

---

# **Adafruit SEESAW Library Documentation**

*Release 1.0*

**Dean Miller**

**Aug 21, 2019**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Building locally</b>	<b>9</b>
4.1	Sphinx documentation . . . . .	9
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	Other Examples . . . . .	12
5.3	seesaw . . . . .	17
5.3.1	Implementation Notes . . . . .	17
<b>6</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



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

---

### Usage Example

---

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



## CHAPTER 3

---

### Contributing

---

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



## CHAPTER 4

---

### Building locally

---

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

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

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

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

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-seesaw --library_
↳location .
```

### 4.1 Sphinx documentation

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

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

Now, once you have the virtual environment activated:

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

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



## 5.1 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)
```

## 5.2 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)

```

## 5.3 seesaw

An I2C to whatever helper chip.

- Author(s): Dean Miller

### 5.3.1 Implementation Notes

**Hardware:**

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

### Software and Dependencies:

- Adafruit CircuitPython firmware for the ESP8622 and M0-based boards: <https://github.com/adafruit/circuitpython/releases>
- Adafruit's Bus Device library: [https://github.com/adafruit/Adafruit\\_CircuitPython\\_BusDevice](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

**`sw_reset`** ()  
Trigger a software reset of the SeeSaw chip

## CHAPTER 6

---

### Indices and tables

---

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





**a**

`adafruit_seesaw.seesaw`, 17



## A

`adafruit_seesaw.seesaw` (*module*), 17

## S

`Seesaw` (*class in `adafruit_seesaw.seesaw`*), 18

`sw_reset()` (*`adafruit_seesaw.seesaw.Seesaw` method*),  
18