

---

# **OPSORO**

*Release 1.0*

**Aug 18, 2017**



---

Contents:

---

<b>1</b>	<b>Development</b>	<b>1</b>
<b>2</b>	<b>Commenting</b>	<b>3</b>
<b>3</b>	<b>OPSORO OS</b>	<b>5</b>
3.1	Modules . . . . .	5
3.2	Module contents . . . . .	23
<b>4</b>	<b>Indices and tables</b>	<b>25</b>
	<b>Python Module Index</b>	<b>27</b>



# CHAPTER 1

---

Development

---



## CHAPTER 2

---

### Commenting

---

```
"""  
Explanation about the function.  
  
:param bool A:    bool input parameter  
:param int B:     int input parameter  
  
:return:         what is the function returning?  
:rtype:          return type  
"""
```





## Modules

### Internal Modules

#### opsoro.apps

`opsoro.apps.constrain` (*n*, *minn*, *maxn*)

#### opsoro.dof

**class** `opsoro.dof.DOF` (*name*, *neutral=0.0*, *poly=None*)

Bases: `object`

**\_\_init\_\_** (*name*, *neutral=0.0*, *poly=None*)

DOF class.

#### Parameters

- **name** (*string*) – name of the DOF.
- **neutral** (*float*) – neutral dof position.
- **poly** (*list*) – 20 dof values linked to emotions.

**calc** (*r*, *phi*, *anim\_time=-1*)

Calculate dof value with the polygon, according to the given *r* and *phi*.

#### Parameters

- **r** (*float*) – radius *r*, intensity of the emotion.
- **phi** (*float*) – (radians) angle of the emotion in the circumplex.
- **anim\_time** (*float*) – time for the servo to move from previous dof to the new dof (-1: animation will be based on dof differences).

**config** (\*\*args)

**reset\_overlay** (anim\_time=-1)

Clears the overlay value and resets the dof position to the last set value.

**Parameters** **anim\_time** (*float*) – time for the servo to move from previous dof to the new dof (-1: animation will be based on dof differences).

**set\_control\_polygon** (neutral=0.0, poly=None)

Sets the control polygon, 20 dof values are linked to certain emotions.

**Parameters**

- **neutral** (*float*) – neutral dof position.
- **poly** (*list*) – 20 dof values linked to emotions.

**set\_overlay\_value** (dof\_value=0, anim\_time=-1, update\_last\_set\_time=True)

Sets the overlay value and overwrites the dof position.

**Parameters**

- **dof\_value** (*float*) – new overlay value of the dof.
- **anim\_time** (*float*) – time for the servo to move from previous dof to the new dof (-1: animation will be based on dof differences).
- **update\_last\_set\_time** (*bool*) – update the last set timer of the dof.

**set\_value** (dof\_value=0, anim\_time=-1, is\_overlay=False, update\_last\_set\_time=True)

Sets the dof value. If the dof value is 2 or larger, set it to a random value.

**Parameters**

- **dof\_value** (*float*) – new value of the dof.
- **anim\_time** (*float*) – time for the servo to move from previous dof to the new dof (-1: animation will be based on dof differences).
- **is\_overlay** (*bool*) – used to determine what priority the dof value has (overlay > default).
- **update\_last\_set\_time** (*bool*) – update the last set timer of the dof.

**update** ()

Updates the dof value according to the animation.

**Returns** True if dof value is updated, False if dof value did not change.

**Return type** bool

opsoro.dof.**constrain** (n, minn, maxn)

## opsoro.dof.servo

**class** opsoro.dof.servo.**Servo** (name, neutral=0.0, poly=None)

Bases: *opsoro.dof.DOF*

**config** (pin=None, min\_range=0, mid\_pos=1500, max\_range=0)

Helper class to turn DOF positions into pulse widths for the servo controller.

**Parameters**

- **pin** (*int*) – Servo pin number

- **min\_range** (*int*) – Minimum range of the servo, can be positive or negative. When  $dof\_pos < 0$ , pulse width =  $mid\_pos + dof\_pos * min\_range$
- **mid\_pos** (*int*) – Pulse width when neutral (DOF position = 0).
- **max\_range** (*int*) – Maximum range of the servo, can be positive or negative. When  $dof\_pos > 0$ , pulse width =  $mid\_pos + dof\_pos * max\_range$

**to\_us** (*dof\_value=None*)

Converts DOF pos to microseconds.

**Parameters** **dof\_value** (*float*) – value to convert to us. If None; dof value of servo object is used

**Returns** servo value (us)

**Return type** int

**update** ()

Updates the servo with the setted dof value.

**Returns** True if dof value is updated, False if dof value did not change

**Return type** bool

`opsoro.dof.servo.constrain` (*n, minn, maxx*)

## opsoro.hardware

This module defines the interface for communicating with the shield.

**class** `opsoro.hardware._Hardware`

Bases: object

**\_\_init\_\_** ()

Hardware class, used to communicate with the shield.

**led\_off** ()

Turns status LED off.

**led\_on** ()

Turns status LED on.

**ping** ()

Returns True if OPSOROHAT rev3 is connected.

**Returns** True if shield is connected

**Return type** bool

**reset** ()

Resets the ATmega328, MPR121 and PCA9685.

## opsoro.hardware.analog

**class** `opsoro.hardware.analog.Analog`

Bases: object

**read\_all\_channels** ()

Reads all analog channels and returns them as a list.

**Returns** analog values

**Return type** list

**read\_channel** (*channel*)

Reads the value of a single analog channel.

**Parameters** **channel** (*int*) – analog channel to read

**Returns** analog value of the channel

**Return type** var

## opsoro.hardware.capacitive

**class** opsoro.hardware.capacitive.**Capacitive**

Bases: object

**get\_baseline\_data** ()

Get list of electrode baseline data. Result is 10 bits, but the 2 least significant bits are set to 0.

**Returns** electrode baseline data (10 bits).

**Return type** list

**get\_filtered\_data** ()

Get list of electrode filtered data (10 bits per electrode).

**Returns** electrode filtered data (10 bits per electrode).

**Return type** list

**get\_touched** ()

Returns the values of the touch registers, each bit corresponds to one electrode.

**Returns** values of the touch registers,

**Return type** list

**init** (*electrodes*, *gpios=0*, *autoconfig=True*)

Initialize the MPR121 capacitive touch sensor.

**Parameters**

- **electrodes** (*int*) – amount of electrodes
- **gpios** (*int*) – amount of gpios
- **autoconfig** (*bool*) –

**read\_gpio** ()

Returns the status of all GPIO channels, each bit corresponds to one gpio channel.

**Returns** status of all GPIO channels.

**Return type** list

**set\_gpio\_pinmode** (*gpio*, *pinmode*)

Sets a GPIO channel's pin mode.

**Parameters**

- **gpio** (*int*) – gpio channel
- **pinmode** (*int*) – pinmode to set

**set\_threshold** (*electrode*, *touch*, *release*)

Set an electrode's touch and release threshold.

**Parameters**

- **electrode** (*int*) – index of electrode
- **touch** (*int*) – threshold value for touch detection
- **release** (*int*) – threshold value for release detection

**write\_gpio** (*gpio*, *data*)

Set GPIO channel value.

**Parameters**

- **gpio** (*int*) – gpio channel
- **data** (*int*) – data to write to gpio channel.

**opsoro.hardware.dummy\_spidev**

**class** opsoro.hardware.dummy\_spidev.**SpiDev**

Bases: object

**\_\_init\_\_** ()

**open** (\*args)

**xfer2** (\*args)

**opsoro.hardware.i2c**

**class** opsoro.hardware.i2c.**I2C**

Bases: object

**detect** (*addr*)

Returns True if an I2C device is found at a particular address.

**Parameters** **addr** (*int*) – address of the I2C device.

**Returns** I2C device detected

**Return type** bool

**read16** (*addr*, *reg*)

Read 2 bytes from an I2C device.

**Parameters**

- **addr** (*int*) – address of the I2C device.
- **reg** (*int*) – register address in the I2C device

**Returns** 2 Bytes

**Return type** var

**read8** (*addr*, *reg*)

Read a Byte from an I2C device.

**Parameters**

- **addr** (*int*) – address of the I2C device.
- **reg** (*int*) – register address in the I2C device

**Returns** what is the function returning?

**Return type** *var*

**write16** (*addr, reg, data*)

Write 2 bytes to an I2C device.

**Parameters**

- **addr** (*int*) – address of the I2C device.
- **reg** (*int*) – register address in the I2C device
- **data** (*var*) – Bytes to send

**write8** (*addr, reg, data*)

Write a Byte to an I2C device.

**Parameters**

- **addr** (*int*) – address of the I2C device.
- **reg** (*int*) – register address in the I2C device
- **data** (*var*) – Byte to send

## opsoro.hardware.neopixel

**class** opsoro.hardware.neopixel.**Neopixel**

Bases: object

**disable** ()

Turns off the NeoPixel MOSFET, disabling the NeoPixels. Data is lost when pixels are disabled.

**enable** ()

Turns on the NeoPixel MOSFET, enabling the NeoPixels. Data is lost when pixels are disabled, so call show() again afterwards.

**init** (*num\_leds*)

Initialize the NeoPixel library.

**Parameters** **num\_leds** (*int*) – number of neopixel leds.

**set\_all** (*r, g, b*)

Set the color of the entire strip.

**Parameters**

- **r** (*int*) – red color value (0-255)
- **g** (*int*) – green color value (0-255)
- **b** (*int*) – blue color value (0-255)

**set\_all\_hsv** (*h, s, v*)

Set the HSV color of the entire strip.

**Parameters**

- **h** (*int*) – hue color value (0-255)
- **s** (*int*) – saturation color value (0-255)
- **v** (*int*) – value color value (0-255)

**set\_brightness** (*brightness*)

Set the NeoPixel's global brightness, 0-255.

**Parameters** `brightness` (*int*) – brightness to set (0-255)

**set\_pixel** (*pixel*, *r*, *g*, *b*)

Set the color of a single pixel.

**Parameters**

- **pixel** (*int*) – pixel index
- **r** (*int*) – red color value (0-255)
- **g** (*int*) – green color value (0-255)
- **b** (*int*) – blue color value (0-255)

**set\_pixel\_hsv** (*pixel*, *h*, *s*, *v*)

Set the HSV color of a single pixel.

**Parameters**

- **pixel** (*int*) – pixel index
- **h** (*int*) – hue color value (0-255)
- **s** (*int*) – saturation color value (0-255)
- **v** (*int*) – value color value (0-255)

**set\_range** (*start*, *end*, *r*, *g*, *b*)

Set the color of a range of pixels.

**Parameters**

- **start** (*int*) – start index of led range
- **end** (*int*) – end index of led range
- **r** (*int*) – red color value (0-255)
- **g** (*int*) – green color value (0-255)
- **b** (*int*) – blue color value (0-255)

**set\_range\_hsv** (*start*, *end*, *h*, *s*, *v*)

Set the HSV color of a range of pixels.

**Parameters**

- **start** (*int*) – start index of led range
- **end** (*int*) – end index of led range
- **h** (*int*) – hue color value (0-255)
- **s** (*int*) – saturation color value (0-255)
- **v** (*int*) – value color value (0-255)

**show** ()

Sends the pixel data from the ATmega328 to the NeoPixels.

### opsoro.hardware.servo

**class** opsoro.hardware.servo.Servo

Bases: object

**disable ()**  
Turns off the servo power MOSFET, disabling all servos.

**enable ()**  
Turns on the servo power MOSFET, enabling all servos.

**init ()**  
Set up the PCA9685 for driving servos.

**neutral ()**  
Set all servos to 1500us.

**set (channel, pos)**  
Set the position of one servo. Pos in us, 500 to 2500

**Parameters**

- **channel** (*int*) – channel of the servo
- **pos** (*int*) – position of the servo (500 to 2500)

**set\_all (pos\_list)**  
Set position of all 16 servos using a list.

**Parameters pos\_list** (*list*) – list of servo positions

**set\_all\_us (us)**  
Set all servos to a certain position (us)

**Parameters us** (*int*) – position in us

### opsoro.hardware.spi

This module defines the interface for communicating with SPI.

**class** opsoro.hardware.spi.\_SPI

Bases: object

**\_\_init\_\_ ()**  
SPI class, used to communicate with the shield.

**command (cmd, params=None, returned=0, delay=0)**

Send a command over the SPI bus to the ATmega328. Optionally reads the result buffer and returns those Bytes.

**Parameters**

- **cmd** (*string*) – spi command
- **params** (*strin*) – parameters for the command
- **returned** (*int*) – size of result reading
- **delay** (*int*) – delay between sending the command and reading the result

**Returns** result buffer (Bytes)

**Return type** list



**opsoro.hardware.usb\_serial**

`opsoro.hardware.usb_serial`  
alias of `opsoro.hardware.usb_serial`

**opsoro.module**

**class** `opsoro.module.Module` (*data=None*)

Bases: `object`

**\_\_init\_\_** (*data=None*)

Module default class. Custom modules should inherit this class and can override functions.

**Parameters** `data` (*dict*) – configuration data to setup the module

**alive\_trigger** (*count\_seed=1*)

This is triggered frequently, when the aliveness is turned on.

**Parameters** `count_seed` (*float*) – seed value for randomization

**Returns** True if the module updated something

**Return type** `bool`

**apply\_poly** (*r, phi, anim\_time=-1*)

Apply poly values `r` and `phi` to the module and calculate dof values

**Parameters**

- `r` (*float*) – `r` radius value
- `phi` (*float*) – `phi` angle value
- `anim_time` (*int*) – animation time in ms

**load\_module** (*data*)

Setup modules with given configuration data

**Parameters** `data` (*dict*) – configuration data to setup the module

**set\_dof** (*tags=[], value=0, anim\_time=-1*)

Set the value of a dof with the given tags. If no tags are provided, all dofs are set with the given value.

**Parameters**

- `tags` (*list*) – name of the DOF
- `value` (*float*) – value to set the DOF
- `anim_time` (*int*) – animation time in ms

**set\_dof\_value** (*dof\_name, value, anim\_time=-1*)

Set the value of a dof with the given name. If no name is provided, all dofs are set with the given value.

**Parameters**

- `dof_name` (*string*) – name of the DOF
- `value` (*float*) – value to set the DOF
- `anim_time` (*int*) – animation time in ms

**update** ()

Update all dof values of this module and return if the update changed a dof.

**Returns** True if a dof has been updated

**Return type** bool

`opsoro.module.constrain` (*n*, *minn*, *maxn*)

### opsoro.module.eye

**class** `opsoro.module.eye.Eye` (*data=None*)

Bases: `opsoro.module.Module`

`__init__` (*data=None*)

Eye module class inherits default module class.

**Parameters** `data` (*dict*) – configuration data to setup the module

**alive\_trigger** (*count\_seed*)

This is triggered frequently, when the aliveness is turned on.

**Parameters** `count_seed` (*float*) – seed value for randomization

**Returns** True if the module updated something

**Return type** bool

**blink** (*anim\_time=0.4*)

Triggers the eye to blink

**Parameters** `anim_time` (*float*) – animation time to perform the blinking action

**Returns** True if the module updated something

**Return type** bool

**look** (*x=0, y=0, z=0*)

Look function to make the eye look at some point in space.

**Parameters**

- `x` (*float*) – x position / horizontal
- `y` (*float*) – y position / vertical
- `z` (*float*) – z position / depth

**Returns** True if the module updated something

**Return type** bool

### opsoro.module.eyebrow

**class** `opsoro.module.eyebrow.Eyebrow` (*data=None*)

Bases: `opsoro.module.Module`

### opsoro.module.mouth

**class** `opsoro.module.mouth.Mouth` (*data=None*)

Bases: `opsoro.module.Module`

## opsoro.module.turn

**class** opsoro.module.turn.**Turn** (*data=None*)  
 Bases: *opsoro.module.Module*

## opsoro.preferences

This module defines the interface for communicating with the settings of the robot.

**class** opsoro.preferences.**\_Preferences**  
 Bases: object

**\_\_init\_\_** ()  
 Preferences class to store and retrieve settings.

**apply\_prefs** (*update\_audio=False, update\_wireless=False, restart\_wireless=False, update\_dns=False*)  
 Apply preferences to the system.

### Parameters

- **update\_audio** (*bool*) – True if audio settings have changed and needs to update.
- **update\_wireless** (*bool*) – True if wireless settings have changed and the wireless interface needs to update.
- **restart\_wireless** (*bool*) – True if wireless settings have changed and the wireless interface needs to restart.
- **update\_dns** (*bool*) – True if DNS settings have changed and needs to update.

**get** (*section, item, default*)  
 Retrieve preference value.

### Parameters

- **section** (*string*) – category in which the item is defined.
- **item** (*string*) – item to retrieve.
- **default** – default value to return if the value is not available.

**Returns** preference value

**load\_prefs** ()  
 Load preferences into data.

**save\_prefs** ()  
 Saves preferences to yaml file.

**set** (*section, item, value*)  
 Set preference value.

### Parameters

- **section** (*string*) – category in which the item is defined.
- **item** (*string*) – item to set.
- **value** – value to set.

opsoro.preferences.**constrain** (*n, minn, maxx*)

## opsoro.server

This module defines the interface for the Server.

```
class opsoro.server.Server
    Bases: object
    __init__()
    app_api(f)
    app_view(f)
    at_exit()
    protected_view(f)
    render_template(template, **kwargs)
    run()
    shutdown()
```

```
class opsoro.server.Server
    Bases: object
    __init__()
    app_api(f)
    app_view(f)
    at_exit()
    protected_view(f)
    render_template(template, **kwargs)
    run()
    shutdown()
```

## opsoro.server.request\_handlers

This module defines the interface for the request handling.

```
class opsoro.server.request_handlers.RHandler(server)
    Bases: object
    __init__(server)
    inject_opsoro_vars()
    page_blockly()
    page_closeapp(appname)
    page_file_list()
    page_index()
    page_login()
    page_logout()
    page_openapp(appname)
    page_restart()
```

```

page_shutdown ()
page_sockjstoken ()
page_virtual ()
render_template (template, **kwargs)
set_urls ()
show_errormessage (error)
sound_data ()

```

```
class opsoro.server.request_handlers.RHandler (server)
```

```
    Bases: object
```

```

    __init__ (server)
    inject_opsoro_vars ()
    page_blockly ()
    page_closeapp (appname)
    page_file_list ()
    page_index ()
    page_login ()
    page_logout ()
    page_openapp (appname)
    page_restart ()
    page_shutdown ()
    page_sockjstoken ()
    page_virtual ()
    render_template (template, **kwargs)
    set_urls ()
    show_errormessage (error)
    sound_data ()

```

### opsoro.server.request\_handlers.opsoro\_data\_requests

```

opsoro.server.request_handlers.opsoro_data_requests.config_expressions_data ()
opsoro.server.request_handlers.opsoro_data_requests.config_robot_data ()
opsoro.server.request_handlers.opsoro_data_requests.constrain (n, minn, maxx)
opsoro.server.request_handlers.opsoro_data_requests.docs_file_data (app_name=None)
opsoro.server.request_handlers.opsoro_data_requests.docs_file_delete (app_name)
opsoro.server.request_handlers.opsoro_data_requests.docs_file_list ()
opsoro.server.request_handlers.opsoro_data_requests.docs_file_save (app_name)
opsoro.server.request_handlers.opsoro_data_requests.robot_dof_data ()

```

```
opsoro.server.request_handlers.opsoro_data_requests.robot_dofs_data()
opsoro.server.request_handlers.opsoro_data_requests.robot_emotion()
opsoro.server.request_handlers.opsoro_data_requests.robot_servo()
opsoro.server.request_handlers.opsoro_data_requests.robot_servos()
opsoro.server.request_handlers.opsoro_data_requests.robot_sound()
opsoro.server.request_handlers.opsoro_data_requests.robot_stop()
opsoro.server.request_handlers.opsoro_data_requests.robot_tts()
```

## **opsoro.sound**

This module defines the interface for communicating with the sound module.

**class** opsoro.sound.\_**Sound**

Bases: object

**\_\_init\_\_**()

Sound class, used to play sound and speak text.

**get\_file**(filename, tts=False)

Returns audio file data according to the given filename.

**Parameters** filename (*string*) – file to return the data from

**Returns** Soundfile data.

**Return type** var

**play\_file**(filename)

Plays an audio file according to the given filename.

**Parameters** filename (*string*) – file to play

**Returns** True if sound is playing.

**Return type** bool

**say\_tts**(text, generate\_only=False)

Converts a string to a soundfile using Text-to-Speech libraries

**Parameters**

- **text** (*string*) – text to convert to speech
- **generate\_only** (*bool*) – do not play the soundfile once it is created

**stop\_sound**()

Stop the played sound.

**wait\_for\_sound**()

Wait until the played sound is done.

## **opsoro.sound.tts**

This module defines the interface for communicating with the TTS libraries.

**class** opsoro.sound.tts.\_**TTS**

Bases: object

`__init__()`

TTS class, used to convert text to speech.

`create(text)`

Takes a string of text, converts it using the PicoTTS engine, and plays it. Wave files are buffered in `/tmp/OnoTTS/<text>.wav`. First call blocks while PicoTTS generates the `.wav`, this may take about a second. Subsequent calls of the same text return immediately. If you wish to avoid this, sound files can be generated on beforehand by using `generate_only=True`.

**Parameters** `text` (*string*) – text to convert to speech

**Returns** path to the sound file

**Return type** string

`create_espeak(text, file_path, language, gender, delay, speed)`

Convert text to speech using the espeak TTS library.

**Parameters**

- **text** (*string*) – text to convert to speech
- **file\_path** (*string*) – file path to store the speech soundfile
- **language** (*string*) – language initials
- **gender** (*string*) – specify gender (m for male, f for female)
- **delay** (*int*) – delay between words in ms
- **speed** (*int*) – speed in words-per-minute

`create_pico(text, file_path)`

Convert text to speech using the pico2wave TTS library.

**Parameters**

- **text** (*string*) – text to convert to speech
- **file\_path** (*string*) – file path to store the speech soundfile

## opsoro.animate

This module defines the interface for animating an expression.

`class opsoro.animate.Animate(times, values)`

Bases: object

`__init__(times, values)`

Class to facilitate the tweening of values in time. The animation starts when the object is created. Once ended, the call method will return the last item in values.

**Parameters**

- **times** (*list*) – A list of timestamps in seconds, in increasing order. Timestamp 0 is the moment the Animate object was created.
- **values** (*list*) – A list of numerical values associated with timestamps. First element should be 0.

`has_ended()`

Returns true if the animation has ended.

`class opsoro.animate.Animate(times, values)`

Bases: object

`__init__(times, values)`

Class to facilitate the tweening of values in time. The animation starts when the object is created. Once ended, the call method will return the last item in values.

**Parameters**

- **times** (*list*) – A list of timestamps in seconds, in increasing order. Timestamp 0 is the moment the Animate object was created.
- **values** (*list*) – A list of numerical values associated with timestamps. First element should be 0.

`has_ended()`

Returns true if the animation has ended.

`class opsoro.animate.AnimatePeriodic(times, values)`

Bases: object

`__init__(times, values)`

Class to facilitate the tweening of values in time. The animation starts when the object is created. This class is a variant of the Animate class that does not end, but instead repeats its pattern indefinitely.

**Parameters**

- **times** (*list*) – A list of timestamps in seconds, in increasing order. Timestamp 0 is the moment the Animate object was created.
- **values** (*list*) – A list of numerical values associated with timestamps. First element should be 0.

**opsoro.console\_msg**

`opsoro.console_msg.print_apploaded(appname)`

`opsoro.console_msg.print_appstarted(appname)`

`opsoro.console_msg.print_appstopped(appname)`

`opsoro.console_msg.print_error(msg)`

`opsoro.console_msg.print_info(msg)`

`opsoro.console_msg.print_spi(msg)`

`opsoro.console_msg.print_warning(msg)`

**opsoro.expression**

This module defines the interface for communicating with the expression.

`class opsoro.expression._Expression`

Bases: object

`__init__()`

`get_emotion_complex()`

Returns current emotion as a complex number

**Returns** current emotion

**Return type** complex



**load\_config** (*file\_name='robot\_expressions.conf'*)

Load expressions from a expressions configurations file

**Parameters** **file\_name** (*string*) – name of the config file

**Returns** True if file is successfully loaded

**Return type** bool

**save\_config** (*file\_name='robot\_expressions.conf'*)

Save the current expressions configurations

**Parameters** **file\_name** (*string*) – name of the config file

**Returns** True if file is successfully saved

**Return type** bool

**set\_config** (*config=None*)

**set\_emotion\_e** (*e=0j, anim\_time=-1*)

Set an emotion with complex number e, within a certain time.

**Parameters**

- **e** (*complex*) – complex number e
- **anim\_time** (*float*) – time to set the emotion

**set\_emotion\_icon** (*icon, anim\_time=-1*)

Set an emotion with icon if defined in expression list, within a certain time.

**Parameters**

- **icon** (*string*) – name of the icon to set
- **anim\_time** (*float*) – time to set the emotion

**set\_emotion\_index** (*index, anim\_time=-1*)

Set an emotion with index in defined expression list, within a certain time.

**Parameters**

- **index** (*integer*) – index of the emotion in the list of emotions
- **anim\_time** (*float*) – time to set the emotion

**set\_emotion\_name** (*name, anim\_time=-1*)

Set an emotion with name if defined in expression list, within a certain time.

**Parameters**

- **name** (*string*) – name of the emotion to set
- **anim\_time** (*float*) – time to set the emotion

**set\_emotion\_r\_phi** (*r, phi, degrees=False, anim\_time=-1*)

Set an emotion with r and phi, within a certain time.

**Parameters**

- **r** (*float*) – radius of the circumplex
- **phi** (*float*) – angle of the circumplex
- **degrees** (*bool*) – is conversion to radians needed?
- **anim\_time** (*float*) – time to set the emotion

**set\_emotion\_random** (*all\_random=True, anim\_time=-1*)

Set an emotion with random index in defined expression list, within a certain time. Or set all dofs to a random position between -1 and 1.

**Parameters**

- **all\_random** (*bool*) – all dofs random or not
- **anim\_time** (*float*) – time to set the emotion

**set\_emotion\_val\_ar** (*valence, arousal, anim\_time=-1*)

Set an emotion with valence and arousal, within a certain time.

**Parameters**

- **valence** (*float*) – valence
- **arousal** (*float*) – arousal
- **anim\_time** (*float*) – time to set the emotion

**update** ()

Old function, not used in new system

**Returns** nothing

**Return type** None

`opsoro.expression.constrain` (*n, minn, maxx*)

## opsoro.robot

This module defines the interface for communicating with the robot.

**class** `opsoro.robot._Robot`

Bases: `object`

`__init__` ()

`alive_loop` ()

`apply_poly` (*r, phi, anim\_time=-1*)

`blink` (*speed*)

`dof_update_loop` ()

`get_dof_values` (*current=True*)

`load_config` (*file\_name='robot\_config.conf'*)

`save_config` (*file\_name='robot\_config.conf'*)

`set_config` (*config=None*)

`set_dof` (*tags=[], value=0, anim\_time=-1*)

`set_dof_list` (*dof\_values, anim\_time=-1*)

`set_dof_value` (*module\_name, dof\_name, dof\_value, anim\_time=-1*)

`set_dof_values` (*dof\_values, anim\_time=-1*)

`sleep` ()

`start` (*alive=True*)

`start_alive_loop` ()

```
start_update_loop()  
stop()  
stop_alive_loop()  
stop_update_loop()  
update()  
wake()
```

### **opsoro.stoppable\_thread**

```
class opsoro.stoppable_thread.StoppableThread(*args, **kwargs)
```

```
    Bases: threading.Thread
```

Thread class with a stop() method. The thread itself has to check regularly for the stopped() condition.

```
    __init__(*args, **kwargs)
```

```
    sleep(secs)
```

```
    stop()
```

```
    stopped()
```

## **Module contents**

```
opsoro.main()
```

```
opsoro.sigterm_handler(_signo, _stack_frame)
```



## CHAPTER 4

---

### Indices and tables

---

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



### O

- opsoro, 23
- opsoro.animate, 19
- opsoro.apps, 5
- opsoro.console\_msg, 20
- opsoro.dof, 5
- opsoro.dof.servo, 6
- opsoro.expression, 20
- opsoro.hardware, 7
  - opsoro.hardware.analog, 7
  - opsoro.hardware.capacitive, 8
  - opsoro.hardware.dummy\_spidev, 9
  - opsoro.hardware.i2c, 9
  - opsoro.hardware.neopixel, 10
  - opsoro.hardware.servo, 11
  - opsoro.hardware.spi, 12
- opsoro.module, 13
  - opsoro.module.eye, 14
  - opsoro.module.eyebrow, 14
  - opsoro.module.mouth, 14
  - opsoro.module.turn, 15
- opsoro.preferences, 15
- opsoro.robot, 22
- opsoro.server, 16
  - opsoro.server.request\_handlers, 16
  - opsoro.server.request\_handlers.opsoro\_data\_requests, 17
- opsoro.sound, 18
  - opsoro.sound.tts, 18
- opsoro.stoppable\_thread, 23





## Symbols

\_Expression (class in opsoro.expression), 20  
 \_Hardware (class in opsoro.hardware), 7  
 \_Preferences (class in opsoro.preferences), 15  
 \_Robot (class in opsoro.robot), 22  
 \_SPI (class in opsoro.hardware.spi), 12  
 \_Sound (class in opsoro.sound), 18  
 \_TTS (class in opsoro.sound.tts), 18  
 \_\_init\_\_() (opsoro.animate.Animate method), 19  
 \_\_init\_\_() (opsoro.animate.AnimatePeriodic method), 20  
 \_\_init\_\_() (opsoro.dof.DOF method), 5  
 \_\_init\_\_() (opsoro.expression.\_Expression method), 20  
 \_\_init\_\_() (opsoro.hardware.\_Hardware method), 7  
 \_\_init\_\_() (opsoro.hardware.dummy\_spidev.SpiDev method), 9  
 \_\_init\_\_() (opsoro.hardware.spi.\_SPI method), 12  
 \_\_init\_\_() (opsoro.module.Module method), 13  
 \_\_init\_\_() (opsoro.module.eye.Eye method), 14  
 \_\_init\_\_() (opsoro.preferences.\_Preferences method), 15  
 \_\_init\_\_() (opsoro.robot.\_Robot method), 22  
 \_\_init\_\_() (opsoro.server.Server method), 16  
 \_\_init\_\_() (opsoro.server.request\_handlers.RHandler method), 16, 17  
 \_\_init\_\_() (opsoro.sound.\_Sound method), 18  
 \_\_init\_\_() (opsoro.sound.tts.\_TTS method), 18  
 \_\_init\_\_() (opsoro.stoppable\_thread.StoppableThread method), 23

## A

alive\_loop() (opsoro.robot.\_Robot method), 22  
 alive\_trigger() (opsoro.module.eye.Eye method), 14  
 alive\_trigger() (opsoro.module.Module method), 13  
 Analog (class in opsoro.hardware.analog), 7  
 Animate (class in opsoro.animate), 19  
 AnimatePeriodic (class in opsoro.animate), 20  
 app\_api() (opsoro.server.Server method), 16  
 app\_view() (opsoro.server.Server method), 16  
 apply\_poly() (opsoro.module.Module method), 13  
 apply\_poly() (opsoro.robot.\_Robot method), 22

apply\_prefs() (opsoro.preferences.\_Preferences method), 15

at\_exit() (opsoro.server.Server method), 16

## B

blink() (opsoro.module.eye.Eye method), 14  
 blink() (opsoro.robot.\_Robot method), 22

## C

calc() (opsoro.dof.DOF method), 5  
 Capacitive (class in opsoro.hardware.capacitive), 8  
 command() (opsoro.hardware.spi.\_SPI method), 12  
 config() (opsoro.dof.DOF method), 5  
 config() (opsoro.dof.servo.Servo method), 6  
 config\_expressions\_data() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17  
 config\_robot\_data() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17  
 constrain() (in module opsoro.apps), 5  
 constrain() (in module opsoro.dof), 6  
 constrain() (in module opsoro.dof.servo), 7  
 constrain() (in module opsoro.expression), 22  
 constrain() (in module opsoro.module), 14  
 constrain() (in module opsoro.preferences), 15  
 constrain() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17  
 create() (opsoro.sound.tts.\_TTS method), 19  
 create\_espeak() (opsoro.sound.tts.\_TTS method), 19  
 create\_pico() (opsoro.sound.tts.\_TTS method), 19

## D

detect() (opsoro.hardware.i2c.I2C method), 9  
 disable() (opsoro.hardware.neopixel.Neopixel method), 10  
 disable() (opsoro.hardware.servo.Servo method), 11

docs\_file\_data() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17

docs\_file\_delete() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17

docs\_file\_list() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17

docs\_file\_save() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17

DOF (class in opsoro.dof), 5

dof\_update\_loop() (opsoro.robot.\_Robot method), 22

## E

enable() (opsoro.hardware.neopixel.Neopixel method), 10

enable() (opsoro.hardware.servo.Servo method), 12

Eye (class in opsoro.module.eyebrow), 14

Eyebrow (class in opsoro.module.eyebrow), 14

## G

get() (opsoro.preferences.\_Preferences method), 15

get\_baseline\_data() (opsoro.hardware.capacitive.Capacitive method), 8

get\_dof\_values() (opsoro.robot.\_Robot method), 22

get\_emotion\_complex() (opsoro.expression.\_Expression method), 20

get\_file() (opsoro.sound.\_Sound method), 18

get\_filtered\_data() (opsoro.hardware.capacitive.Capacitive method), 8

get\_touched() (opsoro.hardware.capacitive.Capacitive method), 8

## H

has\_ended() (opsoro.animate.Animate method), 19, 20

## I

I2C (class in opsoro.hardware.i2c), 9

init() (opsoro.hardware.capacitive.Capacitive method), 8

init() (opsoro.hardware.neopixel.Neopixel method), 10

init() (opsoro.hardware.servo.Servo method), 12

inject\_opsoro\_vars() (opsoro.server.request\_handlers.RHandler method), 16, 17

## L

led\_off() (opsoro.hardware.\_Hardware method), 7

led\_on() (opsoro.hardware.\_Hardware method), 7

load\_config() (opsoro.expression.\_Expression method), 20

load\_config() (opsoro.robot.\_Robot method), 22

load\_module() (opsoro.module.Module method), 13

load\_prefs() (opsoro.preferences.\_Preferences method), 15

look() (opsoro.module.eyebrow.Eyebrow method), 14

## M

main() (in module opsoro), 23

Module (class in opsoro.module), 13

Mouth (class in opsoro.module.mouth), 14

## N

Neopixel (class in opsoro.hardware.neopixel), 10

neutral() (opsoro.hardware.servo.Servo method), 12

## O

open() (opsoro.hardware.dummy\_spidev.SpiDev method), 9

opsoro (module), 23

opsoro.animate (module), 19

opsoro.apps (module), 5

opsoro.console\_msg (module), 20

opsoro.dof (module), 5

opsoro.dof.servo (module), 6

opsoro.expression (module), 20

opsoro.hardware (module), 7

opsoro.hardware.analog (module), 7

opsoro.hardware.capacitive (module), 8

opsoro.hardware.dummy\_spidev (module), 9

opsoro.hardware.i2c (module), 9

opsoro.hardware.neopixel (module), 10

opsoro.hardware.servo (module), 11

opsoro.hardware.spi (module), 12

opsoro.module (module), 13

opsoro.module.eyebrow (module), 14

opsoro.module.eyebrow (module), 14

opsoro.module.mouth (module), 14

opsoro.module.turn (module), 15

opsoro.preferences (module), 15

opsoro.robot (module), 22

opsoro.server (module), 16

opsoro.server.request\_handlers (module), 16

opsoro.server.request\_handlers.opsoro\_data\_requests (module), 17

opsoro.sound (module), 18

opsoro.sound.tts (module), 18

opsoro.stoppable\_thread (module), 23

## P

page\_blockly() (opsoro.server.request\_handlers.RHandler method), 16, 17

page\_closeapp() (opsoro.server.request\_handlers.RHandler method), 16, 17

page\_file\_list() (opsoro.server.request\_handlers.RHandler method), 16, 17  
 page\_index() (opsoro.server.request\_handlers.RHandler method), 16, 17  
 page\_login() (opsoro.server.request\_handlers.RHandler method), 16, 17  
 page\_logout() (opsoro.server.request\_handlers.RHandler method), 16, 17  
 page\_openapp() (opsoro.server.request\_handlers.RHandler method), 16, 17  
 page\_restart() (opsoro.server.request\_handlers.RHandler method), 16, 17  
 page\_shutdown() (opsoro.server.request\_handlers.RHandler method), 16, 17  
 page\_sockjstoken() (opsoro.server.request\_handlers.RHandler method), 17  
 page\_virtual() (opsoro.server.request\_handlers.RHandler method), 17  
 ping() (opsoro.hardware.\_Hardware method), 7  
 play\_file() (opsoro.sound.\_Sound method), 18  
 print\_apploaded() (in module opsoro.console\_msg), 20  
 print\_appstarted() (in module opsoro.console\_msg), 20  
 print\_appstopped() (in module opsoro.console\_msg), 20  
 print\_error() (in module opsoro.console\_msg), 20  
 print\_info() (in module opsoro.console\_msg), 20  
 print\_spi() (in module opsoro.console\_msg), 20  
 print\_warning() (in module opsoro.console\_msg), 20  
 protected\_view() (opsoro.server.Server method), 16

## R

read16() (opsoro.hardware.i2c.I2C method), 9  
 read8() (opsoro.hardware.i2c.I2C method), 9  
 read\_all\_channels() (opsoro.hardware.analog.Analog method), 7  
 read\_channel() (opsoro.hardware.analog.Analog method), 8  
 read\_gpio() (opsoro.hardware.capacitive.Capacitive method), 8  
 render\_template() (opsoro.server.request\_handlers.RHandler method), 17  
 render\_template() (opsoro.server.Server method), 16  
 reset() (opsoro.hardware.\_Hardware method), 7  
 reset\_overlay() (opsoro.dof.DOF method), 6  
 RHandler (class in opsoro.server.request\_handlers), 16, 17  
 robot\_dof\_data() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17  
 robot\_dofs\_data() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 17  
 robot\_emotion() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 18  
 robot\_servo() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 18  
 robot\_servos() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 18  
 robot\_sound() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 18  
 robot\_stop() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 18  
 robot\_tts() (in module opsoro.server.request\_handlers.opsoro\_data\_requests), 18  
 run() (opsoro.server.Server method), 16

## S

save\_config() (opsoro.expression.\_Expression method), 21  
 save\_config() (opsoro.robot.\_Robot method), 22  
 save\_prefs() (opsoro.preferences.\_Preferences method), 15  
 say\_tts() (opsoro.sound.\_Sound method), 18  
 Server (class in opsoro.server), 16  
 Servo (class in opsoro.dof.servo), 6  
 Servo (class in opsoro.hardware.servo), 11  
 set() (opsoro.hardware.servo.Servo method), 12  
 set() (opsoro.preferences.\_Preferences method), 15  
 set\_all() (opsoro.hardware.neopixel.Neopixel method), 10  
 set\_all() (opsoro.hardware.servo.Servo method), 12  
 set\_all\_hsv() (opsoro.hardware.neopixel.Neopixel method), 10  
 set\_all\_us() (opsoro.hardware.servo.Servo method), 12  
 set\_brightness() (opsoro.hardware.neopixel.Neopixel method), 10  
 set\_config() (opsoro.expression.\_Expression method), 21  
 set\_config() (opsoro.robot.\_Robot method), 22  
 set\_control\_polygon() (opsoro.dof.DOF method), 6  
 set\_dof() (opsoro.module.Module method), 13  
 set\_dof() (opsoro.robot.\_Robot method), 22  
 set\_dof\_list() (opsoro.robot.\_Robot method), 22  
 set\_dof\_value() (opsoro.module.Module method), 13  
 set\_dof\_value() (opsoro.robot.\_Robot method), 22  
 set\_dof\_values() (opsoro.robot.\_Robot method), 22  
 set\_emotion\_e() (opsoro.expression.\_Expression method), 21  
 set\_emotion\_icon() (opsoro.expression.\_Expression method), 21  
 set\_emotion\_index() (opsoro.expression.\_Expression method), 21

set\_emotion\_name() (opsoro.expression.\_Expression method), 21  
 set\_emotion\_r\_phi() (opsoro.expression.\_Expression method), 21  
 set\_emotion\_random() (opsoro.expression.\_Expression method), 21  
 set\_emotion\_val\_ar() (opsoro.expression.\_Expression method), 22  
 set\_gpio\_pinmode() (opsoro.hardware.capacitive.Capacitive method), 8  
 set\_overlay\_value() (opsoro.dof.DOF method), 6  
 set\_pixel() (opsoro.hardware.neopixel.Neopixel method), 11  
 set\_pixel\_hsv() (opsoro.hardware.neopixel.Neopixel method), 11  
 set\_range() (opsoro.hardware.neopixel.Neopixel method), 11  
 set\_range\_hsv() (opsoro.hardware.neopixel.Neopixel method), 11  
 set\_threshold() (opsoro.hardware.capacitive.Capacitive method), 8  
 set\_urls() (opsoro.server.request\_handlers.RHandler method), 17  
 set\_value() (opsoro.dof.DOF method), 6  
 show() (opsoro.hardware.neopixel.Neopixel method), 11  
 show\_errormessage() (opsoro.server.request\_handlers.RHandler method), 17  
 shutdown() (opsoro.server.Server method), 16  
 sigterm\_handler() (in module opsoro), 23  
 sleep() (opsoro.robot.\_Robot method), 22  
 sleep() (opsoro.stoppable\_thread.StoppableThread method), 23  
 sound\_data() (opsoro.server.request\_handlers.RHandler method), 17  
 SpiDev (class in opsoro.hardware.dummy\_spidev), 9  
 start() (opsoro.robot.\_Robot method), 22  
 start\_alive\_loop() (opsoro.robot.\_Robot method), 22  
 start\_update\_loop() (opsoro.robot.\_Robot method), 22  
 stop() (opsoro.robot.\_Robot method), 23  
 stop() (opsoro.stoppable\_thread.StoppableThread method), 23  
 stop\_alive\_loop() (opsoro.robot.\_Robot method), 23  
 stop\_sound() (opsoro.sound.\_Sound method), 18  
 stop\_update\_loop() (opsoro.robot.\_Robot method), 23  
 StoppableThread (class in opsoro.stoppable\_thread), 23  
 stopped() (opsoro.stoppable\_thread.StoppableThread method), 23

## T

to\_us() (opsoro.dof.servo.Servo method), 7  
 Turn (class in opsoro.module.turn), 15

## U

update() (opsoro.dof.DOF method), 6  
 update() (opsoro.dof.servo.Servo method), 7  
 update() (opsoro.expression.\_Expression method), 22  
 update() (opsoro.module.Module method), 13  
 update() (opsoro.robot.\_Robot method), 23  
 usb\_serial (in module opsoro.hardware), 13

## W

wait\_for\_sound() (opsoro.sound.\_Sound method), 18  
 wake() (opsoro.robot.\_Robot method), 23  
 write16() (opsoro.hardware.i2c.I2C method), 10  
 write8() (opsoro.hardware.i2c.I2C method), 10  
 write\_gpio() (opsoro.hardware.capacitive.Capacitive method), 9

## X

xfer2() (opsoro.hardware.dummy\_spidev.SpiDev method), 9