

---

# **Tux Eat Pi Documentation**

*Release 1*

**Tux Eat Pi Team**

**Sep 16, 2016**



<b>1 tuxeatpi</b>	<b>3</b>
1.1 tuxeatpi package . . . . .	3
<b>2 Indices and tables</b>	<b>11</b>
<b>Python Module Index</b>	<b>13</b>



Contents:



## 1.1 tuxeatpi package

### 1.1.1 Subpackages

#### tuxeatpi.actionner package

##### Submodules

#### tuxeatpi.actionner.actionner module

Voice component

**exception** `tuxeatpi.actionner.actionner.ActionError`

Bases: `Exception`

Base class for action exceptions

**class** `tuxeatpi.actionner.actionner.Actionner` (*tuxdroid*)

Bases: `multiprocessing.context.Process`

Define voices component

For now voice use Nuance communications services

**run** ()

Action to launch

**stop** ()

Stop process

#### tuxeatpi.actions package

##### Submodules

#### tuxeatpi.actions.tux module

Tux action

**class** `tuxeatpi.actions.tux.Action` (*tuxdroid*)

Bases: `object`

Base class for all NLU Actions

**exception** `tuxeatpi.actions.tux.ActionError`

Bases: `Exception`

Base class for Action exceptions

**class** `tuxeatpi.actions.tux.TuxAction` (*tuxdroid*)

Bases: `tuxeatpi.actions.tux.Action`

Class for NLU actions TuxDroid related

**get\_age** (*print\_it=False, text\_it=False, say\_it=False*)

Return tux age

**get\_birthday** (*print\_it=False, text\_it=False, say\_it=False*)

Return the tux birthday

**get\_name** (*print\_it=False, text\_it=False, say\_it=False*)

Return the tux name

**get\_time** (*print\_it=False, text\_it=False, say\_it=False*)

Return current time

**get\_uptime** (*print\_it=False, text\_it=False, say\_it=False*)

Return the tux uptime

**prefix** = 'tux'

**set\_lang** (*language, print\_it=False, text\_it=False, say\_it=False*)

Change Tux lang

## tuxeatpi.components package

### Submodules

#### tuxeatpi.components.base module

This module contains basic and useful classes to create Tux component like wings, eyes, ...

**class** `tuxeatpi.components.base.BaseComponent` (*pins, event\_queue, logger*)

Bases: `object`

Parent class use for component like wings, eyes, ...

Define some checks about component creation and switches function for handle input events

**pins** = `None`

**class** `tuxeatpi.components.base.Event` (*component, name, pin\_id*)

Bases: `object`

Event are created for each input event And store in Tux event queue

#### tuxeatpi.components.wings module

Wings component



**class** `tuxeatpi.components.wings.Wings` (*settings, event\_queue, logger*)

Bases: `tuxeatpi.components.base.BaseComponent`

Define wings component

Wings use 4 pins:

•**position:**

- INPUT
- Help to determine wings position ('up' or 'down')

•**left\_switch:**

- INPUT
- Event when use push the left wing

•**right\_switch**

- INPUT
- Event when use push the right wing

•**movement**

- OUTPUT
- Use to start/stop wings movement

**get\_position** ()

Return the current wings position and calibrate them if not available

**move\_count** (*count*)

Move wings N times

**move\_start** ()

Start moving wings

**move\_stop** ()

Stop moving wings

**move\_time** (*timeout*)

Move wings during until timeout

**move\_to\_position** (*position*)

Put wings to up position

**pins** = {'movement': None, 'left\_switch': None, 'position': None, 'right\_switch': None}

**exception** `tuxeatpi.components.wings.WingsError`

Bases: `Exception`

Base class for wings exceptions

## tuxeatpi.fake\_components package

### Submodules

#### tuxeatpi.fake\_components.base module

Base functions and classes for faking components

`tuxeatpi.fake_components.base.push_switch (pin_id)`  
Simulate switch pushing

### tuxeatpi.fake\_components.wings module

Fake Wings component

**class** `tuxeatpi.fake_components.wings.FakeWings (pins, event_queue, logger)`  
Bases: `tuxeatpi.components.wings.Wings`

Fake wings class

**move\_start** ()  
Override move\_start function for fake one

**move\_stop** ()  
Override move\_stop function for fake one

**push\_wing** (side)  
Simulation push switch function

**class** `tuxeatpi.fake_components.wings.FakeWingsMover (position_pin)`  
Bases: `threading.Thread`

Thread which simulate wings movement

**run** ()  
Start moving wings

**stop** ()  
Stop moving wings

### tuxeatpi.hotword package

#### Submodules

#### tuxeatpi.hotword.hotword module

#### tuxeatpi.libs package

#### Submodules

#### tuxeatpi.libs.lang module

Module handling i18n and l10n for tuxeatpi

`tuxeatpi.libs.lang.gtt (message)`  
Gettext wrapper

`tuxeatpi.libs.lang.load_languages ()`  
Prepare and load all supported languages  
TODO: make it more dynamic

`tuxeatpi.libs.lang.set_language (lang)`  
Change language on the fly

## tuxeatpi.libs.settings module

Settings module can read, check and write configuration file

**class** `tuxeatpi.libs.settings.Settings` (*config\_file, logger*)

Bases: `dict`

Class to handle settings: read/check/write

**reload** ()

Read Tux configuration from yaml file And set config values

**save** ()

Save settings on disk

**exception** `tuxeatpi.libs.settings.SettingsError`

Bases: `Exception`

Base class for configuration exceptions

## tuxeatpi.libs.websocket module

Module defining abstractWebSocket class

**class** `tuxeatpi.libs.websocket.AbstractWebSocketConnection` (*url, logger*)

Bases: `object`

WebSocket connection object to handle Nuance server communications

**MSG\_AUDIO = 2**

**MSG\_JSON = 1**

**close** ()

Close WebSocket connection

**connect** (*app\_id, app\_key, use\_plaintext=True*)

Connect to the websocket

**receive** ()

Handle server response

**send\_audio** (*audio*)

Send audio to the server

**send\_message** (*msg*)

Send json message to the server

## tuxeatpi.nlu package

### Submodules

#### tuxeatpi.nlu.common module

Utils for Nuance Mix Nlu services

**class** `tuxeatpi.nlu.common.NLUBase` (*settings, action\_queue, nlu\_queue, tts\_queue, logger*)

Bases: `multiprocessing.context.Process`

Define NLU base component

**stop ()**  
Stop NLU process

**class** `tuxeatpi.nlu.common.Recorder` (*device\_index=None, rate=None, channels=None, loop=None*)

Bases: `object`

Record voice from mic

**callback** (*in\_data, frame\_count, time\_info, status\_flags*)  
Callback function

**dequeue ()**

**enqueue** (*audio*)

**pick\_default\_device\_index ()**

**pick\_default\_parameters ()**

**class** `tuxeatpi.nlu.common.WebsocketConnection` (*url, logger*)

Bases: `tuxeatpi.libs.websocket.AbstractWebsocketConnection`

Websocket client

**connect** (*app\_id, app\_key, use\_plaintext=True*)  
Connect to the websocket

**static sign\_credentials** (*datestr, app\_key, app\_id*)  
Handle credentials

## tuxeatpi.nlu.nlu module

### tuxeatpi.voice package

#### Submodules

#### tuxeatpi.voice.common module

Utils functions for Nuance Communications TTS services

**class** `tuxeatpi.voice.common.WebsocketConnection` (*url, logger*)

Bases: `tuxeatpi.libs.websocket.AbstractWebsocketConnection`

WebSocket connection object to handle Nuance server communications

**connect** (*app\_id, app\_key, use\_plaintext=True*)  
Connect to the server

`tuxeatpi.voice.common.do_synthesis` (*url, app\_id, app\_key, language, voice, codec, input\_text, logger*)

The TTS function using Nuance Communications services

#### tuxeatpi.voice.voice module

Voice component

**class** `tuxeatpi.voice.voice.Voice` (*settings, tts\_queue, logger*)

Bases: `multiprocessing.context.Process`

Define voices component

For now voice use Nuance communications services

**is\_mute** ()

Return the mute state

**is\_speaking** ()

Check if the tux is currently speaking

**mute** ()

Mute the tux

**run** ()

Text to speech

**stop** ()

Stop process

**unmute** ()

Unmute the the

**exception** `tuxeatpi.voice.voice.VoicesError`

Bases: `Exception`

Base class for voice exceptions

## 1.1.2 Submodules

### `tuxeatpi.tux` module



---

## Indices and tables

---

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





## t

- tuxeatpi, 3
- tuxeatpi.actionner, 3
- tuxeatpi.actionner.actionner, 3
- tuxeatpi.actions, 3
- tuxeatpi.actions.tux, 3
- tuxeatpi.components, 4
- tuxeatpi.components.base, 4
- tuxeatpi.components.wings, 4
- tuxeatpi.fake\_components, 5
- tuxeatpi.fake\_components.base, 5
- tuxeatpi.fake\_components.wings, 6
- tuxeatpi.libs, 6
- tuxeatpi.libs.lang, 6
- tuxeatpi.libs.settings, 7
- tuxeatpi.libs.websocket, 7
- tuxeatpi.nlu, 7
- tuxeatpi.nlu.common, 7
- tuxeatpi.voice, 8
- tuxeatpi.voice.common, 8
- tuxeatpi.voice.voice, 8



**A**

AbstractWebsocketConnection (class in tuxeatpi.libs.websocket), 7  
 Action (class in tuxeatpi.actions.tux), 3  
 ActionError, 3, 4  
 Actionner (class in tuxeatpi.actionner.actionner), 3

**B**

BaseComponent (class in tuxeatpi.components.base), 4

**C**

callback() (tuxeatpi.nlu.common.Recorder method), 8  
 close() (tuxeatpi.libs.websocket.AbstractWebsocketConnection method), 7  
 connect() (tuxeatpi.libs.websocket.AbstractWebsocketConnection method), 7  
 connect() (tuxeatpi.nlu.common.WebsocketConnection method), 8  
 connect() (tuxeatpi.voice.common.WebsocketConnection method), 8

**D**

dequeue() (tuxeatpi.nlu.common.Recorder method), 8  
 do\_synthesis() (in module tuxeatpi.voice.common), 8

**E**

enqueue() (tuxeatpi.nlu.common.Recorder method), 8  
 Event (class in tuxeatpi.components.base), 4

**F**

FakeWings (class in tuxeatpi.fake\_components.wings), 6  
 FakeWingsMover (class in tuxeatpi.fake\_components.wings), 6

**G**

get\_age() (tuxeatpi.actions.tux.TuxAction method), 4  
 get\_birthday() (tuxeatpi.actions.tux.TuxAction method), 4  
 get\_name() (tuxeatpi.actions.tux.TuxAction method), 4

get\_position() (tuxeatpi.components.wings.Wings method), 5  
 get\_time() (tuxeatpi.actions.tux.TuxAction method), 4  
 get\_uptime() (tuxeatpi.actions.tux.TuxAction method), 4  
 gtt() (in module tuxeatpi.libs.lang), 6

**I**

is\_mute() (tuxeatpi.voice.voice.Voice method), 9  
 is\_speaking() (tuxeatpi.voice.voice.Voice method), 9

**L**

load\_languages() (in module tuxeatpi.libs.lang), 6

**M**

move\_count() (tuxeatpi.components.wings.Wings method), 5  
 move\_start() (tuxeatpi.components.wings.Wings method), 5  
 move\_start() (tuxeatpi.fake\_components.wings.FakeWings method), 6  
 move\_stop() (tuxeatpi.components.wings.Wings method), 5  
 move\_stop() (tuxeatpi.fake\_components.wings.FakeWings method), 6  
 move\_time() (tuxeatpi.components.wings.Wings method), 5  
 move\_to\_position() (tuxeatpi.components.wings.Wings method), 5  
 MSG\_AUDIO (tuxeatpi.libs.websocket.AbstractWebsocketConnection attribute), 7  
 MSG\_JSON (tuxeatpi.libs.websocket.AbstractWebsocketConnection attribute), 7  
 mute() (tuxeatpi.voice.voice.Voice method), 9

**N**

NLUBase (class in tuxeatpi.nlu.common), 7

**P**

pick\_default\_device\_index() (tuxeatpi.nlu.common.Recorder method), 8

`pick_default_parameters()` (tuxeatpi.nlu.common.Recorder method), 8  
`pins` (tuxeatpi.components.base.BaseComponent attribute), 4  
`pins` (tuxeatpi.components.wings.Wings attribute), 5  
`prefix` (tuxeatpi.actions.tux.TuxAction attribute), 4  
`push_switch()` (in module tuxeatpi.fake\_components.base), 5  
`push_wing()` (tuxeatpi.fake\_components.wings.FakeWings method), 6

## R

`receive()` (tuxeatpi.libs.websocket.AbstractWebSocketConnection method), 7  
`Recorder` (class in tuxeatpi.nlu.common), 8  
`reload()` (tuxeatpi.libs.settings.Settings method), 7  
`run()` (tuxeatpi.actionner.actionner.Actionner method), 3  
`run()` (tuxeatpi.fake\_components.wings.FakeWingsMover method), 6  
`run()` (tuxeatpi.voice.voice.Voice method), 9

## S

`save()` (tuxeatpi.libs.settings.Settings method), 7  
`send_audio()` (tuxeatpi.libs.websocket.AbstractWebSocketConnection method), 7  
`send_message()` (tuxeatpi.libs.websocket.AbstractWebSocketConnection method), 7  
`set_lang()` (tuxeatpi.actions.tux.TuxAction method), 4  
`set_language()` (in module tuxeatpi.libs.lang), 6  
`Settings` (class in tuxeatpi.libs.settings), 7  
`SettingsError`, 7  
`sign_credentials()` (tuxeatpi.nlu.common.WebsocketConnection static method), 8  
`stop()` (tuxeatpi.actionner.actionner.Actionner method), 3  
`stop()` (tuxeatpi.fake\_components.wings.FakeWingsMover method), 6  
`stop()` (tuxeatpi.nlu.common.NLUBase method), 7  
`stop()` (tuxeatpi.voice.voice.Voice method), 9

## T

`TuxAction` (class in tuxeatpi.actions.tux), 4  
`tuxeatpi` (module), 3  
`tuxeatpi.actionner` (module), 3  
`tuxeatpi.actionner.actionner` (module), 3  
`tuxeatpi.actions` (module), 3  
`tuxeatpi.actions.tux` (module), 3  
`tuxeatpi.components` (module), 4  
`tuxeatpi.components.base` (module), 4  
`tuxeatpi.components.wings` (module), 4  
`tuxeatpi.fake_components` (module), 5  
`tuxeatpi.fake_components.base` (module), 5  
`tuxeatpi.fake_components.wings` (module), 6  
`tuxeatpi.libs` (module), 6

`tuxeatpi.libs.lang` (module), 6  
`tuxeatpi.libs.settings` (module), 7  
`tuxeatpi.libs.websocket` (module), 7  
`tuxeatpi.nlu` (module), 7  
`tuxeatpi.nlu.common` (module), 7  
`tuxeatpi.voice` (module), 8  
`tuxeatpi.voice.common` (module), 8  
`tuxeatpi.voice.voice` (module), 8

## U

`unmute()` (tuxeatpi.voice.voice.Voice method), 9

## V

`Voice` (class in tuxeatpi.voice.voice), 8  
`VoicesError`, 9

## W

`WebsocketConnection` (class in tuxeatpi.nlu.common), 8  
`WebsocketConnection` (class in tuxeatpi.voice.common), 8  
`Wings` (class in tuxeatpi.components.wings), 4  
`WingsError`, 5