Tux Eat Pi Documentation

Release 1

Tux Eat Pi Team

Sep 16, 2016

Contents

Contents:

CHAPTER 1

tuxeatpi

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

Thie 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

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index