
Spockbots

Release 1.0

Spockbots

Nov 06, 2019

CONTENTS

1	Spockbots API	1
1.1	Menu	1
1.2	City Runs	1
1.3	Spockbots API	2
1.4	Examples	8
2	Indices and tables	9
	Python Module Index	11
	Index	13

SPOCKBOTS API

1.1 Menu

We named it *0_menu.py* so it shows up on the top in the brick program:

```
Crane
>>> Swing
Calibrate
....
```

Displays a menu in which we move with the UP DOWN keys up and down. We leave with the left key and select a program with the right key.

1.2 City Runs

1.2.1 run.calibrate

`run.calibrate.run_calibrate()`

Run the calibration

Returns a file called `calibrate.txt` that contains the minimum black and the maximum white value for the sensors

1.2.2 run.check

`run.check.run_check()`

Checks the robot by driving the large and medium motors and flashing the color sensors

Order:

- Large Motor left
- Large Motor left
- Medium Motor left
- Medium Motor left
- Color Sensor left
- Color Sensor right
- Color Sensor back

1.2.3 run.crane

`run.crane.run_crane()`
TBD

1.2.4 run.led

`run.led.run_led()`
TBD

1.2.5 run.swing

`run.swing.run_swing()`
TBD

1.2.6 run.turn_to_black module

1.3 Spockbots API

1.3.1 spockbots.check

`spockbots.check.check(speed=100, angle=360)`
do a robot check by

- a) turning on the large motors one at a time
- b) turning on the medium motors one at a time
- c) turning on the light sensors one at a time

Parameters

- **speed** –
- **angle** –

Returns

1.3.2 spockbots.colorsensor

class `spockbots.colorsensor.SpockbotsColorSensor` (*port=3*)
Bases: `object`
TBD
clear()
flash()
 flashes the color sensor by switching between color and reflective mode
info()
 prints the black and white value read from the sensor
light()

Returns

read()
reads the color sensor data form the file :return:

reflection()

Returns

set_black()
sets the current value to black

set_white()
sets the current value to white :return:

value()
reads the current value mapped between 0 and 100 :return:

write()
append the black and white value to a file

class spockbots.colorsensor.**SpockbotsColorSensors** (*ports=[2, 3, 4], speed=5*)

Bases: object

TBD

clear()

Returns

flash (*ports=[2, 3, 4]*)

Parameters **ports** –

Returns

info (*ports=[2, 3, 4]*)

Parameters **ports** –

Returns

read()

Returns

sensor (*port*)

Parameters **port** –

Returns

test (*ports=[2, 3, 4]*)

Parameters **ports** –

Returns

value (*i*)

Parameters **i** –

Returns

write (*ports=[2, 3, 4]*)

Parameters **ports** –

Returns

1.3.3 spockbots.gyro

1.3.4 spockbots.motor

`spockbots.motor.PRINT(*args)`

class `spockbots.motor.SpockbotsMotor` (*direction=None*)

Bases: `object`

angle_to_distance (*angle*)

Parameters **angle** –

Returns

beep ()

The robot will make a beep

calibrate (*speed*, *distance=15*, *ports=[2, 3, 4]*, *direction='front'*)

Parameters

- **speed** –
- **distance** –
- **ports** –
- **direction** –

Returns

distance_to_angle (*distance*)

calculation to convert the distance from cm into rotations.

Parameters **distance** – The distance in cm

Returns The rotations to be traveled for the given distance

distance_to_rotation (*distance*)

calculation to convert the distance from cm into rotations.

Parameters **distance** – The distance in cm

Returns The rotations to be traveled for the given distance

followline (*speed=25*, *distance=None*, *t=None*, *port=3*, *right=True*, *black=0*, *white=100*, *delta=-35*,
factor=0.7)

Parameters

- **speed** –
- **distance** –
- **t** –
- **port** –
- **right** –
- **black** –
- **white** –
- **delta** –
- **factor** –

Returns**forward** (*speed, distance, brake=None*)**Parameters**

- **speed** –
- **distance** –
- **brake** –

Returns**gotoblack** (*speed, port, black=10*)

The robot moves to the black line while using the sensor on the given port

Parameters

- **speed** – The speed
- **port** – The port 1,2,3,4
- **black** – The value to stop

gotowhite (*speed, port, white=90*)

The robot moves to the white line while using the sensor on the given port

Parameters

- **speed** – The speed
- **port** – The port 1,2,3,4
- **white** – The value to stop

light (*port*)**Parameters** **port** –**Returns****on** (*speed, steering=0*)**Parameters**

- **speed** –
- **steering** –

Returns**reset** ()**Returns****setup** (*direction=None*)**Parameters** **direction** –**Returns****still** ()**Returns****stop** (*brake=None*)

stops all motors on all different drive modes

Parameters **brake** – None, brake, coast, hold

Returns

tunrtoblack (*speed, direction='left', port=3, black=10*)

turns the robot to the balck line. :param speed: :param port: :param black: :return:

turn (*speed, angle*)

takes the radius of the robot and dives on it for a distance based on the ancle :param speed: :param angle: :return:

1.3.5 spockbots.output

`spockbots.output.PRINT (*args, x=None, y=None)`

Parameters

- **args** –
- **x** –
- **y** –

Returns

`spockbots.output.beep()`

The robot will make a beep

`spockbots.output.clear()`

Returns

`spockbots.output.flash(colors=['RED', 'BLACK', 'RED', 'BLACK', 'GREEN'], delay=0.1)`

The robot will flash the LEDs and beep twice

`spockbots.output.led(color, brightness=255)`

Parameters

- **color** –
- **brightness** –

Returns

`spockbots.output.readfile(name)`

Parameters name –

Returns

`spockbots.output.sound(pitch=1500, duration=300)`

`spockbots.output.voltage()`

Returns

`spockbots.output.writefile(name, msg)`

Parameters

- **name** –
- **msg** –

Returns

1.3.6 spockbots.robot

1.3.7 spockbots.systemgyro

```
class spockbots.systemgyro.Gyro
    Bases: object

    angle ()
        Returns

    connect ()
        Returns

    get ()
        Returns

    info ()
        Returns

    mode (kind)
        GYRO-G&A GYRO-ANG GYRO-RATE GYRO-CAL
        Not supported: GYRO-FAS TILT-RATE TILT-ANG
        Parameters kind –
        Returns

    rate ()
        Returns

    reset ()
        Returns

    still (count=10, still=5)
        Parameters
            • count –
            • still –
        Returns

    test (n)
        Parameters n –
        Returns
```

1.4 Examples

1.4.1 door

1.4.2 gyro

1.4.3 interpreter

1.4.4 m

1.4.5 test

INDICES AND TABLES

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

PYTHON MODULE INDEX

r

`run.calibrate`, 1
`run.check`, 1
`run.crane`, 2
`run.led`, 2
`run.swing`, 2

s

`spockbots.check`, 2
`spockbots.colorsensor`, 2
`spockbots.motor`, 4
`spockbots.output`, 6
`spockbots.robot`, 7
`spockbots.systemgyro`, 7

A

`angle()` (*spockbots.systemgyro.Gyro method*), 7
`angle_to_distance()` (*spockbots.motor.SpockbotsMotor method*), 4

B

`beep()` (*in module spockbots.output*), 6
`beep()` (*spockbots.motor.SpockbotsMotor method*), 4

C

`calibrate()` (*spockbots.motor.SpockbotsMotor method*), 4
`check()` (*in module spockbots.check*), 2
`clear()` (*in module spockbots.output*), 6
`clear()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2
`clear()` (*spockbots.colorsensor.SpockbotsColorSensors method*), 3
`connect()` (*spockbots.systemgyro.Gyro method*), 7

D

`distance_to_angle()` (*spockbots.motor.SpockbotsMotor method*), 4
`distance_to_rotation()` (*spockbots.motor.SpockbotsMotor method*), 4

F

`flash()` (*in module spockbots.output*), 6
`flash()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2
`flash()` (*spockbots.colorsensor.SpockbotsColorSensors method*), 3
`followline()` (*spockbots.motor.SpockbotsMotor method*), 4
`forward()` (*spockbots.motor.SpockbotsMotor method*), 5

G

`get()` (*spockbots.systemgyro.Gyro method*), 7
`gotoblack()` (*spockbots.motor.SpockbotsMotor method*), 5

`gotowhite()` (*spockbots.motor.SpockbotsMotor method*), 5
`Gyro` (*class in spockbots.systemgyro*), 7

I

`info()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2
`info()` (*spockbots.colorsensor.SpockbotsColorSensors method*), 3
`info()` (*spockbots.systemgyro.Gyro method*), 7

L

`led()` (*in module spockbots.output*), 6
`light()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2
`light()` (*spockbots.motor.SpockbotsMotor method*), 5

M

`mode()` (*spockbots.systemgyro.Gyro method*), 7

O

`on()` (*spockbots.motor.SpockbotsMotor method*), 5

P

`PRINT()` (*in module spockbots.motor*), 4
`PRINT()` (*in module spockbots.output*), 6

R

`rate()` (*spockbots.systemgyro.Gyro method*), 7
`read()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 3
`read()` (*spockbots.colorsensor.SpockbotsColorSensors method*), 3
`readfile()` (*in module spockbots.output*), 6
`reflection()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 3
`reset()` (*spockbots.motor.SpockbotsMotor method*), 5
`reset()` (*spockbots.systemgyro.Gyro method*), 7
`run.calibrate` (*module*), 1
`run.check` (*module*), 1

`run.crane(module)`, 2
`run.led(module)`, 2
`run.swing(module)`, 2
`run_calibrate()` (in module `run.calibrate`), 1
`run_check()` (in module `run.check`), 1
`run_crane()` (in module `run.crane`), 2
`run_led()` (in module `run.led`), 2
`run_swing()` (in module `run.swing`), 2

S

`sensor()` (`spockbots.colorsensor.SpockbotsColorSensors` method), 3
`set_black()` (`spockbots.colorsensor.SpockbotsColorSensor` method), 3
`set_white()` (`spockbots.colorsensor.SpockbotsColorSensor` method), 3
`setup()` (`spockbots.motor.SpockbotsMotor` method), 5
`sound()` (in module `spockbots.output`), 6
`spockbots.check(module)`, 2
`spockbots.colorsensor(module)`, 2
`spockbots.motor(module)`, 4
`spockbots.output(module)`, 6
`spockbots.robot(module)`, 7
`spockbots.systemgyro(module)`, 7
`SpockbotsColorSensor` (class in `spockbots.colorsensor`), 2
`SpockbotsColorSensors` (class in `spockbots.colorsensor`), 3
`SpockbotsMotor` (class in `spockbots.motor`), 4
`still()` (`spockbots.motor.SpockbotsMotor` method), 5
`still()` (`spockbots.systemgyro.Gyro` method), 7
`stop()` (`spockbots.motor.SpockbotsMotor` method), 5

T

`test()` (`spockbots.colorsensor.SpockbotsColorSensors` method), 3
`test()` (`spockbots.systemgyro.Gyro` method), 7
`tunrtoblack()` (`spockbots.motor.SpockbotsMotor` method), 6
`turn()` (`spockbots.motor.SpockbotsMotor` method), 6

V

`value()` (`spockbots.colorsensor.SpockbotsColorSensor` method), 3
`value()` (`spockbots.colorsensor.SpockbotsColorSensors` method), 3
`voltage()` (in module `spockbots.output`), 6

W

`write()` (`spockbots.colorsensor.SpockbotsColorSensor` method), 3

`write()` (`spockbots.colorsensor.SpockbotsColorSensors` method), 3
`writefile()` (in module `spockbots.output`), 6