Adafruit SEESAW Library Documentation

Release 1.0

Dean Miller

Aug 21, 2019

Contents

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.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/seesaw_simpletest.py

```
# Simple seesaw test using an LED attached to Pin 15.
#
# See the seesaw Learn Guide for wiring details:
# https://learn.adafruit.com/adafruit-seesaw-atsamd09-breakout?view=all#circuitpython-wiring-and-test
import time

from board import SCL, SDA
import busio
from adafruit_seesaw.seesaw import Seesaw

i2c_bus = busio.I2C(SCL, SDA)

ss = Seesaw(i2c_bus)

ss.pin_mode(15, ss.OUTPUT)

while True:
    ss.digital_write(15, True)    # turn the LED on (True is the voltage level)
    time.sleep(1)                 # wait for a second
    ss.digital_write(15, False)   # turn the LED off by making the voltage LOW
    time.sleep(1)
```

5.2 Other Examples

Here are some other examples using the Seesaw library

Listing 2: examples/seesaw_crickit_test.py

```
from board import SCL, SDA
import busio
from adafruit_seesaw.seesaw import Seesaw
from adafruit_seesaw.pwmout import PWMOut
from adafruit_motor import servo

#from analogio import AnalogOut
#import board

i2c_bus = busio.I2C(SCL, SDA)
ss = Seesaw(i2c_bus)
pwm1 = PWMOut(ss, 17)
pwm2 = PWMOut(ss, 16)
pwm3 = PWMOut(ss, 15)
pwm4 = PWMOut(ss, 14)

pwm1.frequency = 50
pwm2.frequency = 50
pwm3.frequency = 50
pwm4.frequency = 50

S1 = servo.Servo(pwm1)
S2 = servo.Servo(pwm2)
S3 = servo.Servo(pwm3)
S4 = servo.Servo(pwm4)

servos = (S1, S2, S3, S4)

CRCKIT_NUM_ADC = 8
CRCKit_adc = (2, 3, 40, 41, 11, 10, 9, 8)

CRCKIT_NUM_DRIVE = 4
CRCKit_drive = (42, 43, 12, 13)

CAPTOUCH_THRESH = 500

_CRCKIT_M1_A1 = 18
_CRCKIT_M1_A2 = 19
_CRCKIT_M1_B1 = 22
_CRCKIT_M1_B2 = 23

cap_state = [False, False, False, False]
cap_justtouched = [False, False, False, False]
cap_justreleased = [False, False, False, False]

motor1_dir = False
motor2_dir = True

test_servos = False
test_motors = False
test_drives = False
```

(continues on next page)

(continued from previous page)

```
test_speaker = False

counter = 0

#analog_out = AnalogOut(board.A0)
#analog_out.value = 512

while True:
    counter = (counter + 1) % 256

    if counter % 32 == 0:
        print("-------------------- analog ------------------------")
        str_out = ""
        for i in range(8):
            val = ss.analog_read(CRCKit_adc[i]) * 3.3/1024
            str_out = str_out + str(round(val, 2)) + "\t"

        print(str_out + "\n")


    for i in range(4):
        val = ss.touch_read(i)
        cap_justtouched[i] = False
        cap_justreleased[i] = False

        if val > CAPTOUCH_THRESH:
            print("CT" + str(i + 1) + " touched! value: " + str(val))

            if not cap_state[i]:
                cap_justtouched[i] = True

            cap_state[i] = True

        else:
            if cap_state[i]:
                cap_justreleased[i] = True

            cap_state[i] = False

    if cap_justtouched[0]:
        test_servos = not test_servos
        if test_servos:
            print("Testing servos")
        else:
            print("Stopping servos")

    if cap_justtouched[1]:
        test_drives = not test_drives
        if test_drives:
            print("Testing drives")
        else:
            print("Stopping drives")

    if cap_justtouched[2]:
        test_motors = not test_motors
        if test_motors:
            print("Testing motors")
```

(continues on next page)

(continued from previous page)

```
        else:
            print("Stopping motors")

    if cap_justtouched[3]:
        test_speaker = not test_speaker
        if test_speaker:
            print("Testing speaker")
        else:
            print("Stopping speaker")


    if test_servos:
        if counter % 32 == 0:
            print("-------------------- servos ------------------------")
            servonum = int(counter / 32) % 4

            if counter < 128:
                print("SER" + str(servonum) + " LEFT")
                servos[servonum].angle = 0
            else:
                print("SER" + str(servonum) + " RIGHT")
                servos[servonum].angle = 180


    if test_drives:
        if counter % 32 == 0:
            print("-------------------- drives ------------------------")
            drivenum = int(counter / 64) % 4

            if counter % 64 == 0:
                print("DRIVE" + str(drivenum) + " ON")
                ss.analog_write(CRCKit_drive[drivenum], 65535)

            else:
                print("DRIVE" + str(drivenum) + " OFF")
                ss.analog_write(CRCKit_drive[drivenum], 0)

    if test_motors:
        if counter < 128:
            if motor1_dir:
                ss.analog_write(_CRCKIT_M1_A1, 0)
                ss.analog_write(_CRCKIT_M1_A2, counter * 512)
            else:
                ss.analog_write(_CRCKIT_M1_A2, 0)
                ss.analog_write(_CRCKIT_M1_A1, counter * 512)
        else:
            if motor1_dir:
                ss.analog_write(_CRCKIT_M1_A1, 0)
                ss.analog_write(_CRCKIT_M1_A2, (255-counter) * 512)
            else:
                ss.analog_write(_CRCKIT_M1_A2, 0)
                ss.analog_write(_CRCKIT_M1_A1, (255-counter) * 512)
        if counter == 255:
            print("-------------------- motor 1 ------------------------")
            motor1_dir = not motor1_dir

        if counter < 128:
```

(continues on next page)

(continued from previous page)

```
            if motor2_dir:
                ss.analog_write(_CRCKIT_M1_B1, 0)
                ss.analog_write(_CRCKIT_M1_B2, counter * 512)
            else:
                ss.analog_write(_CRCKIT_M1_B2, 0)
                ss.analog_write(_CRCKIT_M1_B1, counter * 512)
        else:
            if motor2_dir:
                ss.analog_write(_CRCKIT_M1_B1, 0)
                ss.analog_write(_CRCKIT_M1_B2, (255-counter) * 512)
            else:
                ss.analog_write(_CRCKIT_M1_B2, 0)
                ss.analog_write(_CRCKIT_M1_B1, (255-counter) * 512)
        if counter == 255:
            print("-------------------- motor 2 ------------------------")
            motor2_dir = not motor2_dir
```

Listing 3: examples/seesaw_joy_featherwing.py

```
import time

from board import SCL, SDA
import busio
from micropython import const

from adafruit_seesaw.seesaw import Seesaw

# pylint: disable=bad-whitespace
BUTTON_RIGHT = const(6)
BUTTON_DOWN  = const(7)
BUTTON_LEFT  = const(9)
BUTTON_UP    = const(10)
BUTTON_SEL   = const(14)
# pylint: enable=bad-whitespace
button_mask = const((1 << BUTTON_RIGHT) |
                    (1 << BUTTON_DOWN) |
                    (1 << BUTTON_LEFT) |
                    (1 << BUTTON_UP) |
                    (1 << BUTTON_SEL))

i2c_bus = busio.I2C(SCL, SDA)

ss = Seesaw(i2c_bus)

ss.pin_mode_bulk(button_mask, ss.INPUT_PULLUP)

last_x = 0
last_y = 0

while True:
    x = ss.analog_read(2)
    y = ss.analog_read(3)

    if  (abs(x - last_x) > 3) or (abs(y - last_y) > 3):
        print(x, y)
        last_x = x
```

(continues on next page)

(continued from previous page)

```
        last_y = y

    buttons = ss.digital_read_bulk(button_mask)
    if not buttons & (1 << BUTTON_RIGHT):
        print("Button A pressed")

    if not buttons & (1 << BUTTON_DOWN):
        print("Button B pressed")

    if not buttons & (1 << BUTTON_LEFT):
        print("Button Y pressed")

    if not buttons & (1 << BUTTON_UP):
        print("Button x pressed")

    if not buttons & (1 << BUTTON_SEL):
        print("Button SEL pressed")

    time.sleep(.01)
```

Listing 4: examples/seesaw_soil_simpletest.py

```
import time

from board import SCL, SDA
import busio

from adafruit_seesaw.seesaw import Seesaw

i2c_bus = busio.I2C(SCL, SDA)

ss = Seesaw(i2c_bus, addr=0x36)

while True:
    # read moisture level through capacitive touch pad
    touch = ss.moisture_read()

    # read temperature from the temperature sensor
    temp = ss.get_temp()

    print("temp: " + str(temp) + "  moisture: " + str(touch))
    time.sleep(1)
```

Listing 5: examples/seesaw_minitft_featherwing.py

```
import time

import board
from micropython import const

from adafruit_seesaw.seesaw import Seesaw

# pylint: disable=bad-whitespace
BUTTON_RIGHT = const(7)
BUTTON_DOWN  = const(4)
BUTTON_LEFT  = const(3)
```

(continues on next page)

(continued from previous page)

```
BUTTON_UP     = const(2)
BUTTON_SEL    = const(11)
BUTTON_A      = const(10)
BUTTON_B      = const(9)

# pylint: enable=bad-whitespace
button_mask = const((1 << BUTTON_RIGHT) |
                    (1 << BUTTON_DOWN) |
                    (1 << BUTTON_LEFT) |
                    (1 << BUTTON_UP) |
                    (1 << BUTTON_SEL) |
                    (1 << BUTTON_A) |
                    (1 << BUTTON_B))

i2c_bus = board.I2C()

ss = Seesaw(i2c_bus, 0x5E)

ss.pin_mode_bulk(button_mask, ss.INPUT_PULLUP)

while True:
    buttons = ss.digital_read_bulk(button_mask)
    if not buttons & (1 << BUTTON_RIGHT):
        print("Button RIGHT pressed")

    if not buttons & (1 << BUTTON_DOWN):
        print("Button DOWN pressed")

    if not buttons & (1 << BUTTON_LEFT):
        print("Button LEFT pressed")

    if not buttons & (1 << BUTTON_UP):
        print("Button UP pressed")

    if not buttons & (1 << BUTTON_SEL):
        print("Button SEL pressed")

    if not buttons & (1 << BUTTON_A):
        print("Button A pressed")

    if not buttons & (1 << BUTTON_B):
        print("Button B pressed")

    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

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

Python Module Index

a

Index

A

S