

---

# Spockbots

*Release 1.0*

**Spockbots**

**Nov 06, 2019**



# CONTENTS

<b>1</b>	<b>Spockbots API</b>	<b>1</b>
1.1	Menu . . . . .	1
1.2	City Runs . . . . .	1
1.3	Spockbots API . . . . .	2
1.4	Examples . . . . .	5
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## 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()
```

### 1.2.4 run.led

```
run.led.run_led()
```

### 1.2.5 run.swing

```
run.swing.run_swing()
```

### 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  
  
    clear()  
  
    flash()  
        flashes the color sensor by switching between color and reflective mode  
  
    info()  
        prints the black and white value read form the sensor  
  
    light()  
  
    read()  
        reads the color sensor data form the file :return:  
  
    reflection()
```

```

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

    clear()

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

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

    read()

    sensor (port)

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

    value (i)

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

```

### 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)

    beep()
        The robot will make a beep

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

    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)

    forward (speed, distance, brake=None)

```

**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*)

**on** (*speed, steering=0*)

**reset** ()

**setup** (*direction=None*)

**still** ()

**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*)

spockbots.output.**beep** ()

The robot will make a beep

spockbots.output.**clear** ()

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*)

spockbots.output.**readfile** (*name*)

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

spockbots.output.**voltage** ()

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



### 1.3.6 spockbots.robot

### 1.3.7 spockbots.systemgyro

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

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

    rate()
    reset()
    still(count=10, still=5)
    test(n)
```

## 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`, 3
- `spockbots.output`, 4
- `spockbots.robot`, 5
- `spockbots.systemgyro`, 5



## INDEX

### A

`angle()` (*spockbots.systemgyro.Gyro method*), 5  
`angle_to_distance()` (*spockbots.motor.SpockbotsMotor method*), 3

### B

`beep()` (*in module spockbots.output*), 4  
`beep()` (*spockbots.motor.SpockbotsMotor method*), 3

### C

`calibrate()` (*spockbots.motor.SpockbotsMotor method*), 3  
`check()` (*in module spockbots.check*), 2  
`clear()` (*in module spockbots.output*), 4  
`clear()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2  
`clear()` (*spockbots.colorsensor.SpockbotsColorSensors method*), 3  
`connect()` (*spockbots.systemgyro.Gyro method*), 5

### D

`distance_to_angle()` (*spockbots.motor.SpockbotsMotor method*), 3  
`distance_to_rotation()` (*spockbots.motor.SpockbotsMotor method*), 3

### F

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

### G

`get()` (*spockbots.systemgyro.Gyro method*), 5  
`gotoblack()` (*spockbots.motor.SpockbotsMotor method*), 3

`gotowhite()` (*spockbots.motor.SpockbotsMotor method*), 4  
`Gyro` (*class in spockbots.systemgyro*), 5

### I

`info()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2  
`info()` (*spockbots.colorsensor.SpockbotsColorSensors method*), 3  
`info()` (*spockbots.systemgyro.Gyro method*), 5

### L

`led()` (*in module spockbots.output*), 4  
`light()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2  
`light()` (*spockbots.motor.SpockbotsMotor method*), 4

### M

`mode()` (*spockbots.systemgyro.Gyro method*), 5

### O

`on()` (*spockbots.motor.SpockbotsMotor method*), 4

### P

`PRINT()` (*in module spockbots.motor*), 3  
`PRINT()` (*in module spockbots.output*), 4

### R

`rate()` (*spockbots.systemgyro.Gyro method*), 5  
`read()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2  
`read()` (*spockbots.colorsensor.SpockbotsColorSensors method*), 3  
`readfile()` (*in module spockbots.output*), 4  
`reflection()` (*spockbots.colorsensor.SpockbotsColorSensor method*), 2  
`reset()` (*spockbots.motor.SpockbotsMotor method*), 4  
`reset()` (*spockbots.systemgyro.Gyro method*), 5  
`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), 2`  
`set_white () (spockbots.colorsensor.SpockbotsColorSensor method), 3`  
`setup () (spockbots.motor.SpockbotsMotor method), 4`  
`sound () (in module spockbots.output), 4`  
`spockbots.check (module), 2`  
`spockbots.colorsensor (module), 2`  
`spockbots.motor (module), 3`  
`spockbots.output (module), 4`  
`spockbots.robot (module), 5`  
`spockbots.systemgyro (module), 5`  
`SpockbotsColorSensor (class in spockbots.colorsensor), 2`  
`SpockbotsColorSensors (class in spockbots.colorsensor), 3`  
`SpockbotsMotor (class in spockbots.motor), 3`  
`still () (spockbots.motor.SpockbotsMotor method), 4`  
`still () (spockbots.systemgyro.Gyro method), 5`  
`stop () (spockbots.motor.SpockbotsMotor method), 4`

## T

`test () (spockbots.colorsensor.SpockbotsColorSensors method), 3`  
`test () (spockbots.systemgyro.Gyro method), 5`  
`tunrtoblack () (spockbots.motor.SpockbotsMotor method), 4`  
`turn () (spockbots.motor.SpockbotsMotor method), 4`

## V

`value () (spockbots.colorsensor.SpockbotsColorSensor method), 3`  
`value () (spockbots.colorsensor.SpockbotsColorSensors method), 3`  
`voltage () (in module spockbots.output), 4`

## W

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

`write () (spockbots.colorsensor.SpockbotsColorSensors method), 3`  
`writefile () (in module spockbots.output), 4`