OPSORO

Release 1.0

Aug 18, 2017

Contents:

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
"""
```

CHAPTER 3

OPSORO OS

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*, *maxn*)

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*, *maxn*)

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*, *maxn*)

`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 convertion 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*, *maxn*)

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

Python Module Index

Index