
Adafruit SimpleIO Library Documentation

Release 1.0

Scott Shawcroft

Jan 09, 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 tests	13
6.2	simpleio - Simple, beginner friendly IO.	15
7	Indices and tables	19
	Python Module Index	21
	Index	23

SimpleIO features a number of helpers to simplify hardware interactions. Many of the functions and classes are inspired by Arduino APIs to make it easier to move to CircuitPython from Arduino.

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)

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

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-simpleio
```

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

```
sudo pip3 install adafruit-circuitpython-simpleio
```

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-simpleio
```


CHAPTER 3

Usage Example

See the examples in the *Simple tests* folder for usage.

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 tests

Ensure your device works with these simple tests.

Listing 1: examples/simpleio_tone_demo.py

```
1 """
2 'tone_demo.py'.
3
4 =====
5 a short piezo song using tone()
6 """
7 import time
8 import board
9 import simpleio
10
11
12 while True:
13     for f in (262, 294, 330, 349, 392, 440, 494, 523):
14         simpleio.tone(board.A0, f, 0.25)
15     time.sleep(1)
```

Listing 2: examples/simpleio_shift_in_out_demo.py

```
1 """
2 'shift_in_out_demo.py'.
3
4 =====
5 shifts data into and out of a data pin
6 """
7
8 import time
9 import board
```

(continues on next page)

(continued from previous page)

```

10 import digitalio
11 import simpleio
12
13 # set up clock, data, and latch pins
14 clk = digitalio.DigitalInOut(board.D12)
15 data = digitalio.DigitalInOut(board.D11)
16 latch = digitalio.DigitalInOut(board.D10)
17 clk.direction = digitalio.Direction.OUTPUT
18 latch.direction = digitalio.Direction.OUTPUT
19
20 while True:
21     data_to_send = 256
22     # shifting 256 bits out of data pin
23     latch.value = False
24     data.direction = digitalio.Direction.OUTPUT
25     print('shifting out...')
26     simpleio.shift_out(data, clk, data_to_send, msb_first=False)
27     latch.value = True
28     time.sleep(3)
29
30     # shifting 256 bits into the data pin
31     latch.value = False
32     data.direction = digitalio.Direction.INPUT
33     print('shifting in...')
34     simpleio.shift_in(data, clk)
35     time.sleep(3)

```

Listing 3: examples/simpleio_map_range_simpletest.py

```

1 """
2 'map_range_demo.py'.
3
4 =====
5 maps a number from one range to another
6 """
7 import time
8 import simpleio
9
10 while True:
11     sensor_value = 150
12
13     # Map the sensor's range from 0<=sensor_value<=255 to 0<=sensor_value<=1023
14     print('original sensor value: ', sensor_value)
15     mapped_value = simpleio.map_range(sensor_value, 0, 255, 0, 1023)
16     print('mapped sensor value: ', mapped_value)
17     time.sleep(2)
18
19     # Map the new sensor value back to the old range
20     sensor_value = simpleio.map_range(mapped_value, 0, 1023, 0, 255)
21     print('original value returned: ', sensor_value)
22     time.sleep(2)

```

6.2 simpleio - Simple, beginner friendly IO.

The `simpleio` module contains classes to provide simple access to IO.

- Author(s): Scott Shawcroft

class `simpleio.DigitalIn` (*pin*, ***kwargs*)

Simple digital input that is valid until reload.

param pin microcontroller.Pin input pin

param pull digitalio.Pull pull configuration for the input

value

The digital logic level of the input pin.

class `simpleio.DigitalOut` (*pin*, ***kwargs*)

Simple digital output that is valid until reload.

param pin microcontroller.Pin output pin

param value bool default value

param drive_mode digitalio.DriveMode drive mode for the output

value

The digital logic level of the output pin.

`simpleio.bitWrite` (*x*, *n*, *b*)

Based on the Arduino `bitWrite` function, changes a specific bit of a value to 0 or 1. The return value is the original value with the changed bit. This function is written for use with 8-bit shift registers

Parameters

- **x** – numeric value
- **n** – position to change starting with least-significant (right-most) bit as 0
- **b** – value to write (0 or 1)

`simpleio.map_range` (*x*, *in_min*, *in_max*, *out_min*, *out_max*)

Maps a number from one range to another. Note: This implementation handles values < `in_min` differently than arduino's `map` function does.

Returns Returns value mapped to new range

Return type `float`

`simpleio.shift_in` (*data_pin*, *clock*, *msb_first=True*)

Shifts in a byte of data one bit at a time. Starts from either the LSB or MSB.

Warning: Data and clock are swapped compared to other CircuitPython libraries in order to match Arduino.

Parameters

- **data_pin** (*DigitalInOut*) – pin on which to input each bit
- **clock** (*DigitalInOut*) – toggles to signal `data_pin` reads
- **msb_first** (*bool*) – True when the first bit is most significant

Returns returns the value read

Return type `int`

`simpleio.shift_out` (*data_pin*, *clock*, *value*, *msb_first=True*, *bitcount=8*)

Shifts out a byte of data one bit at a time. Data gets written to a data pin. Then, the clock pulses hi then low

Warning: Data and clock are swapped compared to other CircuitPython libraries in order to match Arduino.

Parameters

- **data_pin** (*DigitalInOut*) – value bits get output on this pin
- **clock** (*DigitalInOut*) – toggled once the data pin is set
- **msb_first** (*bool*) – True when the first bit is most significant
- **value** (*int*) – byte to be shifted
- **bitcount** (*unsigned*) – number of bits to shift

Example for Metro M0 Express:

```
import digitalio
import simpleio
from board import *
clock = digitalio.DigitalInOut(D12)
data_pin = digitalio.DigitalInOut(D11)
latchPin = digitalio.DigitalInOut(D10)
clock.direction = digitalio.Direction.OUTPUT
data_pin.direction = digitalio.Direction.OUTPUT
latchPin.direction = digitalio.Direction.OUTPUT

while True:
    valueSend = 500
    # shifting out least significant bits
    # must toggle latchPin.value before and after shift_out to push to IC chip
    # this sample code was tested using
    latchPin.value = False
    simpleio.shift_out(data_pin, clock, (valueSend>>8), msb_first = False)
    latchPin.value = True
    time.sleep(1.0)
    latchPin.value = False
    simpleio.shift_out(data_pin, clock, valueSend, msb_first = False)
    latchPin.value = True
    time.sleep(1.0)

    # shifting out most significant bits
    latchPin.value = False
    simpleio.shift_out(data_pin, clock, (valueSend>>8))
    latchPin.value = True
    time.sleep(1.0)
    latchPin.value = False
    simpleio.shift_out(data_pin, clock, valueSend)
    latchPin.value = True
    time.sleep(1.0)
```

`simpleio.tone` (*pin*, *frequency*, *duration=1*, *length=100*)

Generates a square wave of the specified frequency on a pin

Parameters

- **pin** (*Pin*) – Pin on which to output the tone

- **frequency** (*float*) – Frequency of tone in Hz
- **length** (*int*) – Variable size buffer (optional)
- **duration** (*int*) – Duration of tone in seconds (optional)

CHAPTER 7

Indices and tables

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

S

simpleio, 14

B

`bitWrite()` (*in module simpleio*), 15

D

`DigitalIn` (*class in simpleio*), 15

`DigitalOut` (*class in simpleio*), 15

M

`map_range()` (*in module simpleio*), 15

S

`shift_in()` (*in module simpleio*), 15

`shift_out()` (*in module simpleio*), 16

`simpleio` (*module*), 14

T

`tone()` (*in module simpleio*), 16

V

`value` (*simpleio.DigitalIn attribute*), 15

`value` (*simpleio.DigitalOut attribute*), 15