Spockbots

Release 1.0

Spockbots

Nov 06, 2019

CONTENTS

CHAPTER

ONE

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 muve 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 containes 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 form 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

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

s

INDEX

S

T

V

W