AdafruitCrickit Library Documentation

Release 1.0

Dan Halbert

Jan 14, 2020

Contents

This convenience library makes coding for the Crickit robotics boards simpler and shorter.

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit seesaw library
- Adafruit Motor library

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-crickit
```

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

```
sudo pip3 install adafruit-circuitpython-crickit
```

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-crickit
```

CHAPTER 3

Usage Example

This examples shows how to control all the devices supported by the library. In most cases you just need a couple of imports.

```
# This is a mock example showing typical usage of the library for each kind of device.

from adafruit_crickit import crickit

# Add this import if using stepper motors.
# It will expose constants saying how to step: stepper.FORWARD, stepper.BACKWARD, etc.
from adafruit_motor import stepper

# Set servo 1 to 90 degrees
crickit.servo_1.angle = 90

# Change servo settings.
crickit.servo_1.actuation_range = 135
crickit.servo_1.set_pulse_width_range(min_pulse=850, max_pulse=2100)

# You can assign a device to a variable to get a shorter name.
servo_2 = crickit.servo_2
servo_2.throttle = 0

# Run a continous servo on Servo 2 backwards at half speed.
crickit.continuous_servo_2.throttle = -0.5

# Run the motor on Motor 1 terminals at half speed.
crickit.dc_motor_1.throttle = 0.5

# Set Drive 1 terminal to 3/4 strength.
crickit.drive_1.fraction = 0.75

if crickit.touch_1.value:
    print("Touched terminal Touch 1")
```

(continues on next page)

(continued from previous page)

```
# A single stepper motor uses up all the motor terminals.
crickit.stepper_motor.onestep(direction=stepper.FORWARD)

# You can also use the Drive terminals for a stepper motor
crickit.drive_stepper_motor.onestep(direction=stepper.BACKWARD)

# Note: On CPX Crickit, NeoPixel pin is normally connected to A1, not to seesaw,
# so this part of the demo cannot control the NeoPixel terminal.
# Strip or ring of 8 NeoPixels
crickit.init_neopixel(8)
crickit.neopixel.fill((100, 100, 100))

# Set the Crickit's on-board NeoPixel to a dim purple.
crickit.onboard_pixel.brightness = 0.01
crickit.onboard_pixel[0] = (255, 24, 255)
# or
crickit.onboard_pixel.fill((255, 24, 255))
```

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.

CHAPTER 6

Table of Contents

6.1 Examples

The `examples` directory contains simple examples for controlling a number of different devices, such as servos and DC motors.

6.2 adafruit_crickit

Convenience library for using the Adafruit Crickit robotics boards.

- Author(s): Dan Halbert

6.2.1 Implementation Notes

Hardware:

Adafruit Crickit for Circuit Playground Express Adafruit Crickit FeatherWing

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases

class adafruit_crickit.**Crickit**(*seesaw*)

Represents a Crickit board. Provides a number of devices available via properties, such as `servo_1`. Devices are created on demand the first time they are referenced.

It's fine to refer a device multiple times via its property, but it's faster and results in more compact code to assign a device to a variable.

```
import time
from adafruit_crickit import crickit
```

(continues on next page)

(continued from previous page)

```
# This is fine:
crickit.servo_1.angle = 0
time.sleep(1)
crickit.servo_1.angle = 90
time.sleep(1)

# This is slightly faster and more compact:
servo_1 = crickit.servo_1
servo_1.angle = 0
time.sleep(1)
servo_1.angle = 90
time.sleep(1)
```

SIGNAL1 = 2
Signal 1 terminal

SIGNAL2 = 3
Signal 2 terminal

SIGNAL3 = 40
Signal 3 terminal

SIGNAL4 = 41
Signal 4 terminal

SIGNAL5 = 11
Signal 5 terminal

SIGNAL6 = 10
Signal 6 terminal

SIGNAL7 = 9
Signal 7 terminal

SIGNAL8 = 8
Signal 8 terminal

continuous_servo_1
`adafruit_motor.servo.ContinuousServo` object on Servo 1 terminal

continuous_servo_2
`adafruit_motor.servo.ContinuousServo` object on Servo 2 terminal

continuous_servo_3
`adafruit_motor.servo.ContinuousServo` object on Servo 3 terminal

continuous_servo_4
`adafruit_motor.servo.ContinuousServo` object on Servo 4 terminal

dc_motor_1
`adafruit_motor.motor.DCMotor` object on Motor 1 terminals

dc_motor_2
`adafruit_motor.motor.DCMotor` object on Motor 2 terminals

drive_1
`adafruit_seesaw.pwmout.PWMOut` object on Drive 1 terminal, with `frequency=1000`

drive_2
`adafruit_seesaw.pwmout.PWMOut` object on Drive 2 terminal, with `frequency=1000`

drive_3
adafruit_seesaw.pwmout.PWMOut object on Drive 3 terminal, with frequency=1000

drive_4
adafruit_seesaw.pwmout.PWMOut object on Drive 4 terminal, with frequency=1000

drive_stepper_motor
adafruit_motor.motor.StepperMotor object on Drive terminals

feather_drive_1
adafruit_seesaw.pwmout.PWMOut object on Crickit Featherwing Drive 1 terminal, with frequency=1000

feather_drive_2
adafruit_seesaw.pwmout.PWMOut object on Crickit Featherwing Drive 2 terminal, with frequency=1000

feather_drive_3
adafruit_seesaw.pwmout.PWMOut object on Crickit Featherwing Drive 3 terminal, with frequency=1000

feather_drive_4
adafruit_seesaw.pwmout.PWMOut object on Crickit Featherwing Drive 4 terminal, with frequency=1000

feather_drive_stepper_motor
adafruit_motor.motor.StepperMotor object on Drive terminals on Crickit FeatherWing

init_neopixel(*n*, *, *bpp=3*, *brightness=1.0*, *auto_write=True*, *pixel_order=None*)
Set up a seesaw.NeoPixel object

Note: On the CPX Crickit board, the NeoPixel terminal is by default controlled by CPX pin A1, and is not controlled by seesaw. So this object will not be usable. Instead, use the regular NeoPixel library and specify board.A1 as the pin.

You can change the jumper connection on the bottom of the CPX Crickit board to move control of the NeoPixel terminal to seesaw pin #20 (terminal.NEOPIXEL). In addition, the Crickit FeatherWing always uses seesaw pin #20. In either of those cases, this object will work.

```
from adafruit_crickit.crickit import crickit

crickit.init_neopixel(24)
crickit.neopixel.fill((100, 0, 0))
```

neopixel
`adafruit_seesaw.neopixel object on NeoPixel terminal. Raises ValueError if init_neopixel has not been called.

onboard_pixel
`adafruit_seesaw.neopixel object on the Seesaw on-board NeoPixel. Initialize on-board NeoPixel and clear upon first use.

reset()
Reset the whole Crickit board.

seesaw
The Seesaw object that talks to the Crickit. Use this object to manipulate the signal pins that correspond to Crickit terminals.

```
from adafruit_crickit import crickit

ss = crickit.seesaw
ss.pin_mode(crickit.SIGNAL4, ss.OUTPUT)
ss.digital_write(crickit.SIGNAL4], True)
```

servo_1
adafruit_motor.servo.Servo object on Servo 1 terminal

servo_2
adafruit_motor.servo.Servo object on Servo 2 terminal

servo_3
adafruit_motor.servo.Servo object on Servo 3 terminal

servo_4
adafruit_motor.servo.Servo object on Servo 4 terminal

stepper_motor
adafruit_motor.motor.StepperMotor object on Motor 1 and Motor 2 terminals

touch_1
adafruit_crickit.CrickitTouchIn object on Touch 1 terminal

touch_2
adafruit_crickit.CrickitTouchIn object on Touch 2 terminal

touch_3
adafruit_crickit.CrickitTouchIn object on Touch 3 terminal

touch_4
adafruit_crickit.CrickitTouchIn object on Touch 4 terminal

class adafruit_crickit.**CrickitTouchIn**(*seesaw*, *pin*)
Imitate touchio.TouchIn.

raw_value
The raw touch measurement as an int. (read-only)

value
Whether the touch pad is being touched or not. (read-only)

adafruit_crickit.**crickit = None**
A singleton instance to control a single Crickit board, controlled by the default I2C pins.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

V