the GPS

has_fix
Return whether the GPS has a Fix on some satellites

height_geoid
Return the Height GeoID in meters

horizontal_dilution
Return the Horizontal Dilution

latitude
Return the Current Latitude from the GPS

longitude
Return the Current Longitude from the GPS

read(*size*)
Read the UART for any information that may be on it

Parameters **size** (*int*) – The size in bytes of the data to retrieve

Returns Any data that is on the UART

Return type bytearray

satellites
Return the Number of Satellites we have a fix on

send_command(*command*)
Send a bytearray command to the GPS module

Parameters **command** (*bytearray*) – The command to send

speed_knots
Return the GPS calculated speed in knots

speed_kph
Return the GPS calculated speed in Kilometers per Hour

speed_mph
Return the GPS calculated speed in Miles per Hour

timestamp
Return the Fix Timestamp as a struct_time

track_angle
Return the Tracking angle in degrees

update()
Make sure to call `gps.update()` every loop iteration and at least twice as fast as data comes from the GPS unit (usually every second).

Returns Whether it has parsed new data

Return type bool

4.10 adafruit_featherwing.matrix_featherwing

Helper for using the Adafruit 8x16 LED Matrix FeatherWing.

- Author(s): Melissa LeBlanc-Williams

class adafruit_featherwing.matrix_featherwing.**MatrixFeatherWing**(*address=112*, *i2c=None*)

Class representing an Adafruit 8x16 LED Matrix FeatherWing.

Automatically uses the feather's I2C bus.

auto_write
Whether or not we are automatically updating If set to false, be sure to call show() to update

blink_rate
Blink Rate returns the current rate that the pixels blink. 0 = Not Blinking 1-3 = Successively slower blink rates

brightness
Brightness returns the current display brightness. 0-15 = Dimmest to Brightest Setting

fill(*fill*)
Turn all pixels on or off

Parameters **fill** (*bool*) – True turns all pixels on, False turns all pixels off

pixel(*x*, *y*, *color=None*)
Turn a pixel on or off or retrieve a pixel value

Parameters

- **x** (*int*) – The pixel row
- **y** (*int*) – The pixel column
- **color** (*int or bool*) – Whether to turn the pixel on or off

shift_down(*rotate=False*)
Shift all pixels down

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

shift_left(*rotate=False*)
Shift all pixels left

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

shift_right(*rotate=False*)
Shift all pixels right

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

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

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

show()
Update the Pixels. This is only needed if auto_write is set to False This can be very useful for more advanced graphics effects.

4.11 adafruit_featherwing.minitft_featherwing

Helper for using the Mini Color TFT with Joystick FeatherWing.

- Author(s): Melissa LeBlanc-Williams

class adafruit_featherwing.minitft_featherwing.**Buttons**(*up*, *down*, *left*, *right*, *a*, *b*, *select*)

a
Alias for field number 4

b
Alias for field number 5

down
Alias for field number 1

left
Alias for field number 2

right
Alias for field number 3

select
Alias for field number 6

up
Alias for field number 0

class adafruit_featherwing.minitft_featherwing.**MiniTFTFeatherWing**(*address=94*, *i2c=None*, *spi=None*, *cs=None*, *dc=None*)

Class representing an Mini Color TFT with Joystick FeatherWing.

Automatically uses the feather's I2C bus.

backlight
Return the current backlight duty cycle value

buttons
Return a set of buttons with current push values

4.12 adafruit_featherwing.tempmotion_featherwing

Helper for using the Adafruit ADXL343 + ADT7410 Sensor FeatherWing.

- Author(s): Melissa LeBlanc-Williams

class adafruit_featherwing.tempmotion_featherwing.**TempMotionFeatherWing**(*adxl343_address=83*, *adt7410_address=72*, *i2c=None*)

Class helper representing an Adafruit ADXL343 + ADT7410 Sensor FeatherWing.

Automatically uses the feather's I2C bus.

acceleration
Returns the ADXL343 Acceleration

configuration
Returns the ADT7410 Configuration

data_rate
The data rate of the sensor.

disable_freefall_detection()
Disable freefall detection

disable_motion_detection()
Disable motion detection

disable_tap_detection()
Disable freefall detection

enable_freefall_detection(***kwargs*)
Enable freefall detection

enable_motion_detection(***kwargs*)
Enable motion detection

enable_tap_detection(***kwargs*)
Enable freefall detection

events
Returns the ADXL343 Enabled Events

range
The measurement range of the sensor.

status
Returns the ADT7410 Status

temperature
Returns ADT7410 Temperature

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

adafruit_featherwing.alphanum_featherwing, 30
adafruit_featherwing.dotstar_featherwing, 31
adafruit_featherwing.gps_featherwing, 33
adafruit_featherwing.ina219_featherwing, 20
adafruit_featherwing.joy_featherwing, 22
adafruit_featherwing.matrix_featherwing, 34
adafruit_featherwing.minitft_featherwing, 35
adafruit_featherwing.neopixel_featherwing, 31
adafruit_featherwing.rtc_featherwing, 32
adafruit_featherwing.tempmotion_featherwing, 36

Index

A

a (*adafruit_featherwing.minitft_featherwing.Buttons attribute*), 36
acceleration (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing attribute*), 36
adafruit_featherwing.alphanum_featherwing (*module*), 30
adafruit_featherwing.dotstar_featherwing (*module*), 31
adafruit_featherwing.gps_featherwing (*module*), 33
adafruit_featherwing.ina219_featherwing (*module*), 20
adafruit_featherwing.joy_featherwing (*module*), 22
adafruit_featherwing.matrix_featherwing (*module*), 34
adafruit_featherwing.minitft_featherwing (*module*), 35
adafruit_featherwing.neopixel_featherwing (*module*), 31
adafruit_featherwing.rtc_featherwing (*module*), 32
adafruit_featherwing.tempmotion_featherwing (*module*), 36
AlphaNumFeatherWing (*class in adafruit_featherwing.alphanum_featherwing*), 30
altitude (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 33
auto_write (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing attribute*), 35

B

b (*adafruit_featherwing.minitft_featherwing.Buttons attribute*), 36
backlight (*adafruit_featherwing.minitft_featherwing.MiniTFTFeatherWing attribute*), 36
blink_rate (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing attribute*), 35
brightness (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing attribute*), 35
bus_voltage (*adafruit_featherwing.ina219_featherwing.INA219FeatherWing attribute*), 20
button_a (*adafruit_featherwing.joy_featherwing.JoyFeatherWing attribute*), 22
button_b (*adafruit_featherwing.joy_featherwing.JoyFeatherWing attribute*), 23
button_select (*adafruit_featherwing.joy_featherwing.JoyFeatherWing attribute*), 24
button_x (*adafruit_featherwing.joy_featherwing.JoyFeatherWing attribute*), 25
button_y (*adafruit_featherwing.joy_featherwing.JoyFeatherWing attribute*), 26
buttons (*adafruit_featherwing.minitft_featherwing.MiniTFTFeatherWing attribute*), 36
Buttons (*class in adafruit_featherwing.minitft_featherwing*), 35

C

configuration (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing attribute*), 36
current (*adafruit_featherwing.ina219_featherwing.INA219FeatherWing attribute*), 21

D

data_rate (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing attribute*), 36
datetime (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 32
day (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 32
disable_freefall_detection() (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing method*), 36
disable_motion_detection() (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing method*), 37

disable_tap_detection() (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing method*), 37
DotStarFeatherWing (*class in adafruit_featherwing.dotstar_featherwing*), 31
down (*adafruit_featherwing.minitft_featherwing.Buttons attribute*), 36

E

enable_freefall_detection() (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing method*), 37
enable_motion_detection() (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing method*), 37
enable_tap_detection() (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing method*), 37
events (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing attribute*), 37

F

fill() (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing method*), 35
fix_quality (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 33

G

get_month_days() (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing method*), 32
GPSFeatherWing (*class in adafruit_featherwing.gps_featherwing*), 33

H

has_fix (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34
height_geoid (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34
horizontal_dilution (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34
hour (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 32

I

INA219FeatherWing (*class in adafruit_featherwing.ina219_featherwing*), 20
is_leap_year() (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing method*), 32

J

JoyFeatherWing (*class in adafruit_featherwing.joy_featherwing*), 22
joystick (*adafruit_featherwing.joy_featherwing.JoyFeatherWing attribute*), 27
joystick_offset (*adafruit_featherwing.joy_featherwing.JoyFeatherW attribute*), 28

L

latitude (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34
left (*adafruit_featherwing.minitft_featherwing.Buttons attribute*), 36
longitude (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34

M

MatrixFeatherWing (*class in adafruit_featherwing.matrix_featherwing*), 34
MiniTFTFeatherWing (*class in adafruit_featherwing.minitft_featherwing*), 36
minute (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 33
month (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 33

N

NeoPixelFeatherWing (*class in adafruit_featherwing.neopixel_featherwing*), 31
now (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 33

P

pixel() (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing method*), 35

R

range (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeathe attribute*), 37
read() (*adafruit_featherwing.gps_featherwing.GPSFeatherWing method*), 34
right (*adafruit_featherwing.minitft_featherwing.Buttons attribute*), 36
RTCFeatherWing (*class in adafruit_featherwing.rtc_featherwing*), 32

S

satellites (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34

second (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 33
select (*adafruit_featherwing.minitft_featherwing.Buttons attribute*), 36
send_command() (*adafruit_featherwing.gps_featherwing.GPSFeatherWing method*), 34
set_date() (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing method*), 33
set_time() (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing method*), 33
shift_down() (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing method*), 35
shift_down() (*adafruit_featherwing.neopixel_featherwing.NeoPixelFeatherWing method*), 31
shift_left() (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing method*), 35
shift_right() (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing method*), 35
shift_up() (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing method*), 35
shift_up() (*adafruit_featherwing.neopixel_featherwing.NeoPixelFeatherWing method*), 32
show() (*adafruit_featherwing.matrix_featherwing.MatrixFeatherWing method*), 35
shunt_voltage (*adafruit_featherwing.ina219_featherwing.INA219FeatherWing attribute*), 21
speed_knots (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34
speed_kph (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34
speed_mph (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34
status (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing attribute*), 37

T

temperature (*adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing attribute*), 37
TempMotionFeatherWing (*class in adafruit_featherwing.tempmotion_featherwing*), 36
timestamp (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34
track_angle (*adafruit_featherwing.gps_featherwing.GPSFeatherWing attribute*), 34

U

unixtime (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 33
up (*adafruit_featherwing.minitft_featherwing.Buttons attribute*), 36
update() (*adafruit_featherwing.gps_featherwing.GPSFeatherWing method*), 34

V

voltage (*adafruit_featherwing.ina219_featherwing.INA219FeatherWing attribute*), 22

W

weekday (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 33

Y

year (*adafruit_featherwing.rtc_featherwing.RTCFeatherWing attribute*), 33

Z

zero_joystick() (*adafruit_featherwing.joy_featherwing.JoyFeatherW method*), 29