
Adafruitfeatherwing Library Documentation

Release 1.0

Scott Shawcroft

Jan 15, 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](#).

4.1 Simple tests

Ensure your device works with this simple test.

Listing 1: examples/featherwing_ina219_simpletest.py

```
1 """ Example to print out the voltage and current using the INA219 """
2 import time
3 from adafruit_featherwing import ina219_featherwing
4
5 INA219 = ina219_featherwing.INA219FeatherWing()
6
7 while True:
8     print("Bus Voltage:    {} V".format(INA219.bus_voltage))
9     print("Shunt Voltage: {} V".format(INA219.shunt_voltage))
10    print("Voltage:        {} V".format(INA219.voltage))
11    print("Current:         {} mA".format(INA219.current))
12    print("")
13    time.sleep(0.5)
```

Listing 2: examples/featherwing_joy_simpletest.py

```
1 """This example zeros the joystick, and prints when the joystick moves
2    or the buttons are pressed."""
3 import time
4 from adafruit_featherwing import joy_featherwing
5
6 wing = joy_featherwing.JoyFeatherWing()
7 last_x = 0
8 last_y = 0
9
10 while True:
11     x, y = wing.joystick
```

(continues on next page)

(continued from previous page)

```

12     if (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
13         last_x = x
14         last_y = y
15         print(x, y)
16     if wing.button_a:
17         print("Button A!")
18     if wing.button_b:
19         print("Button B!")
20     if wing.button_x:
21         print("Button X!")
22     if wing.button_y:
23         print("Button Y!")
24     if wing.button_select:
25         print("Button SELECT!")
26     time.sleep(.01)

```

Listing 3: examples/featherwing_alphanum_simpletest.py

```

1  """This example changes the fill, brightness, blink rates,
2  shows number and text printing, displays a counter
3  and then shows off the new marquee features."""
4
5  from time import sleep
6  from adafruit_featherwing import alphanum_featherwing
7
8  display = alphanum_featherwing.AlphaNumFeatherWing()
9
10 #Fill and empty all segments
11 for count in range(0, 3):
12     display.fill(True)
13     sleep(0.5)
14     display.fill(False)
15     sleep(0.5)
16
17 #Display a number and text
18 display.print(1234)
19 sleep(1)
20 display.print('Text')
21
22 #Change brightness
23 for brightness in range(0, 16):
24     display.brightness = brightness
25     sleep(0.1)
26
27 #Change blink rate
28 for blink_rate in range(3, 0, -1):
29     display.blink_rate = blink_rate
30     sleep(4)
31 display.blink_rate = 0
32
33 #Show a counter using decimals
34 count = 975.0
35 while count < 1025:
36     count += 1
37     display.print(count)
38     sleep(0.1)

```

(continues on next page)

(continued from previous page)

```

39
40 #Show the Marquee
41 display.marquee('This is a really long message!!! ', 0.2)

```

Listing 4: examples/featherwing_dotstar_simpletest.py

```

1  """
2  This plays various animations
3  and then draws random pixels at random locations
4  """
5
6  from time import sleep
7  import random
8  from adafruit_featherwing import dotstar_featherwing
9
10 dotstar = dotstar_featherwing.DotStarFeatherWing()
11
12 # HELPERS
13 # a random color 0 -> 224
14 def random_color():
15     return random.randrange(0, 8) * 32
16
17 # Fill screen with random colors at random brightnesses
18 for i in range(0, 5):
19     dotstar.fill((random_color(), random_color(), random_color()))
20     dotstar.brightness = random.randrange(2, 10) / 10
21     sleep(.2)
22
23 # Set display to 30% brightness
24 dotstar.brightness = 0.3
25
26 # Create a gradient drawing each pixel
27 for x in range(0, dotstar.columns):
28     for y in range(dotstar.rows - 1, -1, -1):
29         dotstar[x, y] = (y * 42, 255, y * 42, 1)
30
31 #Rotate everything left 36 frames
32 for i in range(0, 36):
33     dotstar.shift_down(True)
34
35 # Draw dual gradient and then update
36 dotstar.auto_write = False
37 for y in range(0, dotstar.rows):
38     for x in range(0, 6):
39         dotstar[x, y] = (y * 84, x * 42, x * 42, 1)
40     for x in range(6, 12):
41         dotstar[x, y] = (255 - (y * 84), 255 - ((x - 6) * 42), 255 - ((x - 6) * 42),
42             ↪1)
43
44 # Rotate everything left 36 frames
45 for i in range(0, 36):
46     dotstar.shift_left(True)
47     dotstar.shift_up(True)
48     dotstar.show()
49 dotstar.auto_write = True

```

(continues on next page)

(continued from previous page)

```

50 # Shift pixels without rotating for an animated screen wipe
51 for i in range(0, 6):
52     dotstar.shift_down()
53
54 # Show pixels in random locations of random color
55 # Bottom left corner is (0,0)
56 while True:
57     x = random.randrange(0, dotstar.columns)
58     y = random.randrange(0, dotstar.rows)
59     dotstar[x, y] = (random_color(), random_color(), random_color())
60     sleep(.1)

```

Listing 5: examples/featherwing_neopixel_simpletest.py

```

1  """
2  This example plays various animations
3  and then draws random pixels at random locations
4  """
5
6  from time import sleep
7  import random
8  from adafruit_featherwing import neopixel_featherwing
9
10 neopixel = neopixel_featherwing.NeoPixelFeatherWing()
11
12 # HELPERS
13 # a random color 0 -> 224
14 def random_color():
15     return random.randrange(0, 8) * 32
16
17 # Fill screen with random colors at random brightnesses
18 for i in range(0, 5):
19     neopixel.fill((random_color(), random_color(), random_color()))
20     neopixel.brightness = random.randrange(2, 10) / 10
21     sleep(.2)
22
23 # Set display to 30% brightness
24 neopixel.brightness = 0.3
25
26 # Create a gradient drawing each pixel
27 for x in range(0, neopixel.columns):
28     for y in range(neopixel.rows - 1, -1, -1):
29         neopixel[x, y] = (y * 63, 255, y * 63)
30
31 # Rotate everything left 36 frames
32 for i in range(0, 36):
33     neopixel.shift_down(True)
34     sleep(0.1)
35
36 # Draw dual gradient and then update
37 #neopixel.auto_write = False
38 for y in range(0, neopixel.rows):
39     for x in range(0, 4):
40         neopixel[x, y] = (y * 16 + 32, x * 8, 0)
41     for x in range(4, 8):
42         neopixel[x, y] = ((4 - y) * 16 + 32, (8 - x) * 8, 0)

```

(continues on next page)

(continued from previous page)

```

43 neopixel.show()
44
45 # Rotate everything left 36 frames
46 for i in range(0, 36):
47     neopixel.shift_left(True)
48     neopixel.shift_up(True)
49     neopixel.show()
50     sleep(0.1)
51 neopixel.auto_write = True
52
53 # Shift pixels without rotating for an animated screen wipe
54 for i in range(0, neopixel.rows):
55     neopixel.shift_down()
56     sleep(0.4)
57
58 # Show pixels in random locations of random color
59 # Bottom left corner is (0,0)
60 while True:
61     x = random.randrange(0, neopixel.columns)
62     y = random.randrange(0, neopixel.rows)
63     neopixel[x, y] = (random_color(), random_color(), random_color())
64     sleep(.1)

```

Listing 6: examples/featherwing_sevensegment_simpletest.py

```

1 """This example changes the fill, brightness, blink rates,
2 shows number and text printing, displays a counter
3 and then shows off the new marquee features."""
4
5 from time import sleep
6 from adafruit_featherwing import sevensegment_featherwing
7
8 display = sevensegment_featherwing.SevenSegmentFeatherWing()
9
10 #Fill and empty all segments
11 for count in range(0, 3):
12     display.fill(True)
13     sleep(0.5)
14     display.fill(False)
15     sleep(0.5)
16
17 #Display a number and text
18 display.print(1234)
19 sleep(1)
20 display.print('FEED')
21
22 #Change brightness
23 for brightness in range(0, 16):
24     display.brightness = brightness
25     sleep(0.1)
26
27 #Change blink rate
28 for blink_rate in range(3, 0, -1):
29     display.blink_rate = blink_rate
30     sleep(4)
31 display.blink_rate = 0

```

(continues on next page)

(continued from previous page)

```

32
33 #Show a counter using decimals
34 count = 975.0
35 while count < 1025:
36     count += 1
37     display.print(count)
38     sleep(0.1)
39
40 #Display a Time
41 hour = 12
42 for minute in range(15, 26):
43     display.print("{}:{}".format(hour, minute))
44     sleep(1)
45
46 #Show the Marquee
47 display.marquee('Deadbeef 192.168.100.102... ', 0.2)

```

Listing 7: examples/featherwing_rtc_simpletest.py

```

1  """
2  This example will allow you to set the date and time
3  and then loop through and display the current time
4  """
5  import time
6  from adafruit_featherwing import rtc_featherwing
7
8  days = ("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
9
10 # Create the RTC instance:
11 rtc = rtc_featherwing.RTCFeatherWing()
12
13 #pylint: disable-msg=using-constant-test
14 if True: # Change this to True to set the date and time
15     rtc.set_time(13, 34) # Set the time (seconds are optional)
16     print(rtc.now)
17     rtc.set_date(16, 1, 2016) # Set the date
18     print(rtc.now)
19     rtc.year = 2019 # Set just the Year
20     print(rtc.now)
21     rtc.month = 2 # Set Just the Month
22     print(rtc.now)
23     rtc.hour = 16 # Set just the hour
24     print(rtc.now)
25     rtc.weekday = 6 # Set just the day of the week (Sunday = 0)
26     print(rtc.now)
27     rtc.unixtime = 1550335257 # Or set the date and time with a unix timestamp
28
29 # Main loop:
30 while True:
31     now = rtc.now
32     print("The date is {} {}/{}/{}{}".format(days[now.weekday], now.day, now.month, now.
33     ↪year))
34     print("The time is {}:02:{}".format(now.hour, now.minute, now.second))
35     print("The UNIX timestamp is {}".format(rtc.unixtime))
36     print("The number of days in the current month is {}".format(rtc.get_month_
37     ↪days()))

```

(continues on next page)

(continued from previous page)

```

36     if rtc.is_leap_year():
37         print("This year is a leap year")
38     else:
39         print("This year is not a leap year")
40     time.sleep(1) # wait a second

```

Listing 8: examples/featherwing_gps_simpletest.py

```

1  """
2  This example will connect to the GPS at the default 9600 baudrate and
3  update once per second. Initialization is automatically handled and there
4  are some additional features such as MPH and KPH calculations.
5  """
6  import time
7  from adafruit_featherwing import gps_featherwing
8
9  # Create a GPS featherwing instance.
10 gps = gps_featherwing.GPSFeatherWing()
11
12 # Main loop runs forever printing the location, etc. every second.
13 last_print = time.monotonic()
14 while True:
15     # Make sure to call gps.update() every loop iteration and at least twice
16     # as fast as data comes from the GPS unit (usually every second).
17     # This returns a bool that's true if it parsed new data (you can ignore it
18     # though if you don't care and instead look at the has_fix property).
19     gps.update()
20     # Every second print out current location details if there's a fix.
21     current = time.monotonic()
22     if current - last_print >= 1.0:
23         last_print = current
24         if not gps.has_fix:
25             # Try again if we don't have a fix yet.
26             print('Waiting for fix...')
27             continue
28         # Print out details about the fix like location, date, etc.
29         print('=' * 40) # Print a separator line.
30         print('Fix timestamp: {}/{}/{} {:02}:{:02}:{:02}'.format(
31             gps.timestamp.tm_mon, # Grab parts of the time from the
32             gps.timestamp.tm_mday, # struct_time object that holds
33             gps.timestamp.tm_year, # the fix time. Note you might
34             gps.timestamp.tm_hour, # not get all data like year, day,
35             gps.timestamp.tm_min, # month!
36             gps.timestamp.tm_sec))
37         print('Latitude: {0:.6f} degrees'.format(gps.latitude))
38         print('Longitude: {0:.6f} degrees'.format(gps.longitude))
39         print('Fix quality: {}'.format(gps.fix_quality))
40         # Some attributes beyond latitude, longitude and timestamp are optional
41         # and might not be present. Check if they're None before trying to use!
42         if gps.satellites is not None:
43             print('# satellites: {}'.format(gps.satellites))
44         if gps.altitude is not None:
45             print('Altitude: {} meters'.format(gps.altitude))
46         if gps.speed_knots is not None:
47             print('Speed (Knots): {} knots'.format(gps.speed_knots))
48         if gps.speed_mph is not None:

```

(continues on next page)

(continued from previous page)

```

49     print('Speed (Miles Per Hour): {} MPH'.format(gps.speed_mph))
50     if gps.speed_kph is not None:
51         print('Speed (KM Per Hour): {} KPH'.format(gps.speed_kph))
52     if gps.track_angle is not None:
53         print('Track angle: {} degrees'.format(gps.track_angle))
54     if gps.horizontal_dilution is not None:
55         print('Horizontal dilution: {}'.format(gps.horizontal_dilution))
56     if gps.height_geoid is not None:
57         print('Height geo ID: {} meters'.format(gps.height_geoid))

```

Listing 9: examples/featherwing_matrix_simpletest.py

```

1  """
2  This example will demonstrate some graphic effects and then
3  draw a smiley face and shift it around the display
4  """
5  import time
6  from adafruit_featherwing import matrix_featherwing
7
8  matrix = matrix_featherwing.MatrixFeatherWing()
9
10 # Create a Fade-in Effect
11 matrix.brightness = 0
12 matrix.fill(True)
13 for level in range(0, 16):
14     matrix.brightness = level
15     time.sleep(0.1)
16
17 # Show the different Blink Rates
18 for level in range(3, -1, -1):
19     matrix.blink_rate = level
20     time.sleep(4)
21
22 # Create a Fade-out Effect
23 for level in range(15, -1, -1):
24     matrix.brightness = level
25     time.sleep(0.1)
26 matrix.fill(False)
27
28 # Reset the brightness to full
29 matrix.brightness = 15
30
31 # Clear the Screen
32 matrix.fill(False)
33
34 # Draw a Smiley Face
35 for row in range(2, 6):
36     matrix[row, 0] = 1
37     matrix[row, 7] = 1
38
39 for column in range(2, 6):
40     matrix[0, column] = 1
41     matrix[7, column] = 1
42
43 matrix[1, 1] = 1
44 matrix[1, 6] = 1

```

(continues on next page)

(continued from previous page)

```

45 matrix[6, 1] = 1
46 matrix[6, 6] = 1
47 matrix[2, 5] = 1
48 matrix[5, 5] = 1
49 matrix[2, 3] = 1
50 matrix[5, 3] = 1
51 matrix[3, 2] = 1
52 matrix[4, 2] = 1
53
54 # Move the Smiley Face Around
55 while True:
56     for frame in range(0, 8):
57         matrix.shift_right()
58     for frame in range(0, 8):
59         matrix.shift_down(True)
60     for frame in range(0, 8):
61         matrix.shift_left()
62     for frame in range(0, 8):
63         matrix.shift_up(True)

```

Listing 10: examples/featherwing_minitft_simpletest.py

```

1  """
2  This example display a CircuitPython console and
3  print which button that is being pressed if any
4  """
5  import time
6  from adafruit_featherwing import minitft_featherwing
7
8  minitft = minitft_featherwing.MiniTFTFeatherWing()
9
10 while True:
11     buttons = minitft.buttons
12
13     if buttons.right:
14         print("Button RIGHT!")
15
16     if buttons.down:
17         print("Button DOWN!")
18
19     if buttons.left:
20         print("Button LEFT!")
21
22     if buttons.up:
23         print("Button UP!")
24
25     if buttons.select:
26         print("Button SELECT!")
27
28     if buttons.a:
29         print("Button A!")
30
31     if buttons.b:
32         print("Button B!")
33
34     time.sleep(.001)

```

Listing 11: examples/featherwing_tempmotion_simpletest.py

```
1 """
2 This example will show the current temperature in the Serial Console
3 whenever the FeatherWing senses that it has been tapped
4 """
5
6 import time
7 from adafruit_featherwing import tempmotion_featherwing
8 temp_motion = tempmotion_featherwing.TempMotionFeatherWing()
9 temp_motion.enable_tap_detection()
10 while True:
11     if temp_motion.events['tap']:
12         print("The temperature is %f"%temp_motion.temperature)
13     time.sleep(1)
```

4.2 Other Examples

Listing 12: examples/featherwing_dotstar_palette_example.py

```
1 """
2 This creates a palette of colors, draws a pattern and
3 rotates through the palette creating a moving rainbow.
4 """
5
6 from math import sqrt, cos, sin, radians
7 from adafruit_featherwing import dotstar_featherwing
8
9 dotstar = dotstar_featherwing.DotStarFeatherWing()
10
11 # Remap the calculated rotation to 0 - 255
12 def remap(vector):
13     return int(((255 * vector + 85) * 0.75) + 0.5)
14
15 # Calculate the Hue rotation starting with Red as 0 degrees
16 def rotate(degrees):
17     cosA = cos(radians(degrees))
18     sinA = sin(radians(degrees))
19     red = cosA + (1.0 - cosA) / 3.0
20     green = 1./3. * (1.0 - cosA) + sqrt(1./3.) * sinA
21     blue = 1./3. * (1.0 - cosA) - sqrt(1./3.) * sinA
22     return (remap(red), remap(green), remap(blue))
23
24 palette = []
25 pixels = []
26
27 # Generate a rainbow palette
28 for degree in range(0, 360):
29     color = rotate(degree)
30     palette.append(color[0] << 16 | color[1] << 8 | color[2])
31
32 # Create the Pattern
33 for y in range(0, dotstar.rows):
34     for x in range(0, dotstar.columns):
```

(continues on next page)

(continued from previous page)

```

35         pixels.append(x * 30 + y * -30)
36
37     # Clear the screen
38     dotstar.fill()
39
40     # Start the Animation
41     dotstar.auto_write = False
42     while True:
43         for color in range(0, 360, 10):
44             for index in range(0, dotstar.rows * dotstar.columns):
45                 palette_index = pixels[index] + color
46                 if palette_index >= 360:
47                     palette_index -= 360
48                 elif palette_index < 0:
49                     palette_index += 360
50                 dotstar[index] = palette[palette_index]
51             dotstar.show()

```

Listing 13: examples/featherwing_neopixel_palette_example.py

```

1  """
2  This creates a palette of colors, draws a pattern and
3  rotates through the palette creating a moving rainbow.
4  """
5
6  from math import sqrt, cos, sin, radians
7  from adafruit_featherwing import neopixel_featherwing
8
9  neopixel = neopixel_featherwing.NeoPixelFeatherWing()
10
11  # Remap the calculated rotation to 0 - 255
12  def remap(vector):
13      return int(((255 * vector + 85) * 0.75) + 0.5)
14
15  # Calculate the Hue rotation starting with Red as 0 degrees
16  def rotate(degrees):
17      cosA = cos(radians(degrees))
18      sinA = sin(radians(degrees))
19      red = cosA + (1.0 - cosA) / 3.0
20      green = 1./3. * (1.0 - cosA) + sqrt(1./3.) * sinA
21      blue = 1./3. * (1.0 - cosA) - sqrt(1./3.) * sinA
22      return (remap(red), remap(green), remap(blue))
23
24  palette = []
25  pixels = []
26
27  # Generate a rainbow palette
28  for degree in range(0, 360):
29      color = rotate(degree)
30      palette.append(color[0] << 16 | color[1] << 8 | color[2])
31
32  # Create the Pattern
33  for y in range(0, neopixel.rows):
34      for x in range(0, neopixel.columns):
35          pixels.append(x * 30 + y * -30)
36

```

(continues on next page)

(continued from previous page)

```

37 # Clear the screen
38 neopixel.fill()
39
40 # Start the Animation
41 neopixel.auto_write = False
42 while True:
43     for color in range(0, 360, 10):
44         for index in range(0, neopixel.rows * neopixel.columns):
45             palette_index = pixels[index] + color
46             if palette_index >= 360:
47                 palette_index -= 360
48             elif palette_index < 0:
49                 palette_index += 360
50             neopixel[index] = palette[palette_index]
51         neopixel.show()

```

4.3 adafruit_featherwing.ina219_featherwing

Helper for using the [INA219 FeatherWing](#).

- Author(s): Kattni Rembor

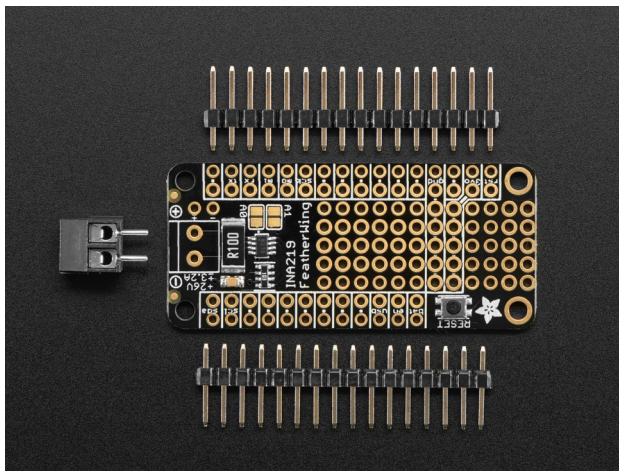
class `adafruit_featherwing.ina219_featherwing.INA219FeatherWing` (*i2c=None*)

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:

```

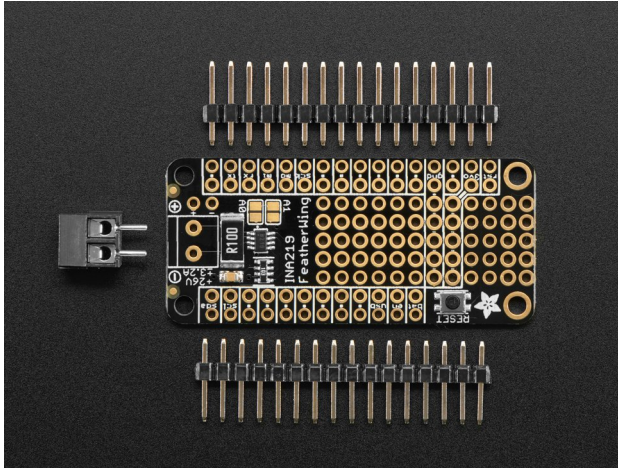
(continues on next page)

(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.

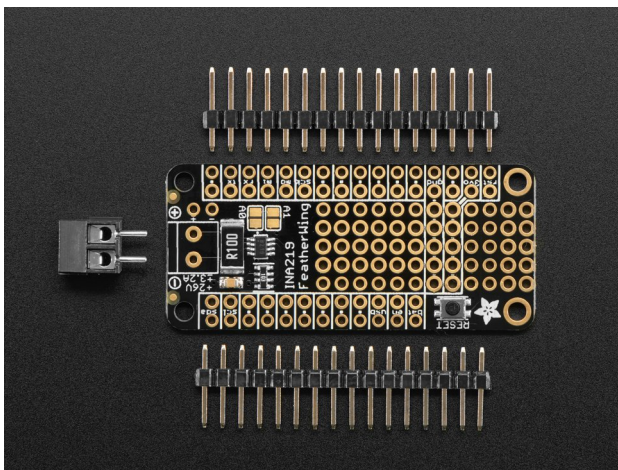
```
from adafruit_featherwing import ina219_featherwing
import time

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)

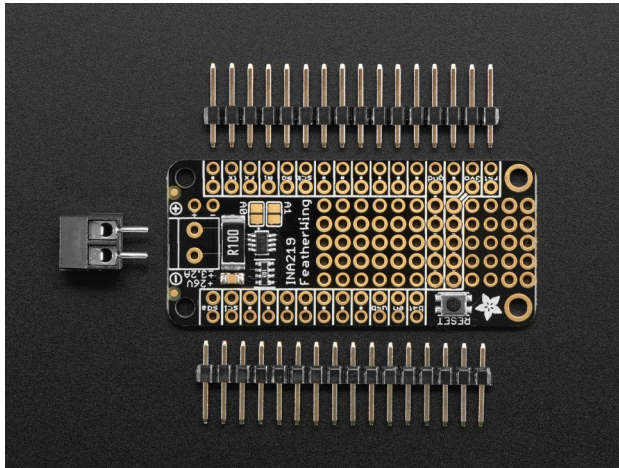
(continued from previous page)

```
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

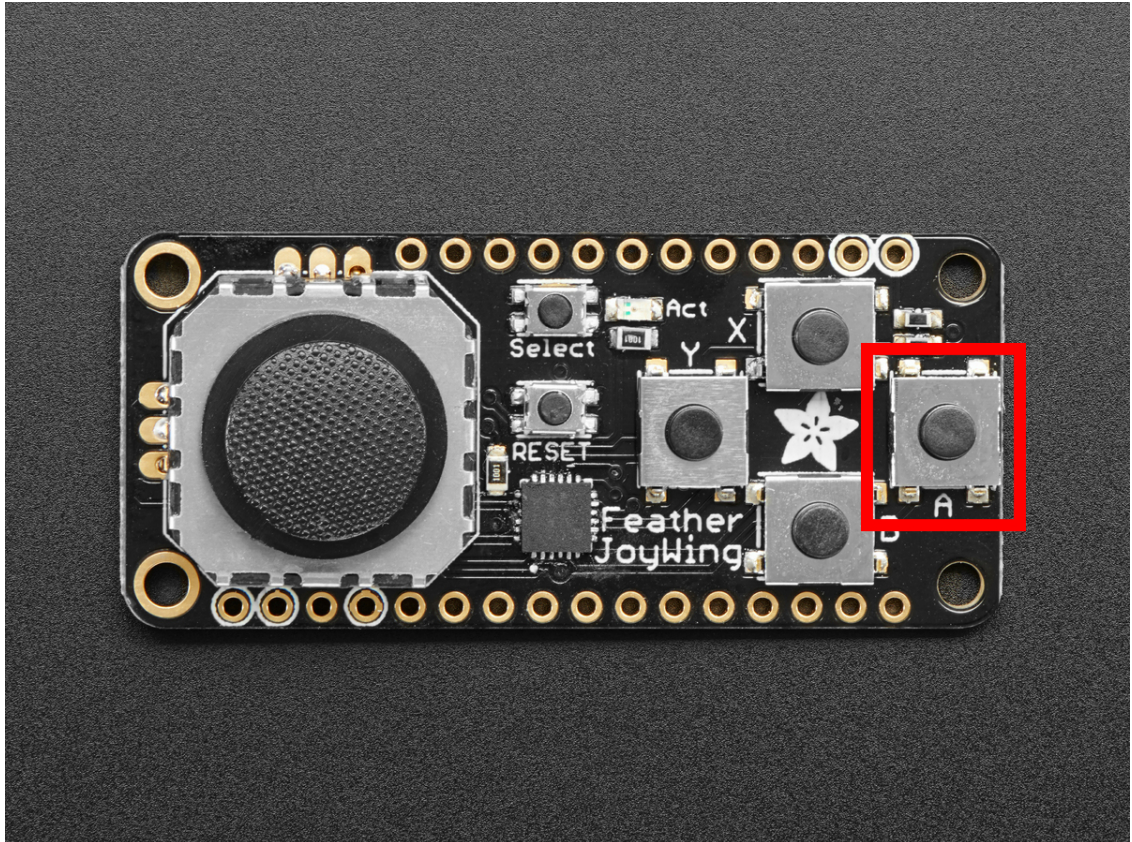
class adafruit_featherwing.joy_featherwing.**JoyFeatherWing** (*i2c=None*)

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.

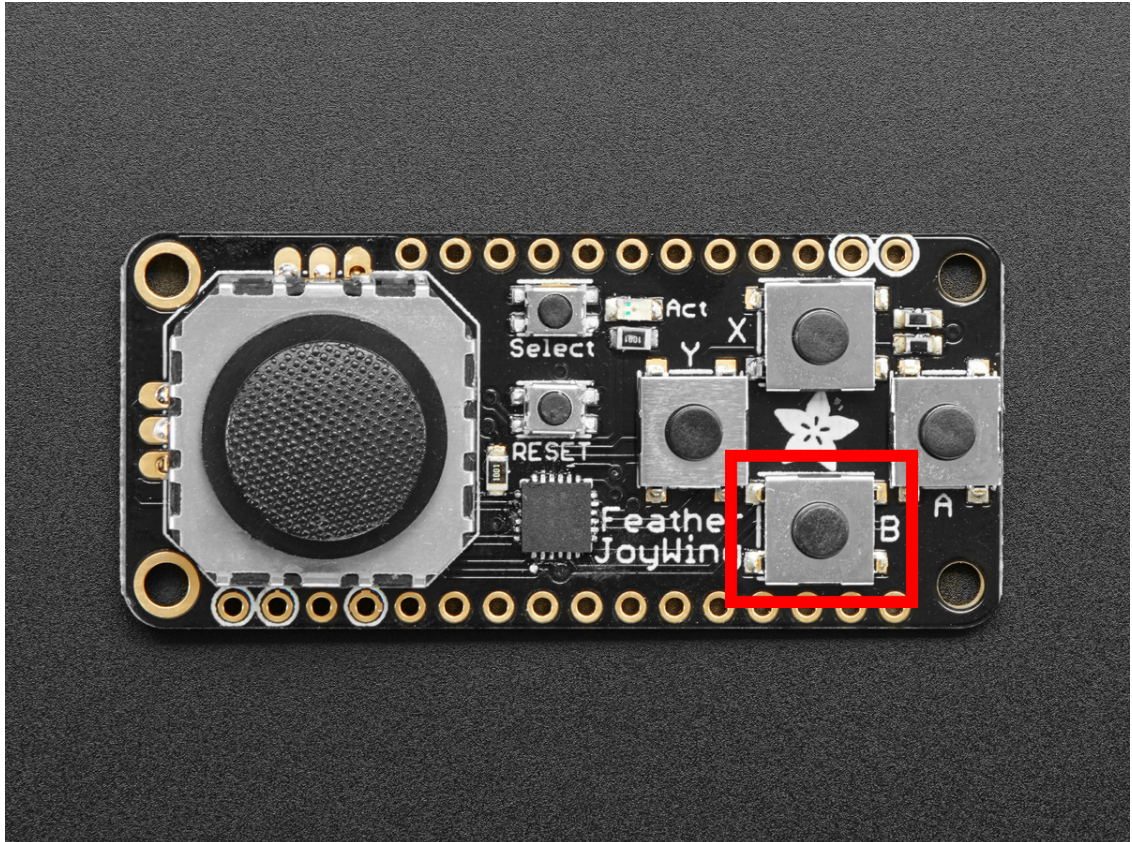
```
from adafruit_featherwing import joy_featherwing
import time

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.

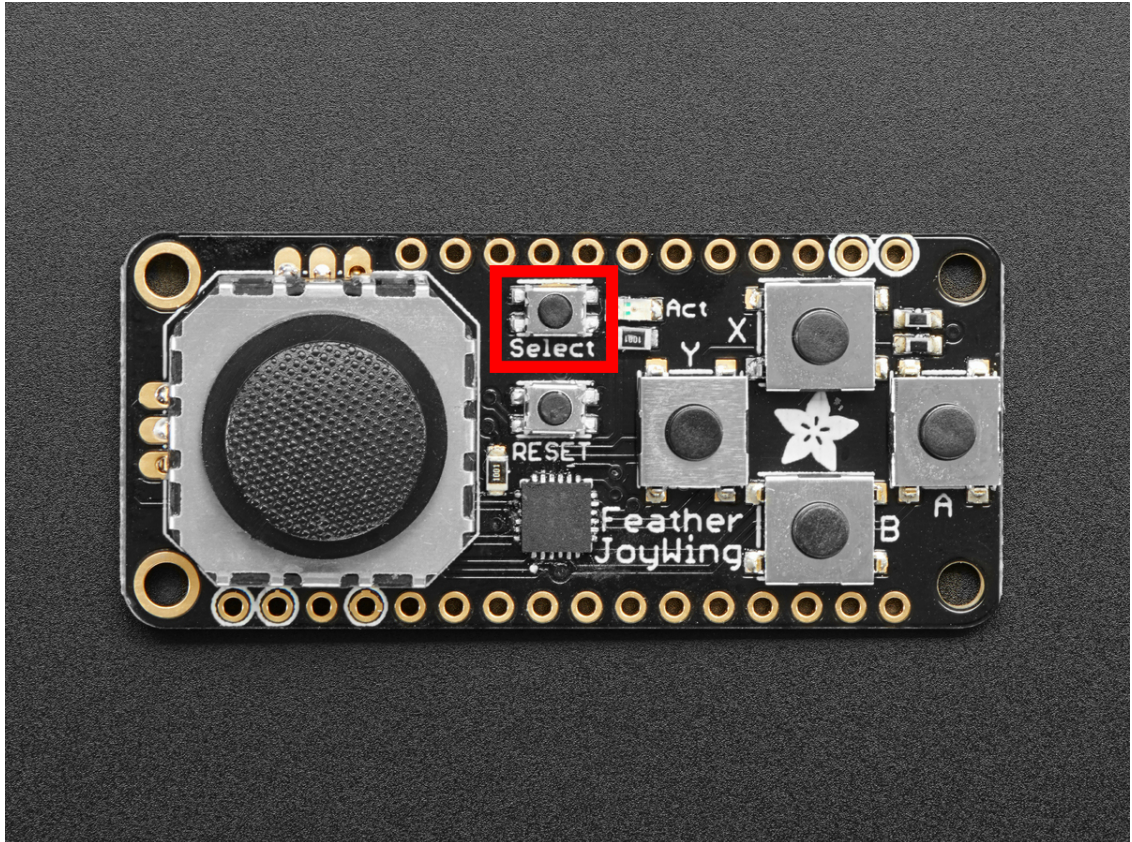
```
from adafruit_featherwing import joy_featherwing
import time

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.

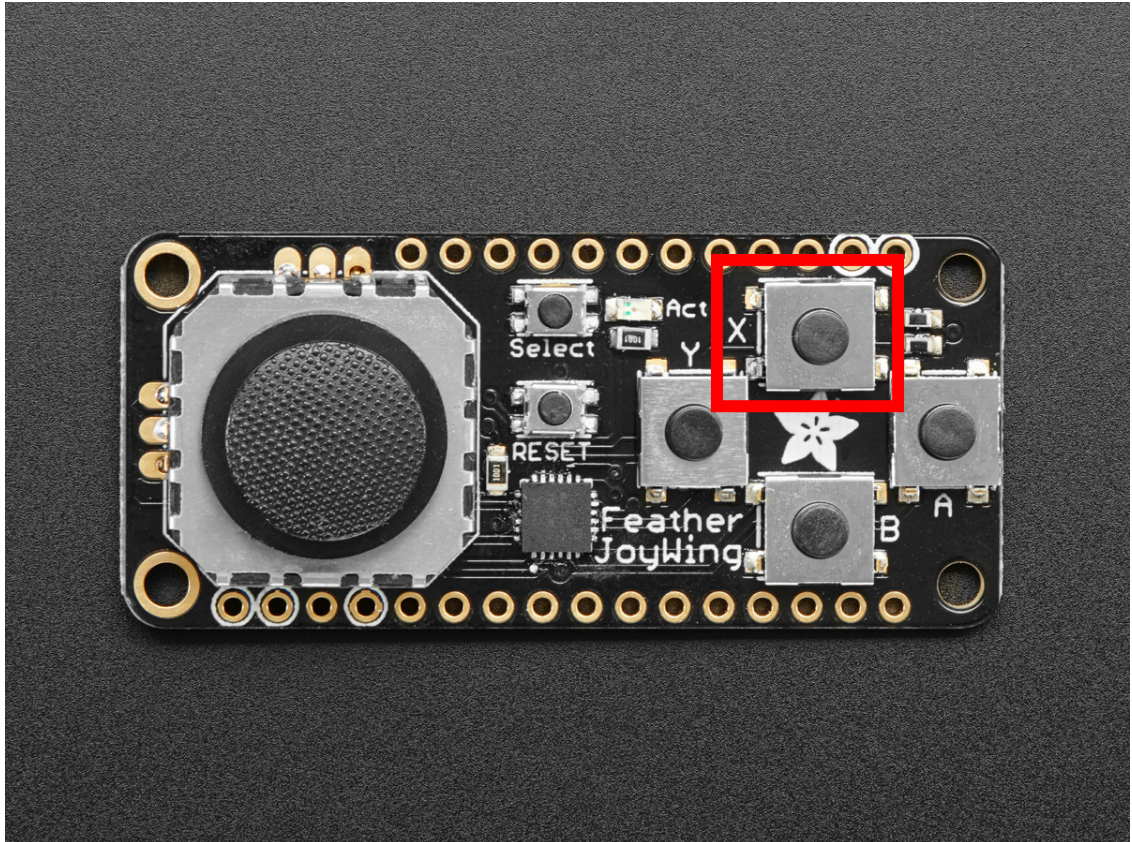
```
from adafruit_featherwing import joy_featherwing
import time

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.

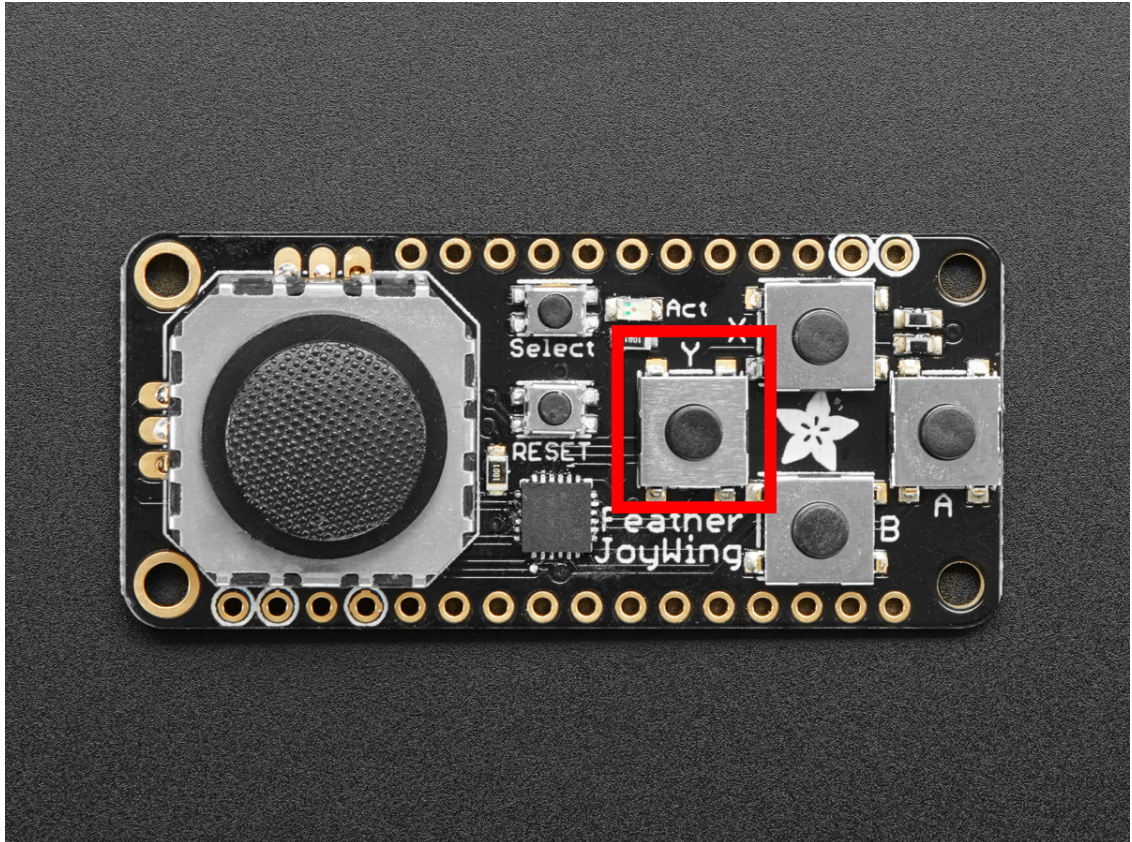
```
from adafruit_featherwing import joy_featherwing
import time

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.

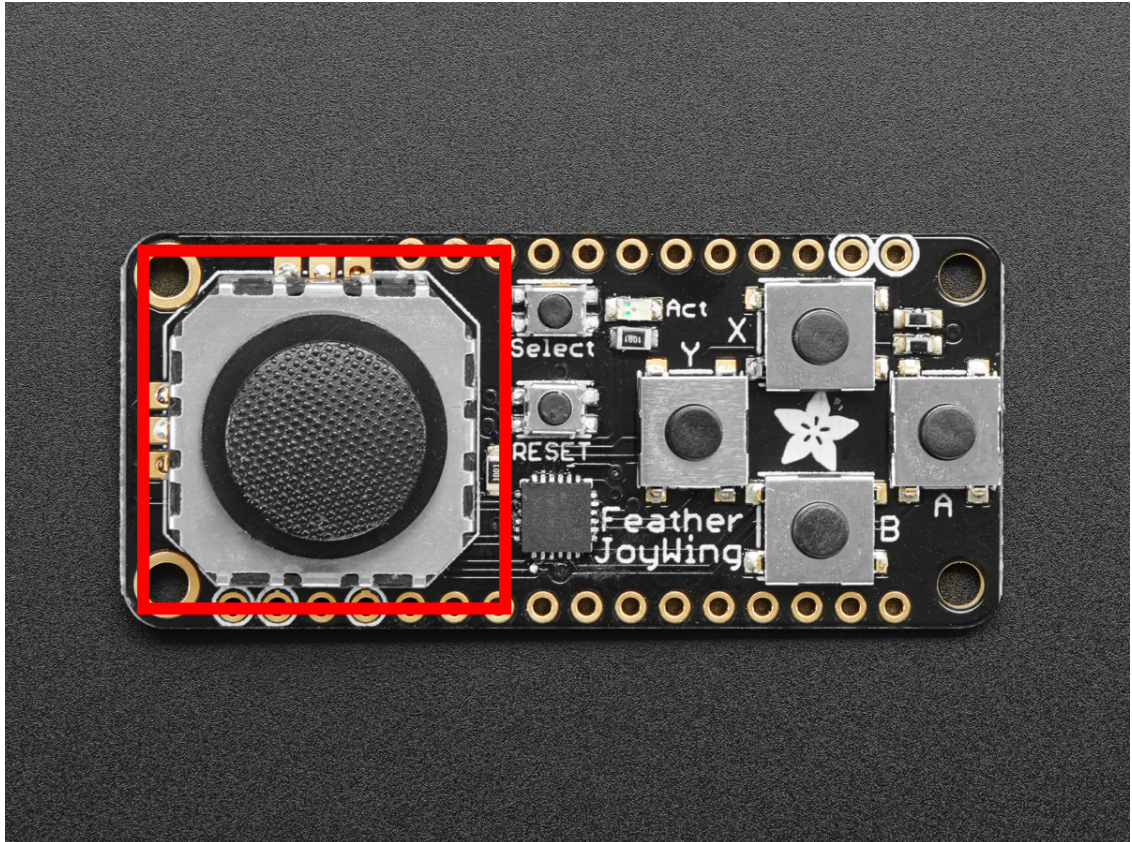
```
from adafruit_featherwing import joy_featherwing
import time

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.

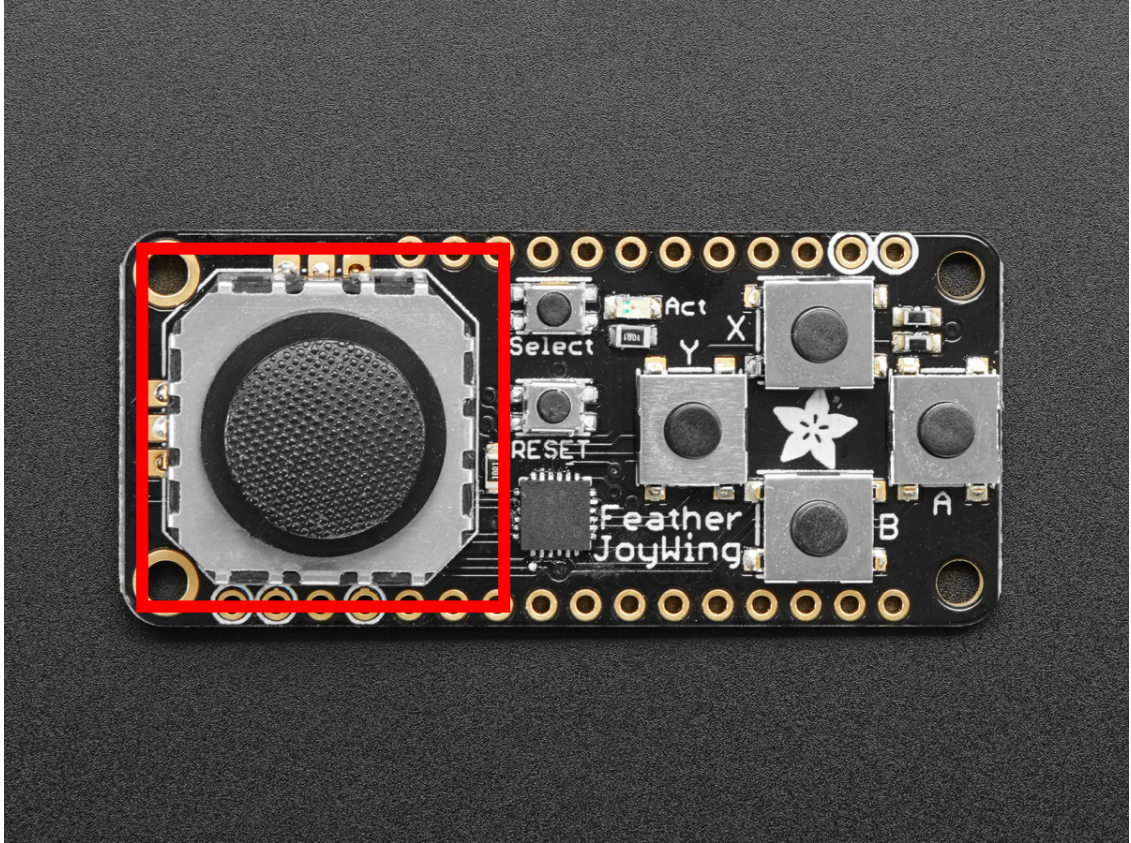
```
from adafruit_featherwing import joy_featherwing
import time

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.

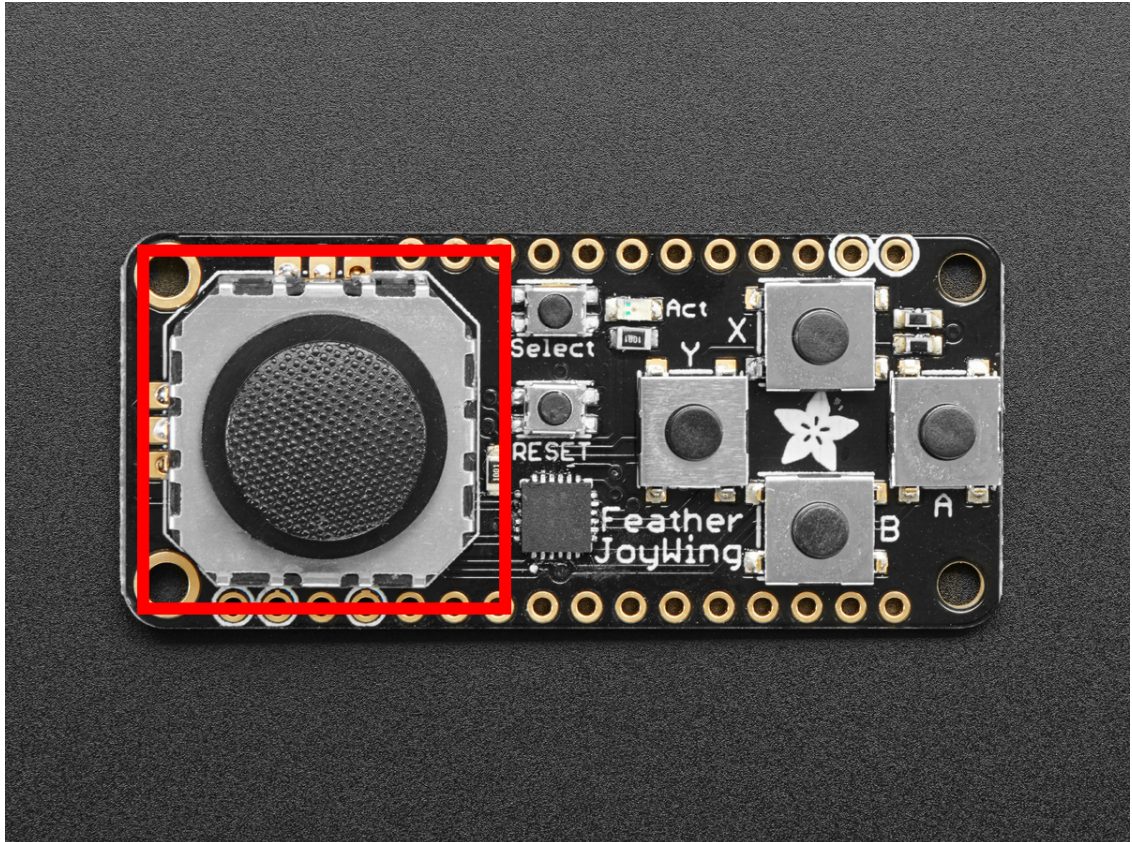
```
from adafruit_featherwing import joy_featherwing
import time

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.alphanumeric_featherwing

Helper for using the 14-Segment AlphaNumeric FeatherWing.

- Author(s): Melissa LeBlanc-Williams

```
class adafruit_featherwing.alphanumeric_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>,  
bright-  
ness=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.in  
object>,  
bright-  
ness=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`

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

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

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` (class in `adafruit_featherwing.tempmotion_featherwing`), 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` (class in `adafruit_featherwing.tempmotion_featherwing`), 37)

`enable_motion_detection()`

(`adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing` (class in `adafruit_featherwing.tempmotion_featherwing`), 37)

`enable_tap_detection()`

(`adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing` (class in `adafruit_featherwing.tempmotion_featherwing`), 37)

`events` (`adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing` attribute), 37

F

`fill()` (`adafruit_featherwing.matrix_featherwing.MatrixFeatherWing` (class in `adafruit_featherwing.matrix_featherwing`), 35)

`fix_quality` (`adafruit_featherwing.gps_featherwing.GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 33)

G

`get_month_days()` (`adafruit_featherwing rtc_featherwing.RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 32)

`GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 33

H

`has_fix` (`adafruit_featherwing.gps_featherwing.GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 34)

`height_geoid` (`adafruit_featherwing.gps_featherwing.GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 34)

`horizontal_dilution`

(`adafruit_featherwing.gps_featherwing.GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 34)

`hour` (`adafruit_featherwing rtc_featherwing.RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 32)

I

`INA219FeatherWing` (class in `adafruit_featherwing.ina219_featherwing`), 20

`is_leap_year()` (`adafruit_featherwing rtc_featherwing.RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 32)

J

`JoystickFeatherWing` (class in `adafruit_featherwing.joy_featherwing`), 22

`joystick` (`adafruit_featherwing.joy_featherwing.JoyFeatherWing` (class in `adafruit_featherwing.joy_featherwing`), 27)

`joystick_offset` (`adafruit_featherwing.joy_featherwing.JoyFeatherWing` (class in `adafruit_featherwing.joy_featherwing`), 28)

L

`latitude` (`adafruit_featherwing.gps_featherwing.GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 34)

`left` (`adafruit_featherwing.minitft_featherwing.Buttons` attribute), 36

`longitude` (`adafruit_featherwing.gps_featherwing.GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 34)

M

`MatrixFeatherWing` (class in `adafruit_featherwing.matrix_featherwing`), 34

`MiniTFTFeatherWing` (class in `adafruit_featherwing.minitft_featherwing`), 36

`minute` (`adafruit_featherwing rtc_featherwing.RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 33)

`month` (`adafruit_featherwing rtc_featherwing.RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 33)

(`adafruit_featherwing rtc_featherwing.RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 33)

N

`NeopixelFeatherWing` (class in `adafruit_featherwing.neopixel_featherwing`), 31

`now` (`adafruit_featherwing rtc_featherwing.RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 33)

P

`pixel()` (`adafruit_featherwing.matrix_featherwing.MatrixFeatherWing` (class in `adafruit_featherwing.matrix_featherwing`), 35)

(`adafruit_featherwing.matrix_featherwing.MatrixFeatherWing` (class in `adafruit_featherwing.matrix_featherwing`), 35)

R

`range` (`adafruit_featherwing.tempmotion_featherwing.TempMotionFeatherWing` (class in `adafruit_featherwing.tempmotion_featherwing`), 37)

`read()` (`adafruit_featherwing.gps_featherwing.GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 34)

`right` (`adafruit_featherwing.minitft_featherwing.Buttons` attribute), 36

`RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 32

(`adafruit_featherwing rtc_featherwing.RTCFeatherWing` (class in `adafruit_featherwing rtc_featherwing`), 32)

`satellites` (`adafruit_featherwing.gps_featherwing.GPSFeatherWing` (class in `adafruit_featherwing.gps_featherwing`), 34)

second (adafruit_featherwing.rtc_featherwing.RTCFeatherWing
attribute), 33

select (adafruit_featherwing.minift_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.minift_featherwing.Buttons at-
tribute), 36

update() (adafruit_featherwing.gps_featherwing.GPSFeatherWing
method), 34