
Adafruit SEESAW Library Documentation

Release 1.0

Dean Miller

Feb 13, 2020

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

6.2 Other Examples

Here are some other examples using the Seesaw library

Listing 2: examples/seesaw_crickit_test.py

```

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

```

(continues on next page)

(continued from previous page)

```

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

```

(continues on next page)

(continued from previous page)

```

109     else:
110         print("Stopping motors")
111
112     if cap_justtouched[3]:
113         test_speaker = not test_speaker
114         if test_speaker:
115             print("Testing speaker")
116         else:
117             print("Stopping speaker")
118
119
120     if test_servos:
121         if counter % 32 == 0:
122             print("----- servos -----")
123             servonum = int(counter / 32) % 4
124
125             if counter < 128:
126                 print("SER" + str(servonum) + " LEFT")
127                 servos[servonum].angle = 0
128             else:
129                 print("SER" + str(servonum) + " RIGHT")
130                 servos[servonum].angle = 180
131
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  import time
2
3  from board import SCL, SDA
4  import busio
5  from micropython import const
6
7  from adafruit_seesaw.seesaw import Seesaw
8
9  # pylint: disable=bad-whitespace
10 BUTTON_RIGHT = const(6)
11 BUTTON_DOWN  = const(7)
12 BUTTON_LEFT  = const(9)
13 BUTTON_UP    = const(10)
14 BUTTON_SEL   = const(14)
15 # pylint: enable=bad-whitespace
16 button_mask = const((1 << BUTTON_RIGHT) |
17                     (1 << BUTTON_DOWN) |
18                     (1 << BUTTON_LEFT) |
19                     (1 << BUTTON_UP) |
20                     (1 << BUTTON_SEL))
21
22 i2c_bus = busio.I2C(SCL, SDA)
23
24 ss = Seesaw(i2c_bus)
25
26 ss.pin_mode_bulk(button_mask, ss.INPUT_PULLUP)
27
28 last_x = 0
29 last_y = 0
30
31 while True:
32     x = ss.analog_read(2)
33     y = ss.analog_read(3)
34
35     if (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
36         print(x, y)
37         last_x = x

```

(continues on next page)

(continued from previous page)

```

38     last_y = y
39
40     buttons = ss.digital_read_bulk(button_mask)
41     if not buttons & (1 << BUTTON_RIGHT):
42         print("Button A pressed")
43
44     if not buttons & (1 << BUTTON_DOWN):
45         print("Button B pressed")
46
47     if not buttons & (1 << BUTTON_LEFT):
48         print("Button Y pressed")
49
50     if not buttons & (1 << BUTTON_UP):
51         print("Button x pressed")
52
53     if not buttons & (1 << BUTTON_SEL):
54         print("Button SEL pressed")
55
56     time.sleep(.01)

```

Listing 4: examples/seesaw_soil_simpletest.py

```

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

```

Listing 5: examples/seesaw_minift_featherwing.py

```

1  import time
2
3  import board
4  from micropython import const
5
6  from adafruit_seesaw.seesaw import Seesaw
7
8  # pylint: disable=bad-whitespace
9  BUTTON_RIGHT = const(7)
10 BUTTON_DOWN  = const(4)
11 BUTTON_LEFT  = const(3)

```

(continues on next page)

(continued from previous page)

```
12 BUTTON_UP    = const(2)
13 BUTTON_SEL   = const(11)
14 BUTTON_A     = const(10)
15 BUTTON_B     = const(9)
16
17 # pylint: enable=bad-whitespace
18 button_mask = const((1 << BUTTON_RIGHT) |
19                    (1 << BUTTON_DOWN) |
20                    (1 << BUTTON_LEFT) |
21                    (1 << BUTTON_UP) |
22                    (1 << BUTTON_SEL) |
23                    (1 << BUTTON_A) |
24                    (1 << BUTTON_B))
25
26 i2c_bus = board.I2C()
27
28 ss = Seesaw(i2c_bus, 0x5E)
29
30 ss.pin_mode_bulk(button_mask, ss.INPUT_PULLUP)
31
32 while True:
33     buttons = ss.digital_read_bulk(button_mask)
34     if not buttons & (1 << BUTTON_RIGHT):
35         print("Button RIGHT pressed")
36
37     if not buttons & (1 << BUTTON_DOWN):
38         print("Button DOWN pressed")
39
40     if not buttons & (1 << BUTTON_LEFT):
41         print("Button LEFT pressed")
42
43     if not buttons & (1 << BUTTON_UP):
44         print("Button UP pressed")
45
46     if not buttons & (1 << BUTTON_SEL):
47         print("Button SEL pressed")
48
49     if not buttons & (1 << BUTTON_A):
50         print("Button A pressed")
51
52     if not buttons & (1 << BUTTON_B):
53         print("Button B pressed")
54
55     time.sleep(.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.robohat - Pin definition for RoboHAT

class adafruit_seesaw.robohat.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=<sphinx.ext.autodoc.importer._MockObject object>, 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.robohat](#), 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), 23

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.MMI_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), 23

`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), 22

`EDGE_HIGH` (*adafruit_seesaw.keypad.Keypad* attribute), 22

`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*), 20
 get_temp() (*adafruit_seesaw.seesaw.Seesaw method*), 20
 get_version() (*adafruit_seesaw.seesaw.Seesaw method*), 20
 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*), 20

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*), 21
 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*), 24
 Seesaw (*class in adafruit_seesaw.seesaw*), 20
 select (*adafruit_seesaw.tftshield18.Buttons attribute*), 25
 set_backlight() (*adafruit_seesaw.tftshield18.TFTShield18 method*), 25
 set_backlight_freq() (*adafruit_seesaw.tftshield18.TFTShield18 method*), 25
 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), 25
 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 attribute), 21
 value (adafruit_seesaw.digitalio.DigitalIO attribute), 22

W

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