# PyArtNet

spacemanspiff2007

Mar 06, 2023

# CONTENTS:

CHAPTER

# ONE

# PYARTNET

pyartnet is a python implementation of the ArtNet protocol using asyncio. Supported protocols are ArtNet, sACN and KiNet.

## 1.1 Getting Started

```
import asyncio
from pyartnet import ArtNetNode

async def main():
    # Run this code in your async function
    node = ArtNetNode('IP', 6454)

    # Create universe 0
    universe = node.add_universe(0)

    # Add a channel to the universe which consists of 3 values
    # Default size of a value is 8Bit (0..255) so this would fill
    # the DMX values 1..3 of the universe
    channel = universe.add_channel(start=1, width=3)

    # Fade channel to 255,0,0 in 5s
    # The fade will automatically run in the background
    channel.add_fade([255,0,0], 1000)

    # this can be used to wait till the fade is complete
    await channel

asyncio.run(main())
```

## 1.2 Channels

### 1.2.1 Accessing channels

Created channels can be requested from the universe through the [] syntax or through *BaseUniverse.get_channel()*. If no channel name is specified during creation the default name will be built with {START}/{WIDTH}.

```
# create node/universe
    node = ArtNetNode('IP', 6454)
    universe = node.add_universe(0)

    # create the channel
    channel = universe.add_channel(start=1, width=3)

    # after creation this would also work (default name)
    channel = universe['1/3']
    channel = universe.get_channel('1/3')


    # it's possible to name the channel during creation
    universe.add_channel(start=4, width=3, channel_name='Dimmer1')

    # access is then by name
    channel = universe['Dimmer1']
    channel = universe.get_channel('Dimmer1')
```

### 1.2.2 Wider channels

Currently there is support for 8Bit, 16Bit, 24Bit and 32Bit channels. Channel properties can be set when creating the channel through *BaseUniverse.add_channel()*.

```
# create node/universe
    node = ArtNetNode('IP', 6454)
    universe = node.add_universe(0)

    # create a 16bit channel
    channel = universe.add_channel(start=1, width=3, byte_size=2)
```

## 1.3 Output correction

### 1.3.1 Output correction

It is possible to use an output correction to create different brightness curves. Output correction can be set on the channel, the universe or the node. The universe output correction overrides the node output correction and the channel output correction overwrites the universe output correction.

The graph shows different output values depending on the output correction.

From left to right: linear (default when nothing is set), quadratic, cubic then quadruple

Quadratic or cubic results in much smoother and more pleasant fades when using LED Strips.

### 1.3.2 Example

```
from pyartnet import ArtNetNode, output_correction

    # create node/universe/channel
    node = ArtNetNode('IP', 6454)
    universe = node.add_universe(0)
    channel = universe.add_channel(start=1, width=3)

    # set quadratic correction for the whole universe to quadratic
    universe.set_output_correction(output_correction.quadratic)

    # Explicitly set output for this channel to linear
    channel.set_output_correction(output_correction.linear)

    # Remove output correction for the channel.
    # The channel will now use the correction from the universe again
    channel.set_output_correction(None)
```

## 1.4 Class Reference

### 1.4.1 Universe and Channel

**class** pyartnet.**BaseUniverse**(*node*, *universe=0*)

**add_channel**(*start*, *width*, *channel_name=''*, *byte_size=1*, *byte_order='little'*)

Add a new channel to the universe. This will automatically resize the universe accordingly.

**Parameters**

- **start** (int) – start position in the universe
- **width** (int) – how many values the channel has
- **channel_name** (str) – name of the channel for requesting it from the universe
- **byte_size** (int) – byte size of a value
- **byte_order** (Literal['big', 'little']) – byte order of a value

**Return type**
*Channel*

**get_channel**(*channel_name*)

Return a channel by name or raise an exception

**Parameters**
**channel_name** (str) – name of the channel

**Return type**
*Channel*

set_output_correction(*func*)

Set the output correction function.

**Parameters**
**func** (Optional[Callable[[float, int], float]]) – None to disable output correction or the function which will be used to transform the values

**Return type**
None

class pyartnet.**Channel**(*universe*, *start*, *width*, *byte_size=1*, *byte_order='little'*)

get_values()

Get the current (uncorrected) channel values

**Return type**
List[int]

**Returns**
list of channel values

set_fade(*values*, *duration_ms*, *fade_class=<class 'pyartnet.fades.fade_linear.LinearFade'>*)

Add and schedule a new fade for the channel

**Parameters**

- **values** (Collection[Union[int, FadeBase]]) – Target values for the fade
- **duration_ms** (int) – Duration for the fade in ms
- **fade_class** (Type[FadeBase]) – What kind of fade

set_output_correction(*func*)

Set the output correction function.

**Parameters**
**func** (Optional[Callable[[float, int], float]]) – None to disable output correction or the function which will be used to transform the values

**Return type**
None

set_values(*values*)

Set values for a channel without a fade

**Parameters**
**values** (Collection[Union[int, float]]) – Iterable of values with the same size as the channel width

### 1.4.2 Node implementations

class pyartnet.**ArtNetNode**(*ip*, *port*, ***, *max_fps=25*, *refresh_every=2*, *start_refresh_task=True*, *source_address=None*, *sequence_counter=True*)

add_universe(*nr=0*)

Creates a new universe and adds it to the parent node

**Parameters**
**nr** (int) – universe nr

**Return type**
TypeVar(TYPE_U, bound= pyartnet.base.BaseUniverse)

**Returns**
The universe

`get_universe`(*nr*)

Get universe by number

**Parameters**
**nr** (int) – universe nr

**Return type**
TypeVar(TYPE_U, bound= pyartnet.base.BaseUniverse)

**Returns**
The universe

`set_output_correction`(*func*)

Set the output correction function.

**Parameters**
**func** (Optional[Callable[[float, int], float]]) – None to disable output correction or the function which will be used to transform the values

**Return type**
None

`start_refresh`()

Manually start the refresh task (if not already running)

`stop_refresh`()

Manually stop the refresh task

class pyartnet.**KiNetNode**(*ip*, *port*, *, *max_fps=25*, *refresh_every=2*, *start_refresh_task=True*, *source_address=None*)

`add_universe`(*nr=0*)

Creates a new universe and adds it to the parent node

**Parameters**
**nr** (int) – universe nr

**Return type**
TypeVar(TYPE_U, bound= pyartnet.base.BaseUniverse)

**Returns**
The universe

`get_universe`(*nr*)

Get universe by number

**Parameters**
**nr** (int) – universe nr

**Return type**
TypeVar(TYPE_U, bound= pyartnet.base.BaseUniverse)

**Returns**
The universe

set_output_correction(*func*)

Set the output correction function.

**Parameters**
**func** (Optional[Callable[[float, int], float]]) – None to disable output correction or the function which will be used to transform the values

**Return type**
None

start_refresh()

Manually start the refresh task (if not already running)

stop_refresh()

Manually stop the refresh task

class pyartnet.SacnNode(*ip*, *port*, *, *max_fps=25*, *refresh_every=2*, *start_refresh_task=True*, *source_address=None*, *cid=None*, *source_name=None*)

add_universe(*nr=0*)

Creates a new universe and adds it to the parent node

**Parameters**
**nr** (int) – universe nr

**Return type**
TypeVar(TYPE_U, bound= pyartnet.base.BaseUniverse)

**Returns**
The universe

get_universe(*nr*)

Get universe by number

**Parameters**
**nr** (int) – universe nr

**Return type**
TypeVar(TYPE_U, bound= pyartnet.base.BaseUniverse)

**Returns**
The universe

set_output_correction(*func*)

Set the output correction function.

**Parameters**
**func** (Optional[Callable[[float, int], float]]) – None to disable output correction or the function which will be used to transform the values

**Return type**
None

start_refresh()

Manually start the refresh task (if not already running)

stop_refresh()

Manually stop the refresh task

### 1.4.3 Fades

**class** pyartnet.fades.**LinearFade**

**debug_initialize**()

return debug string of the calculated values in initialize fade

**Return type**
str

### 1.4.4 Available output corrections

pyartnet.output_correction.**linear**(*val*, *max_val=255*)

linear output correction

**Return type**
float

pyartnet.output_correction.**quadratic**(*val*, *max_val=255*)

Quadratic output correction

**Return type**
float

pyartnet.output_correction.**cubic**(*val*, *max_val=255*)

Cubic output correction

**Return type**
float

pyartnet.output_correction.**quadruple**(*val*, *max_val=255*)

Quadruple output correction

**Return type**
float

CHAPTER

TWO

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

# INDEX