robopython Documentation

Release 1.0.4

Jonathan William Morley

Mar 17, 2021

Getting Started:

CHAPTER 1

robopython

Robo Wunderkind Python API - BLED112 USB Dongle Required

Official Robo Wunderkind Modular Robotics Kit Python Interface

For more information about Robo Wunderkind please visit https://www.robowunderkind.com/

Python 2.x Supported Python 3.x Supported

- Free software: Apache Software License 2.0
- Documentation: https://robopython.readthedocs.io.

1.1 Upcoming Features

- New Module integration such as Servo, Motor, Hinge, and Camera
- Fuzzy Logic Filter for Sensors/States

1.2 Getting Started

- Install with: pip install robopython
- from robopython import Robo
- Create an instance of Robo object by doing: my_robo = Robo("BLE Name")
- Test Functionality by playing a sound with: my_robo.System.play_sound(0)

1.3 Chrome OS

- Update to latest Chrome OS
- Set into Developer Mode
- $ sudo dev_install python
- Install ChromeBrew: https://skycocker.github.io/chromebrew/
- crew install freestyle
- pip install robopython
- Run python as sudo for robopython to work

1.4 Troubleshooting

If you get an error saying No BGAPI compatable device is detected please insert the BLED112 USB dongle or switch USB ports If problem persists you can identify the COM port explicitley with my_robo = Robo("BLE Name", COM PORT)

CHAPTER 2

Installation

2.1 Python and Pip Installation

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 Stable release

To install robopython, run this command in your terminal:

```
$ pip install robopython
```

This is the preferred method to install robopython, as it will always install the most recent stable release.

2.3 From sources

The sources for robopython can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/JonRobo/robopython
```

Or download the tarball:

```
$ curl  -OL https://github.com/JonRobo/robopython/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use robopython in a project:

```
import robopython
OR
from robopython import Robo

my_robo = Robo("BLE_Name")
```

CHAPTER 4

Robo

Create Robo Instance

from robopython import Robo

BLE_Name = “Robo1” – example BLE name

```
my_robo = Robo(BLE_Name)
```

4.1 Battery Level

battery_level(`self`)

Returns the battery level in % and the status.

```
>>> print my_robo.battery_level()
>>> 80, Discharging
```

4.2 Change BLE Name

change_ble_name(`self`, `name`)

`name` must be 16 characters or less e.g name = “Robo1”
Robo will reboot once the command is received and the name is changed

You can use an instance of BLED112 to scan() and see all discoverable devices or simply check on your phone that the new ble name has been taken

```
my_robo.change_ble_name("Robo2")
```

4.3 Check Drive Action

check_drive_action(self)

returns the status of the drive action 1 = done, None = in progress, 0 = failed

```
if my_robo.check_drive_action():
        print "Drive Action Complete"
```

4.4 Check Turn Action

check_turn_action(self)

returns the status of the turn action 1 = done, None = in progress, 0 = failed

```
if my_robo.check_turn_action():
        print "Turn Action Complete"
```

4.5 Delay

delay(self, delay_time)

delay_time is the time in seconds to wait until function returns

```
my_robo.delay(1)
```

4.6 Display Text

display_text(self, text, matrices)

Cascaded diplay of text on one or more LED Matices

text is a string to be displayed on the matrices
matrices is a list of Matrix objects that should participate in the text display

```
my_robo.display_text("Welcome to Robo Wunderkind", [my_robo.Matrix1, my_robo.Matrix2,␣
↪my_robo.Matrix3])
```

4.7 Drive

drive(`self`, `vel`, `distance`, `direction`, `wait=1`, `motors=(1, 2)`, `wd=89`)

drive is Robo's simplified command to have Robo drive a certain distance assuming a 2 motor configuration

`vel` is the desired velocity from 0-100%
`distance` is how far in centimeters Robo should drive
`direction` will detirmine forward or backward
`wait` is a flag that indicates if we wait in the function until the action is complete. set wait = 0 if we want to exit the function while driving
`motors` is a tuple of the two motors used to drive Robo
`wd` is the diameter of the wheels used, default is 89mm or 0x59mm in hex.
If you choose to change the wheels be sure to pass in the new wheel diameter

```
my_robo.drive(80, 30, 1)
```

4.8 Drive Forever

drive_inf(`self`, `vel`, `direction`)

`vel` is the desired velocity from 0-100%
`direction` will detirmine forward or backward

```
my_robo.drive_inf(80, 1)
```

4.9 Turn Forever

turn_inf(`self`, `vel`, `direction`)

`vel` is the desired velocity from 0-100%
`direction` will detirmine forward or backward

```
my_robo.turn_inf(80, 1)
```

4.10 Firmware Version

firmware(`self`)

Returns the firmware version of Robo

```
my_robo.firmware()
```

4.11 Get Robo Build

get_build(`self`)

returns a list of Robo Wunderkind moduels that are currently attached
The is updated automatically upon initialization of Robo object as well as when there has been a change in the build
The latest build is stored in self.build -> my_robo.build

```
build = my_robo.get_build()
```

4.12 Get BLE Characteristics

get_characteristics(`self`)

characteristics = my_robo.get_characteristics()
returns a list of GATT characteristics

```
characteristics = my_robo.get_characteristics()
```

4.13 Get RSSI

get_rssi(`self`)

rssi = my_robo.get_rssi()
returns the BLE signal strength rssi value

```
signal_strength = my_robo.get_rssi()
```

4.14 Set Drive Command

set_drive(`self`, `motor_cmds`, `vel`, `distance`, `action_id`, `wd=0x59`)

set_drive is Robo's generic command to set the velocity and distance commands to multiple motors at once

`motor_cmds` is a list of motor objects folloed by the direction that motor should spin: [[1,0],[2,1],[3,0],[4,1]] motors from 1-6 are valid if connected
`vel` is the desired velocity from 0-100%
`distance` is the desired distance to travel in centimeters
`action_id` is a unique identifier that is sent back once Robo has completed the action. Use the self.drive_id by default, use check_drive_action() to know when it is done
`wd` is the diameter of the wheels used, default is 89mm or 0x59mm in hex. If you choose to change the wheels be sure to pass in the new wheel diameter

```
my_robo.set_drive([[1,0],[2,1],[3,0],[4,1]], 50, 100, my_robo.drive_id)
```

4.15 Sound Playback

sound(`self`, `sound`)

Plays the desired sound clip on the system cube speaker 0-7 are valid

```
my_robo.sound(0)
```

4.16 Stop

stop(`self`)

stops all motors from moving

```
my_robo.stop()
```

4.17 Stop All Actions

stop_all(`self`)

stops all outputs

```
my_robo.stop_all()
```

4.18 Turn

turn(self, vel, angle, direction, wait=1, motors=(1, 2), wd=89, turning_radius=91)

turn is Robo's simplified command to have Robo turn a number of degrees assuming a 2 motor configuration

vel is the desired velocity from 0-100%
angle is the amount to have Robo turn in degrees
direction will detirmine clockwise or counter clockwise rotation
wait is a flag that indicates if we wait in the function until the turn is complete. set wait = 0 if we want to exit the function while turning
motors is a tuple of the two motors used to turn Robo
wd is the diameter of the wheels used, default is 89mm or 0x59mm in hex. If you choose to change the wheels be sure to pass in the new wheel diameter
turning_radius is the distance from the wheel to the centre of Robo's turning axle in millimeters

```
my_robo.turn(40, 90, 1)
```

CHAPTER 5

BLED112

Create BLED112 Instance

```
BLE = BLED112(com_port=None)
```

5.1 Connect to Device

connect_ble(`self`, `name`)

Connects to the BLE device with the name passed into the function if it exists

```
BLE.connect_ble("Robo")
```

5.2 Scan

scan(`self`)

Scans for nearby BLE devices and returns a list of dictionaries containing BLE device data

```
BLE.scan()
```

5.3 Start

start(`self`)

Starts the BLE dongle

```
BLE.start()
```

5.4 Stop

start(`self`)

Starts the BLE dongle

```
BLE.start()
```

CHAPTER 6

Accelerometer

6.1 Get Sensor Values

get_values(`self`)

Returns a dictionary of all three acceleration values in gravitational units and all three gyro values as degrees per second

```
values = Robo.IMU1.get_values()
```

CHAPTER 7

Button

7.1 Get Button State

get_state(`self`)

Returns the current state of the button

```
state = Robo.Button1.get_state()
```

7.2 Set Button Trigger

set_trigger(`self`, `comparitor`)

`comparitor` 0 sets pressed, -1 sets released, 1-254 defines the number of clicks before triggered

```
Robo.Button1.set_trigger(5) # sets the trigger for 5 clicks
```

7.3 Check Button Trigger

check_trigger(`self`)

Returns 1 if the trigger has happened

```
while not Robo.Button1.check_trigger():
        # do stuff
```

CHAPTER 8

Display

8.1 Pre-Programmed Image

image(`self`, `image_num`, `orientation`, `delay`)

`image_num` is the index of the pre-programmed image 0-4
`orientation` is the desired orientation to display the image. 0 - 3 -> 0 - 270 degrees
`delay` is the time in milliseconds that the image will be displayed for 0->65535
Displays the pre-programmed `image_num` on the LED Display cube

```
Robo.Display1.image(0, 2, 1000)
```

8.2 Custom Image

custom_image(`self`, `image`, `orientation`, `delay`)

`image` is a 32 element array of 8-bit numbers representing the 16 rows of the desired image.
`orientation` is the desired orientation to display the image. 0 - 3 -> 0 - 270 degrees
`delay` is the time in milliseconds that the image will be displayed for 0->65535
Displays the given `image` on the LED Display cube

```
Robo_Logo = [0xff, 0xfc, 0xff, 0xfe, 0xef, 0xf7, 0xc7, 0xe3, 0xc4, 0x23, 0xee, 0x77,␣
↪0xff, 0xfe, 0xff, 0xfc, 0x00, 0x00, 0xfe, 0x3c, 0xfe, 0x7e, 0xfe, 0xff, 0xfe, 0xff,␣
↪0xfe, 0xff, 0xfe, 0x7e, 0xfe, 0x3c ]

Robo.Display1.custom_image(Robo_Logo, 1, 1000)
```

8.3 Pre-Programmed Animation

animate(`self`, `animation_num`, `repeats`, `reverse`, `orientation`)

`animation_num` is the index of the pre-programmed animation 0-2
`repeats` number of times to repeat the animation
`reverse` 0 or 1 to indicate if the animation should play forwards then backwards or just forwards
`orientation` is the desired orientation to display the image. 0 - 3 -> 0 - 270 degrees
`delay` is the time in milliseconds that the image will be displayed 0->65535
Displays the pre-programmed `animation_num` on the LED Display cube

```
Robo.Display1.animate(0, 5, 1, 0)
```

8.4 Custom Animation

custom_animation(`self`, `animation`, `repeats`, `reverse`, `orientation`, `frame_rate`)

`animation` is a list of images: 32 element arrays of 8-bit numbers representing the 16 rows of the desired images.
`repeats` number of times to repeat the animation
`reverse` 0 or 1 to indicate if the animation should play forwards then backwards or just forwards
`orientation` is the desired orientation to display the image. 0 - 3 -> 0 - 270 degrees
`frame_rate` is the time in milliseconds that the image frame will be displayed 0->65535
Displays the given `animation` on the LED Display cube

```
Robo_Logo = [0xff, 0xfc, 0xff, 0xfe, 0xef, 0xf7, 0xc7, 0xe3, 0xc4, 0x23, 0xee, 0x77,␣
↪0xff, 0xfe, 0xff, 0xfc, 0x00, 0x00, 0xfe, 0x3c, 0xfe, 0x7e, 0xfe, 0xff, 0xfe, 0xff,␣
↪0xfe, 0xff, 0xfe, 0x7e, 0xfe, 0x3c ]
One = [0x07, 0xc0, 0x0f, 0xc0, 0x1d, 0xc0, 0x19, 0xc0, 0x01, 0xc0, 0x01, 0xc0, 0x01,␣
↪0xc0, 0x01, 0xc0, 0x01, 0xc0, 0x01, 0xc0, 0x01, 0xc0, 0x01, 0xc0, 0x01, 0xc0, 0x1f,␣
↪0xfc, 0x1f, 0xfc, 0x1f, 0xfc]
Two = [0x07, 0x80, 0x0f, 0xc0, 0x1c, 0xe0, 0x18, 0x70, 0x18, 0x30, 0x00, 0x30, 0x00,␣
↪0x30, 0x00, 0x30, 0x00, 0x30, 0x00, 0x70, 0x00, 0x70, 0x00, 0xf0, 0x03, 0xc0, 0x07,␣
↪0xc0, 0x0f, 0xfc, 0x0f, 0xfc]
Three = [0x0f, 0xfe, 0x0f, 0xfe, 0x0f, 0x1e, 0x0e, 0x0e, 0x00, 0x0e, 0x00, 0x0e, 0x00,
↪ 0x0e, 0x00, 0xfe, 0x00, 0xfe, 0x00, 0xfe, 0x00, 0x0e, 0x00, 0x0e, 0x0e, 0x0e, 0x0f,
↪ 0x1e, 0x0f, 0xfe, 0x0f, 0xfe]

animation = [Robo_Logo, One, Two, Three]

Robo.Display1.custom_animation(animation, 1, 0, 0, 1000)
```

8.5 Reset

reset(`self`)

Clears and resets the LED Display cube

```
Robo.Display1.reset()
```

CHAPTER 9

Light Sensor

9.1 Get Light Value

get_light(`self`)

Returns the light value in lx of the specified light cube

```
light = Robo.Light1.get_light()
```

9.2 Set Light Trigger

set_trigger(`self`, `value`, `comparitor`)

`value` is the lx value you want to se the trigger to
`comparitor` 0 is less than, 1 is greater than the set value

```
Robo.Light1.set_trigger(500, 0)
```

9.3 Check Light Trigger

check_trigger(`self`)

Returns 1 if the trigger has happened

```
while not Robo.Light1.check_trigger():
        # do stuff
```

CHAPTER 10

Line Tracker LT

10.1 Get Sensor Values

get_sensor_values(`self`)

Returns a list of all three values of the lintracker IR sensors
[left, center, right]

```
values = Robo.LT1.get_sensor_values()
```

10.2 Get Right Value

get_right_value(`self`)

Returns the lintracker's right IR sensor value

```
value = Robo.LT1.get_right_value()
```

10.3 Get Center Value

get_center_value(`self`)

Returns the lintracker's center IR sensor value

```
value = Robo.LT1.get_center_value()
```

10.4 Get Left Value

get_left_value(self)

Returns the lintracker's left IR sensor value

```
value = Robo.LT1.get_left_value()
```

10.5 Track Line

track(self,''direction'','speed'')
direction must be 0 or 1 to indicate forwards or backwards
speed must be between 0 and 100%

```
Robo.LT1.track(0,75)
```

10.6 Check Line Tracker Action

check_action(self)

Returns 1 if the action is completed

```
while not Robo.LT1.check_action():
        # do stuff
```

CHAPTER 11

LED Matrix

11.1 Display

set_display(`self`, `rows`)

`rows` is an 8 element array of 8-bit strings.
Displays the given `rows` on the specified LED Matrix

```
rows = ['00000000',
        '11000011',
        '11000011',
        '00000000',
        '10000001',
        '11000011',
        '00111100',
        '00000000'
       ],

Robo.Matrix1.set_display(rows)
```

11.2 Timed Display

set_display(`self`, `rows`, `duration`)

`rows` is an 8 element array of 8-bit strings.
`duration` is the time in seconds
Displays the given `rows` on the specified LED Matrix for the specified amount of time

```
rows = ['00000000',
        '11000011',
        '11000011',
        '00000000',
        '10000001',
        '11000011',
        '00111100',
        '00000000'
       ],

Robo.Matrix1.set_display(rows, 4)
```

11.3 Off

off(self)

Turns off the LED Matrix

```
Robo.Matrix1.off()
```

11.4 Check Action

check_action(self)

Returns 1 if the the action has completed

```
while not Robo.Matrix1.check_action():
        # do stuff while the timed display is on
```

11.5 Rotate Right

rotate_right(self, rows)

Rotates the image to the right by 90 degrees

```
image = Robo.Matrix1.rotate_right(rows)
```

11.6 Rotate Left

rotate_left(self, rows)

Rotates the image to the left by 90 degrees

```
image = Robo.Matrix1.rotate_left(rows)
```

11.7 Flip X

flip_x(`self`, `rows`)

Flips the image in the X axis

```
image = Robo.Matrix1.flip_x(rows)
```

11.8 Flip Y

flip_y(`self`, `rows`)

Flips the image in the Y axis

```
image = Robo.Matrix1.flip_y(rows)
```

CHAPTER 12

Motion Sensor PIR

12.1 Get Motion State

get_state(`self`)

Returns the current state of the PIR

```
state = Robo.Motion1.get_state()
```

12.2 Set Motion Trigger

set_trigger(`self`, `comparitor`)

`comparitor` 0 sets no motion, 1 sets motion

```
Robo.Motion1.set_trigger(1) # sets the trigger to set off with motion
```

12.3 Check Motion Trigger

check_trigger(`self`)

Returns 1 if the trigger has happened

```
while not Robo.Motion1.check_trigger():
        # do stuff
```

CHAPTER 13

Motor

13.1 Set PWM

set_pwm(`self`, `pwm`)

`pwm` must be between 0 and 255; where 127 = 100% CW and 129 = 100% CCW
0-100% CW -> 0-127 pwm 0-100% CCW -> 255-129
Sets a constant torque PWM signal to the motor

```
Robo.Motor1.set_pwm(60)
```

13.2 Spin Distance

spin_distance(`self`, `vel`, `distance`, `wd=89`)

`vel` must be between 0 and 100%
`distance` must be an integer in centimeters
`wd` is the wheel diameter and is by default 89 mm, the diameter of the large wheel.
If different wheels are attached please specify the diameter in mm

Robo will spin the motor at a constant velocity for the specified distance. If resistance is applied the motor will increase torque to try and maintain the velocity.
When approaching the end distance the motor slows down to ensure it covers the specified distance and does not overshoot

```
Robo.Motor1.set_distance(60, 80)
```

13.3 Spin Velocity

spin_velocity(`self`, `vel`)

`vel` must be between 0 and 100%

Robo will spin the motor at a constant velocity. If resistance is applied the motor will increase torque to try and maintain the velocity.

```
Robo.Motor1.set_velocity(60)
```

13.4 Stop

stop(`self`)

Stops the motor

```
Robo.Motor1.stop()
```

13.5 Check Motor Action

check_action(`self`)

Returns 1 if the action is completed

```
while not Robo.Motor1.check_action():
        # do stuff
```

CHAPTER 14

RGB LED

14.1 Set Colour

set_rgb(`self`, `red`, `green`, `blue`)

Sets the RGB to the desired colour using a combination of red, green blue colour intensities

`red` 0-255 value for the red colour intensity
`green` 0-255 value for the green colour intensity
`blue` 0-255 value for the blue colour intensity

```
Robo.RGB1.set_rgb(120,25,255)
```

14.2 Blink RGB

blink_rgb(`self`, `red`, `green`, `blue`, `num_blinks`,‘‘period‘‘)

`red` 0-255 value for the red colour intensity
`green` 0-255 value for the green colour intensity
`blue` 0-255 value for the blue colour intensity
`num_blinks` number of blinks
`period` period in milliseconds

```
Robo.RGB1.blink_rgb(255, 255, 255, 5, 1000)      # blink white 5 times
```

14.3 Timed RGB

timed_rgb(`self`, `red`, `green`, `blue`, `time`)

`red` 0-255 value for the red colour intensity
`green` 0-255 value for the green colour intensity
`blue` 0-255 value for the blue colour intensity
`time` LED on time in seconds

```
Robo.RGB1.timed_rgb(255, 255, 255, 2)    # white light for 2 seconds
```

14.4 Off

off(`self`)

Turns off the LED

```
Robo.RGB1.off()
```

14.5 Red

red(`self`)

Turns the LED Red

```
Robo.RGB1.red()
```

14.6 Green

green(`self`)

Turns the LED Green

```
Robo.RGB1.green()
```

14.7 Blue

blue(`self`)

Turns the LED Blue

```
Robo.RGB1.blue()
```

14.8 Yellow

yellow(`self`)

Turns the LED Yellow

```
Robo.RGB1.yellow()
```

14.9 Orange

orange(`self`)

Turns the LED Orange

```
Robo.RGB1.orange()
```

14.10 White

white(`self`)

Turns the LED White

```
Robo.RGB1.white()
```

14.11 Random Colour

random(`self`)

Shows a random colour on the LED

```
Robo.RGB1.random()
```

14.12 Check LED Action

check_action(`self`)

Returns 1 if the action is completed

```
while not Robo.RGB1.check_action():
        # do stuff
```

CHAPTER 15

Servo

15.1 Set Angle

set_angle(`self`, `angle`)

`angle` must be between 0 and 255

Sets the servo to be positioned at the desired angle

```
Robo.Motor1.set_angle(90)
```

15.2 Check Servo Action

check_action(`self`)

Returns 1 if the action is completed

```
while not Robo.Servo1.check_action():
        # do stuff
```

CHAPTER 16

Ultrasonic Sensor

16.1 Get Distance

get_distance(`self`)

Returns the current distance

```
distance = Robo.Ultrasonic1.get_distance()
```

16.2 Get Sound Level

get_sound_level(`self`)

Returns the current sound level

```
sound = Robo.Ultrasonic1.get_sound()
```

16.3 Set Distance Trigger

set_trigger(`self`, `value`,‘‘comparitor‘‘)

`value` is the trigger value in centimeters
`comparitor` 0 sets less than, 1 sets greater than value

```
Robo.Ultrasonic1.set_trigger(50, 1)  # sets the trigger to set off when a distance␣
↪greater than 50 cm is seen
```

16.4 Set Sound Trigger

set_sound_trigger(`self`, `value`,‘‘comparitor‘‘)

`value` is the trigger value
`comparitor` 0 sets less than, 1 sets greater than value

```
Robo.Ultrasonic1.set_sound_trigger(150, 1)  # sets the trigger to set off when the␣
↪sound is greater than 150
```

16.5 Check Distance Trigger

check_distance_trigger(`self`)

Returns 1 if the trigger has happened

```
while not Robo.Ultrasonic1.check_distance_trigger():
        # do stuff
```

16.6 Check Sound Trigger

check_distance_trigger(`self`)

Returns 1 if the trigger has happened

```
while not Robo.Ultrasonic1.check_sound_trigger():
        # do stuff
```

CHAPTER 17

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

17.1 Types of Contributions

17.1.1 Report Bugs

Report bugs at https://github.com/JonRobo/robopython/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

17.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

17.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

17.1.4 Write Documentation

robopython could always use more documentation, whether as part of the official robopython docs, in docstrings, or even on the web in blog posts, articles, and such.

17.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/JonRobo/robopython/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

17.2 Get Started!

Ready to contribute? Here's how to set up *robopython* for local development.

1. Fork the *robopython* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/robopython.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv robopython
$ cd robopython/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 robopython tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

17.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/JonRobo/robopython/pull_requests and make sure that the tests pass for all supported Python versions.

17.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_robopython
```

17.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 18

Credits

18.1 Development Lead

- Jonathan William Morley <jon@robowunderkind.com>

18.2 Contributors

None yet. Why not be the first?

CHAPTER 19

History

19.1 0.1.0 (2018-11-06)

- First release on PyPI.

19.2 0.4.3 (2019-01-16)

19.3 0.4.4 (2019-10-15)

19.4 0.4.5 (2019-10-18)

19.5 0.4.6 (2019-10-27)

19.6 0.4.7 (2019-11-13)

19.7 0.4.8 (2020-02-19)

19.8 0.5.0 (2020-03-04)

19.9 1.0.1 (2020-09-03)

19.10 1.0.2 (2020-09-17)

19.11 1.0.3 (2021-03-15)

19.12 1.0.4 (2021-03-17)

- Latest release on PyPI.

CHAPTER 20

Indices and tables

- genindex
- modindex
- search