
Adafruit seesaw Library Documentation

Release 1.0

Dean Miller

Feb 02, 2021

Contents

1	Dependencies	3
2	Installing from PyPI	5
3	Usage Example	7
4	Contributing	9
5	Documentation	11
6	Table of Contents	13
6.1	Simple test	13
6.2	Other Examples	14
6.3	adafruit_seesaw.seesaw	20
6.3.1	Implementation Notes	20
6.4	adafruit_seesaw.crickit - Pin definition for Adafruit CRICKIT	21
6.5	adafruit_seesaw.analoginput	21
6.6	adafruit_seesaw.digitalio	22
6.7	adafruit_seesaw.keypad	22
6.8	adafruit_seesaw.neopixel	23
6.9	adafruit_seesaw.pwmout	24
6.10	adafruit_seesaw.robohat - Pin definition for RoboHAT	24
6.11	adafruit_seesaw.samd09 - Pin definition for Adafruit SAMD09 Breakout with seesaw	25
6.12	adafruit_seesaw.tftshield18 - Pin definitions for 1.8" TFT Shield V2	25
7	Indices and tables	27
	Python Module Index	29
	Index	31

CircuitPython module for use with the Adafruit ATSAMD09 seesaw.

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).

CHAPTER 2

Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally [from PyPI](#). To install for current user:

```
pip3 install adafruit-circuitpython-seesaw
```

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

```
sudo pip3 install adafruit-circuitpython-seesaw
```

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-seesaw
```


CHAPTER 3

Usage Example

See `examples/seesaw_simpletest.py` for usage example.

CHAPTER 4

Contributing

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out [this guide](#).

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/seesaw_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  # Simple seesaw test using an LED attached to Pin 15.
5  #
6  # See the seesaw Learn Guide for wiring details:
7  # https://learn.adafruit.com/adafruit-seesaw-atsamd09-breakout?view=all#circuitpython-
8  ↪wiring-and-test
9  import time
10
11 from board import SCL, SDA
12 import busio
13 from adafruit_seesaw.seesaw import Seesaw
14
15 i2c_bus = busio.I2C(SCL, SDA)
16
17 ss = Seesaw(i2c_bus)
18
19 ss.pin_mode(15, ss.OUTPUT)
20
21 while True:
22     ss.digital_write(15, True) # turn the LED on (True is the voltage level)
23     time.sleep(1) # wait for a second
24     ss.digital_write(15, False) # turn the LED off by making the voltage LOW
25     time.sleep(1)
```

6.2 Other Examples

Here are some other examples using the Seesaw library

Listing 2: examples/seesaw_crickit_test.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  from board import SCL, SDA
5  import busio
6  from adafruit_motor import servo
7  from adafruit_seesaw.seesaw import Seesaw
8  from adafruit_seesaw.pwmout import PWMOut
9
10 # from analogio import AnalogOut
11 # import board
12
13 i2c_bus = busio.I2C(SCL, SDA)
14 ss = Seesaw(i2c_bus)
15 pwm1 = PWMOut(ss, 17)
16 pwm2 = PWMOut(ss, 16)
17 pwm3 = PWMOut(ss, 15)
18 pwm4 = PWMOut(ss, 14)
19
20 pwm1.frequency = 50
21 pwm2.frequency = 50
22 pwm3.frequency = 50
23 pwm4.frequency = 50
24
25 S1 = servo.Servo(pwm1)
26 S2 = servo.Servo(pwm2)
27 S3 = servo.Servo(pwm3)
28 S4 = servo.Servo(pwm4)
29
30 servos = (S1, S2, S3, S4)
31
32 CRCKIT_NUM_ADC = 8
33 CRCKit_adc = (2, 3, 40, 41, 11, 10, 9, 8)
34
35 CRCKIT_NUM_DRIVE = 4
36 CRCKit_drive = (42, 43, 12, 13)
37
38 CAPTOUCH_THRESH = 500
39
40 _CRCKIT_M1_A1 = 18
41 _CRCKIT_M1_A2 = 19
42 _CRCKIT_M1_B1 = 22
43 _CRCKIT_M1_B2 = 23
44
45 cap_state = [False, False, False, False]
46 cap_justtouched = [False, False, False, False]
47 cap_justreleased = [False, False, False, False]
48
49 motor1_dir = False
50 motor2_dir = True
51
```

(continues on next page)

(continued from previous page)

```

52 test_servos = False
53 test_motors = False
54 test_drives = False
55 test_speaker = False
56
57 counter = 0
58
59 # analog_out = AnalogOut(board.A0)
60 # analog_out.value = 512
61
62 while True:
63     counter = (counter + 1) % 256
64
65     if counter % 32 == 0:
66         print("----- analog -----")
67         str_out = ""
68         for i in range(8):
69             val = ss.analog_read(CRCKit_adc[i]) * 3.3 / 1024
70             str_out = str_out + str(round(val, 2)) + "\t"
71
72         print(str_out + "\n")
73
74     for i in range(4):
75         val = ss.touch_read(i)
76         cap_justtouched[i] = False
77         cap_justreleased[i] = False
78
79         if val > CAPTOUCH_THRESH:
80             print("CT" + str(i + 1) + " touched! value: " + str(val))
81
82             if not cap_state[i]:
83                 cap_justtouched[i] = True
84
85                 cap_state[i] = True
86
87         else:
88             if cap_state[i]:
89                 cap_justreleased[i] = True
90
91                 cap_state[i] = False
92
93     if cap_justtouched[0]:
94         test_servos = not test_servos
95         if test_servos:
96             print("Testing servos")
97         else:
98             print("Stopping servos")
99
100     if cap_justtouched[1]:
101         test_drives = not test_drives
102         if test_drives:
103             print("Testing drives")
104         else:
105             print("Stopping drives")
106
107     if cap_justtouched[2]:
108         test_motors = not test_motors

```

(continues on next page)

(continued from previous page)

```

109     if test_motors:
110         print("Testing motors")
111     else:
112         print("Stopping motors")
113
114     if cap_justtouched[3]:
115         test_speaker = not test_speaker
116         if test_speaker:
117             print("Testing speaker")
118         else:
119             print("Stopping speaker")
120
121     if test_servos:
122         if counter % 32 == 0:
123             print("----- servos -----")
124             servonum = int(counter / 32) % 4
125
126             if counter < 128:
127                 print("SER" + str(servonum) + " LEFT")
128                 servos[servonum].angle = 0
129             else:
130                 print("SER" + str(servonum) + " RIGHT")
131                 servos[servonum].angle = 180
132
133     if test_drives:
134         if counter % 32 == 0:
135             print("----- drives -----")
136             drivenum = int(counter / 64) % 4
137
138             if counter % 64 == 0:
139                 print("DRIVE" + str(drivenum) + " ON")
140                 ss.analog_write(CRCKIT_drive[drivenum], 65535)
141
142             else:
143                 print("DRIVE" + str(drivenum) + " OFF")
144                 ss.analog_write(CRCKIT_drive[drivenum], 0)
145
146     if test_motors:
147         if counter < 128:
148             if motor1_dir:
149                 ss.analog_write(_CRCKIT_M1_A1, 0)
150                 ss.analog_write(_CRCKIT_M1_A2, counter * 512)
151             else:
152                 ss.analog_write(_CRCKIT_M1_A2, 0)
153                 ss.analog_write(_CRCKIT_M1_A1, counter * 512)
154         else:
155             if motor1_dir:
156                 ss.analog_write(_CRCKIT_M1_A1, 0)
157                 ss.analog_write(_CRCKIT_M1_A2, (255 - counter) * 512)
158             else:
159                 ss.analog_write(_CRCKIT_M1_A2, 0)
160                 ss.analog_write(_CRCKIT_M1_A1, (255 - counter) * 512)
161         if counter == 255:
162             print("----- motor 1 -----")
163             motor1_dir = not motor1_dir
164
165     if counter < 128:

```

(continues on next page)

(continued from previous page)

```

166         if motor2_dir:
167             ss.analog_write(_CRCKIT_M1_B1, 0)
168             ss.analog_write(_CRCKIT_M1_B2, counter * 512)
169         else:
170             ss.analog_write(_CRCKIT_M1_B2, 0)
171             ss.analog_write(_CRCKIT_M1_B1, counter * 512)
172     else:
173         if motor2_dir:
174             ss.analog_write(_CRCKIT_M1_B1, 0)
175             ss.analog_write(_CRCKIT_M1_B2, (255 - counter) * 512)
176         else:
177             ss.analog_write(_CRCKIT_M1_B2, 0)
178             ss.analog_write(_CRCKIT_M1_B1, (255 - counter) * 512)
179     if counter == 255:
180         print("----- motor 2 -----")
181         motor2_dir = not motor2_dir

```

Listing 3: examples/seesaw_joy_featherwing.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5
6  from board import SCL, SDA
7  import busio
8  from micropython import const
9
10 from adafruit_seesaw.seesaw import Seesaw
11
12 BUTTON_RIGHT = const(6)
13 BUTTON_DOWN = const(7)
14 BUTTON_LEFT = const(9)
15 BUTTON_UP = const(10)
16 BUTTON_SEL = const(14)
17 button_mask = const(
18     (1 << BUTTON_RIGHT)
19     | (1 << BUTTON_DOWN)
20     | (1 << BUTTON_LEFT)
21     | (1 << BUTTON_UP)
22     | (1 << BUTTON_SEL)
23 )
24
25 i2c_bus = busio.I2C(SCL, SDA)
26
27 ss = Seesaw(i2c_bus)
28
29 ss.pin_mode_bulk(button_mask, ss.INPUT_PULLUP)
30
31 last_x = 0
32 last_y = 0
33
34 while True:
35     x = ss.analog_read(2)
36     y = ss.analog_read(3)
37

```

(continues on next page)

(continued from previous page)

```

38     if (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
39         print(x, y)
40         last_x = x
41         last_y = y
42
43     buttons = ss.digital_read_bulk(button_mask)
44     if not buttons & (1 << BUTTON_RIGHT):
45         print("Button A pressed")
46
47     if not buttons & (1 << BUTTON_DOWN):
48         print("Button B pressed")
49
50     if not buttons & (1 << BUTTON_LEFT):
51         print("Button Y pressed")
52
53     if not buttons & (1 << BUTTON_UP):
54         print("Button x pressed")
55
56     if not buttons & (1 << BUTTON_SEL):
57         print("Button SEL pressed")
58
59     time.sleep(0.01)

```

Listing 4: examples/seesaw_soil_simpletest.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5
6  from board import SCL, SDA
7  import busio
8
9  from adafruit_seesaw.seesaw import Seesaw
10
11  i2c_bus = busio.I2C(SCL, SDA)
12
13  ss = Seesaw(i2c_bus, addr=0x36)
14
15  while True:
16      # read moisture level through capacitive touch pad
17      touch = ss.moisture_read()
18
19      # read temperature from the temperature sensor
20      temp = ss.get_temp()
21
22      print("temp: " + str(temp) + "  moisture: " + str(touch))
23      time.sleep(1)

```

Listing 5: examples/seesaw_minift_featherwing.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import time
5

```

(continues on next page)

(continued from previous page)

```
6 import board
7 from micropython import const
8
9 from adafruit_seesaw.seesaw import Seesaw
10
11 BUTTON_RIGHT = const(7)
12 BUTTON_DOWN = const(4)
13 BUTTON_LEFT = const(3)
14 BUTTON_UP = const(2)
15 BUTTON_SEL = const(11)
16 BUTTON_A = const(10)
17 BUTTON_B = const(9)
18
19 button_mask = const(
20     (1 << BUTTON_RIGHT)
21     | (1 << BUTTON_DOWN)
22     | (1 << BUTTON_LEFT)
23     | (1 << BUTTON_UP)
24     | (1 << BUTTON_SEL)
25     | (1 << BUTTON_A)
26     | (1 << BUTTON_B)
27 )
28
29 i2c_bus = board.I2C()
30
31 ss = Seesaw(i2c_bus, 0x5E)
32
33 ss.pin_mode_bulk(button_mask, ss.INPUT_PULLUP)
34
35 while True:
36     buttons = ss.digital_read_bulk(button_mask)
37     if not buttons & (1 << BUTTON_RIGHT):
38         print("Button RIGHT pressed")
39
40     if not buttons & (1 << BUTTON_DOWN):
41         print("Button DOWN pressed")
42
43     if not buttons & (1 << BUTTON_LEFT):
44         print("Button LEFT pressed")
45
46     if not buttons & (1 << BUTTON_UP):
47         print("Button UP pressed")
48
49     if not buttons & (1 << BUTTON_SEL):
50         print("Button SEL pressed")
51
52     if not buttons & (1 << BUTTON_A):
53         print("Button A pressed")
54
55     if not buttons & (1 << BUTTON_B):
56         print("Button B pressed")
57
58     time.sleep(0.01)
```

6.3 adafruit_seesaw.seesaw

An I2C to whatever helper chip.

- Author(s): Dean Miller

6.3.1 Implementation Notes

Hardware:

- Adafruit [ATSAMD09 Breakout with seesaw](#) (Product ID: 3657)

Software and Dependencies:

- Adafruit CircuitPython firmware: <https://circuitpython.org/>
- or Adafruit Blinka: <https://circuitpython.org/blinka>
- Adafruit's Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

class `adafruit_seesaw.seesaw.Seesaw` (*i2c_bus*, *addr*=73, *drdy*=None)
Driver for Seesaw i2c generic conversion trip

Parameters

- **i2c_bus** (*I2C*) – Bus the SeeSaw is connected to
- **addr** (*int*) – I2C address of the SeeSaw device
- **drdy** (*DigitalInOut*) – Pin connected to SeeSaw's 'ready' output

analog_read (*pin*)

Read the value of an analog pin by number

analog_write (*pin*, *value*)

Set the value of an analog output by number

digital_read (*pin*)

Get the value of an input pin by number

digital_read_bulk (*pins*, *delay*=0.008)

Get the values of all the pins on the 'A' port as a bitmask

digital_read_bulk_b (*pins*, *delay*=0.008)

Get the values of all the pins on the 'B' port as a bitmask

digital_write (*pin*, *value*)

Set the value of an output pin by number

digital_write_bulk (*pins*, *value*)

Set the mode of pins on the 'A' port as a bitmask

digital_write_bulk_b (*pins*, *value*)

Set the mode of pins on the 'B' port as a bitmask

eeeprom_read8 (*addr*)

Read a single byte directly to the device's EEPROM

eeeprom_write (*addr*, *buf*)

Write multiple bytes directly to the device's EEPROM

eeeprom_write8 (*addr*, *val*)

Write a single byte directly to the device's EEPROM

get_i2c_addr()
Return the device's I2C address stored in its EEPROM

get_options()
Retrieve the 'options' word from the SeeSaw board

get_temp()
Read the temperature

get_version()
Retrieve the 'version' word from the SeeSaw board

moisture_read()
Read the value of the moisture sensor

pin_mode(pin, mode)
Set the mode of a pin by number

pin_mode_bulk(pins, mode)
Set the mode of all the pins on the 'A' port as a bitmask

pin_mode_bulk_b(pins, mode)
Set the mode of all the pins on the 'B' port as a bitmask

read(reg_base, reg, buf, delay=0.008)
Read an arbitrary I2C register range on the device

read8(reg_base, reg)
Read an arbitrary I2C byte register on the device

set_GPIO_interrupts(pins, enabled)
Enable or disable the GPIO interrupt

set_i2c_addr(addr)
Store a new address in the device's EEPROM and reboot it.

set_pwm_freq(pin, freq)
Set the PWM frequency of a pin by number

sw_reset()
Trigger a software reset of the SeeSaw chip

touch_read(pin)
Read the value of a touch pin by number

uart_set_baud(baud)
Set the serial baudrate of the device

write(reg_base, reg, buf=None)
Write an arbitrary I2C register range on the device

write8(reg_base, reg, value)
Write an arbitrary I2C byte register on the device

6.4 adafruit_seesaw.crickit - Pin definition for Adafruit CRICKIT

6.5 adafruit_seesaw.analoginput

class adafruit_seesaw.analoginput.**AnalogInput**(seesaw, pin)
CircuitPython-compatible class for analog inputs

This class is intended to be a compatible subset of `analogio.AnalogIn`

Parameters

- **seesaw** (*Seesaw*) – The device
- **pin** (*int*) – The pin number on the device

reference_voltage

The reference voltage for the pin

value

The current analog value on the pin, as an integer from 0..65535 (inclusive)

6.6 adafruit_seesaw.digitalio

class `adafruit_seesaw.digitalio.DigitalIO` (*seesaw*, *pin*)

CircuitPython-compatible class for digital I/O pins

This class is intended to be a compatible subset of `digitalio.DigitalInOut`.

Due to technical limitations, PULL_DOWNs are not supported.

Parameters

- **seesaw** (*Seesaw*) – The device
- **pin** (*int*) – The pin number on the device

direction

Retrieve or set the direction of the pin

drive_mode

Retrieve or set the drive mode of an output pin

pull

Retrieve or set the pull mode of an input pin

switch_to_input

 (*pull=None*)

Switch the pin to input mode

switch_to_output

 (*value=False*, *drive_mode=<sphinx.ext.autodoc.importer._MockObject object>*)

Switch the pin to output mode

value

Retrieve or set the value of the pin

6.7 adafruit_seesaw.keypad

class `adafruit_seesaw.keypad.KeyEvent` (*num*, *edge*)

Holds information about a key event in its properties

Parameters

- **num** (*int*) – The number of the key
- **edge** (*int*) – One of the EDGE properties of `adafruit_seesaw.keypad.Keypad`

class `adafruit_seesaw.keypad.Keypad` (*i2c_bus*, *addr=73*, *drdy=None*)

On compatible SeeSaw devices, reads from a keypad.

Parameters

- **i2c_bus** (*I2C*) – Bus the SeeSaw is connected to
- **addr** (*int*) – I2C address of the SeeSaw device
- **drdy** (*DigitalInOut*) – Pin connected to SeeSaw’s ‘ready’ output

EDGE_FALLING = 2

Indicates that the key was recently pressed

EDGE_HIGH = 0

Indicates that the key is currently pressed

EDGE_LOW = 1

Indicates that the key is currently released

EDGE_RISING = 3

Indicates that the key was recently released

count

Retrieve or set the number of keys

interrupt_enabled

Retrieve or set the interrupt enable flag

read_keypad (*num*)

Read data from the keypad

Parameters **num** (*int*) – The number of bytes to read**set_event** (*key, edge, enable*)

Control which kinds of events are set

Parameters

- **key** (*int*) – The key number
- **edge** (*int*) – The type of event
- **enable** (*bool*) – True to enable the event, False to disable it

6.8 adafruit_seesaw.neopixel

adafruit_seesaw.neopixel.GRB = (1, 0, 2)

Green Red Blue

adafruit_seesaw.neopixel.GRBW = (1, 0, 2, 3)

Green Red Blue White

class **adafruit_seesaw.neopixel.NeoPixel** (*seesaw, pin, n, *, bpp=3, brightness=1.0,*
auto_write=True, pixel_order=None)

Control NeoPixels connected to a seesaw

Parameters

- **seesaw** (*Seesaw*) – The device
- **pin** (*int*) – The pin number on the device
- **n** (*int*) – The number of pixels
- **bpp** (*int*) – The number of bytes per pixel

- **brightness** (*float*) – The brightness, from 0.0 to 1.0
- **auto_write** (*bool*) – Automatically update the pixels when changed
- **pixel_order** (*tuple*) – The layout of the pixels. Use one of the order constants such as RGBW.

__getitem__ (*key, color*)

Set one pixel to a new value

brightness

Overall brightness of the pixel

fill (*color*)

Set all pixels to the same value

show ()

Update the pixels even if auto_write is False

`adafruit_seesaw.neopixel.RGB = (0, 1, 2)`

Red Green Blue

`adafruit_seesaw.neopixel.RGBW = (0, 1, 2, 3)`

Red Green Blue White

6.9 adafruit_seesaw.pwmout

class `adafruit_seesaw.pwmout.PWMOut` (*seesaw, pin*)

A single seesaw channel that matches the `PWMOut` API.

duty_cycle

16-bit value that dictates how much of one cycle is high (1) versus low (0). 65535 (0xffff) will always be high, 0 will always be low, and 32767 (0x7fff) will be half high and then half low.

fraction

Expresses duty_cycle as a fractional value. Ranges from 0.0-1.0.

frequency

The overall PWM frequency in Hertz.

6.10 adafruit_seesaw.robocat - Pin definition for RoboHAT

class `adafruit_seesaw.robocat.MM1_Pinmap`

This class is automatically used by `adafruit_seesaw.seesaw.Seesaw` when a RoboHAT board is detected.

It is also a reference for the capabilities of each pin.

analog_pins = (35, 34)

The pins capable of analog output

pwm_pins = (16, 17, 18, 19, 11, 10, 9, 8, 40, 41, 42, 43)

The pins capable of PWM output

pwm_width = 16

The effective bit resolution of the PWM pins

touch_pins = (7, 6, 5, 4)

The pins capable of touch input

6.11 `adafruit_seesaw.samd09` - Pin definition for Adafruit SAMD09 Breakout with seesaw

class `adafruit_seesaw.samd09.SAMD09_Pinmap`

This class is automatically used by `adafruit_seesaw.seesaw.Seesaw` when a SAMD09 Breakout is detected.

It is also a reference for the capabilities of each pin.

analog_pins = (2, 3, 4, 5)

The effective bit resolution of the PWM pins

pwm_pins = (4, 5, 6, 7)

No pins on this board are capable of touch input

pwm_width = 8

The pins capable of PWM output

6.12 `adafruit_seesaw.tftshield18` - Pin definitions for 1.8" TFT Shield V2

class `adafruit_seesaw.tftshield18.Buttons` (*right, down, left, up, select, a, b, c*)

static `__new__` (*_cls, right, down, left, up, select, a, b, c*)

Create new instance of Buttons(right, down, left, up, select, a, b, c)

`__repr__` ()

Return a nicely formatted representation string

a

Alias for field number 5

b

Alias for field number 6

c

Alias for field number 7

down

Alias for field number 1

left

Alias for field number 2

right

Alias for field number 0

select

Alias for field number 4

up

Alias for field number 3

class `adafruit_seesaw.tftshield18.TFTShield18` (*i2c_bus=None, addr=46*)

buttons

Return a set of buttons with current push values

set_backlight (*value*)

Set the backlight on

set_backlight_freq (*freq*)

Set the backlight frequency of the TFT Display

tft_reset (*rst=True*)

Reset the TFT Display

CHAPTER 7

Indices and tables

- `genindex`
- `modindex`
- `search`

a

- `adafruit_seesaw`, [19](#)
- `adafruit_seesaw.__init__`, [22](#)
- `adafruit_seesaw.analoginput`, [21](#)
- `adafruit_seesaw.crickit`, [21](#)
- `adafruit_seesaw.digitalio`, [22](#)
- `adafruit_seesaw.keypad`, [22](#)
- `adafruit_seesaw.neopixel`, [23](#)
- `adafruit_seesaw.pwmout`, [24](#)
- `adafruit_seesaw.robocat`, [24](#)
- `adafruit_seesaw.samd09`, [24](#)
- `adafruit_seesaw.seesaw`, [19](#)
- `adafruit_seesaw.tftshield18`, [25](#)

Symbols

`__new__()` (*adafruit_seesaw.tftshield18.Buttons* static method), 25

`__repr__()` (*adafruit_seesaw.tftshield18.Buttons* method), 25

`__setitem__()` (*adafruit_seesaw.neopixel.NeoPixel* method), 24

A

`a` (*adafruit_seesaw.tftshield18.Buttons* attribute), 25

`adafruit_seesaw` (module), 19

`adafruit_seesaw.__init__` (module), 22

`adafruit_seesaw.analoginput` (module), 21

`adafruit_seesaw.crickit` (module), 21

`adafruit_seesaw.digitalio` (module), 22

`adafruit_seesaw.keypad` (module), 22

`adafruit_seesaw.neopixel` (module), 23

`adafruit_seesaw.pwmout` (module), 24

`adafruit_seesaw.robohat` (module), 24

`adafruit_seesaw.samd09` (module), 24

`adafruit_seesaw.seesaw` (module), 19

`adafruit_seesaw.tftshield18` (module), 25

`analog_pins` (*adafruit_seesaw.robohat.MM1_Pinmap* attribute), 24

`analog_pins` (*adafruit_seesaw.samd09.SAMD09_Pinmap* attribute), 25

`analog_read()` (*adafruit_seesaw.seesaw.Seesaw* method), 20

`analog_write()` (*adafruit_seesaw.seesaw.Seesaw* method), 20

`AnalogInput` (class in *adafruit_seesaw.analoginput*), 21

B

`b` (*adafruit_seesaw.tftshield18.Buttons* attribute), 25

`brightness` (*adafruit_seesaw.neopixel.NeoPixel* attribute), 24

`buttons` (*adafruit_seesaw.tftshield18.TFTShield18* attribute), 25

`Buttons` (class in *adafruit_seesaw.tftshield18*), 25

C

`c` (*adafruit_seesaw.tftshield18.Buttons* attribute), 25

`count` (*adafruit_seesaw.keypad.Keypad* attribute), 23

D

`digital_read()` (*adafruit_seesaw.seesaw.Seesaw* method), 20

`digital_read_bulk()` (*adafruit_seesaw.seesaw.Seesaw* method), 20

`digital_read_bulk_b()` (*adafruit_seesaw.seesaw.Seesaw* method), 20

`digital_write()` (*adafruit_seesaw.seesaw.Seesaw* method), 20

`digital_write_bulk()` (*adafruit_seesaw.seesaw.Seesaw* method), 20

`digital_write_bulk_b()` (*adafruit_seesaw.seesaw.Seesaw* method), 20

`DigitalIO` (class in *adafruit_seesaw.digitalio*), 22

`direction` (*adafruit_seesaw.digitalio.DigitalIO* attribute), 22

`down` (*adafruit_seesaw.tftshield18.Buttons* attribute), 25

`drive_mode` (*adafruit_seesaw.digitalio.DigitalIO* attribute), 22

`duty_cycle` (*adafruit_seesaw.pwmout.PWMOut* attribute), 24

E

`EDGE_FALLING` (*adafruit_seesaw.keypad.Keypad* attribute), 23

`EDGE_HIGH` (*adafruit_seesaw.keypad.Keypad* attribute), 23

`EDGE_LOW` (*adafruit_seesaw.keypad.Keypad* attribute), 23

EDGE_RISING (*adafruit_seesaw.keypad.Keypad* attribute), 23
 eeprom_read8() (*adafruit_seesaw.seesaw.Seesaw* method), 20
 eeprom_write() (*adafruit_seesaw.seesaw.Seesaw* method), 20
 eeprom_write8() (*adafruit_seesaw.seesaw.Seesaw* method), 20

F

fill() (*adafruit_seesaw.neopixel.NeoPixel* method), 24
 fraction (*adafruit_seesaw.pwmout.PWMOut* attribute), 24
 frequency (*adafruit_seesaw.pwmout.PWMOut* attribute), 24

G

get_i2c_addr() (*adafruit_seesaw.seesaw.Seesaw* method), 20
 get_options() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 get_temp() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 get_version() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 GRB (in module *adafruit_seesaw.neopixel*), 23
 GRBW (in module *adafruit_seesaw.neopixel*), 23

I

interrupt_enabled (*adafruit_seesaw.keypad.Keypad* attribute), 23

K

KeyEvent (class in *adafruit_seesaw.keypad*), 22
 Keypad (class in *adafruit_seesaw.keypad*), 22

L

left (*adafruit_seesaw.tftshield18.Buttons* attribute), 25

M

MM1_Pinmap (class in *adafruit_seesaw.robohat*), 24
 moisture_read() (*adafruit_seesaw.seesaw.Seesaw* method), 21

N

NeoPixel (class in *adafruit_seesaw.neopixel*), 23

P

pin_mode() (*adafruit_seesaw.seesaw.Seesaw* method), 21

pin_mode_bulk() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 pin_mode_bulk_b() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 pull (*adafruit_seesaw.digitalio.DigitalIO* attribute), 22
 pwm_pins (*adafruit_seesaw.robohat.MM1_Pinmap* attribute), 24
 pwm_pins (*adafruit_seesaw.samd09.SAMD09_Pinmap* attribute), 25
 pwm_width (*adafruit_seesaw.robohat.MM1_Pinmap* attribute), 24
 pwm_width (*adafruit_seesaw.samd09.SAMD09_Pinmap* attribute), 25
 PWMOut (class in *adafruit_seesaw.pwmout*), 24

R

read() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 read8() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 read_keypad() (*adafruit_seesaw.keypad.Keypad* method), 23
 reference_voltage (*adafruit_seesaw.analoginput.AnalogInput* attribute), 22
 RGB (in module *adafruit_seesaw.neopixel*), 24
 RGBW (in module *adafruit_seesaw.neopixel*), 24
 right (*adafruit_seesaw.tftshield18.Buttons* attribute), 25

S

SAMD09_Pinmap (class in *adafruit_seesaw.samd09*), 25
 Seesaw (class in *adafruit_seesaw.seesaw*), 20
 select (*adafruit_seesaw.tftshield18.Buttons* attribute), 25
 set_backlight() (*adafruit_seesaw.tftshield18.TFTShield18* method), 26
 set_backlight_freq() (*adafruit_seesaw.tftshield18.TFTShield18* method), 26
 set_event() (*adafruit_seesaw.keypad.Keypad* method), 23
 set_GPIO_interrupts() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 set_i2c_addr() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 set_pwm_freq() (*adafruit_seesaw.seesaw.Seesaw* method), 21
 show() (*adafruit_seesaw.neopixel.NeoPixel* method), 24
 sw_reset() (*adafruit_seesaw.seesaw.Seesaw* method), 21

```
switch_to_input()
    (adafruit_seesaw.digitalio.DigitalIO method),
    22
switch_to_output()
    (adafruit_seesaw.digitalio.DigitalIO method),
    22
```

T

```
tft_reset() (adafruit_seesaw.tftshield18.TFTShield18
method), 26
TFTShield18 (class in adafruit_seesaw.tftshield18),
    25
touch_pins (adafruit_seesaw.robocat.MMI_Pinmap
attribute), 24
touch_read() (adafruit_seesaw.seesaw.Seesaw
method), 21
```

U

```
uart_set_baud() (adafruit_seesaw.seesaw.Seesaw
method), 21
up (adafruit_seesaw.tftshield18.Buttons attribute), 25
```

V

```
value (adafruit_seesaw.analoginput.AnalogInput at-
tribute), 22
value (adafruit_seesaw.digitalio.DigitalIO attribute),
    22
```

W

```
write() (adafruit_seesaw.seesaw.Seesaw method), 21
write8() (adafruit_seesaw.seesaw.Seesaw method), 21
```