
CANberry Documentation

Release 0.4.post0.dev3+g5e56ad5

Andreas Wilhelm, Florian Wilhelm

October 28, 2015

1	Manuals	3
1.1	Installation	3
1.2	Configuration	3
1.3	Development	3
1.4	Credits	4
1.5	Note	4
1.6	Contents	4
1.7	Indices and tables	8
	Python Module Index	9

A small web app that displays sensor data and controls a [MOVIDRIVE Antriebsumrichter](#) (traction converter) of SEW EURODRIVE connected to the [Raspberry Pi](#) with the help of a controller area network (CAN) bus.

- MOVIDRIVE Serielle Kommunikation (10531602)
- MOVIDRIVE Handbuch (09191607)
- MOVIDRIVE Betriebsanleitung (10532609)
- MOVIDRIVE Operating Instruction (10532617)

1.1 Installation

In order to install CANberry just create a virtual environment and use pip:

```
pip install canberry
```

1.2 Configuration

Create a configuration file `.canrc` with following content in your home directory:

```
[default]
interface = socketcan
channel = can0

[canberry]
identifier = 16
# Is server externally visible? 'true' or 'false'
external = true
# Run the server in debug mode? 'true' or 'false'
debug = false
```

The identifier is the default target address.

Note: Running an externally visible server in debug mode is not recommended!

1.3 Development

Installation:

- Create a virtual environment *virtualenv venv* and activate it with *source venv/bin/activate*.
- Install all dependencies with *pip install -r requirements.txt*.
- Run *python setup.py develop* to install CANberry in your virtual environment.
- Run *canberry* from the command line to start the web application. Try *canberry -h* for help on more options.

Updating the javascript components with:

- Install *npm* with *sudo apt-get install nodejs*.
- Install *bower* with *sudo npm install -g bower*.
- Use *bower install -S* to install and update js dependencies like concise, ractive, jquery, flotcharts.

1.4 Credits

This project uses following libraries:

- jQuery
- Concise CSS
- Ractive.js
- Flot
- python-can

1.5 Note

This project has been set up using PyScaffold. For details and usage information on PyScaffold see <http://pyscaffold.readthedocs.org/>.

1.6 Contents

1.6.1 License

Copyright (c) 2015, Florian Wilhelm
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the <organization> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL <COPYRIGHT HOLDER> BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.6.2 Developers

- Andreas Wilhelm <Andreas_Wilhelm@gmx.de>
- Florian Wilhelm <Florian.Wilhelm@gmail.com>

1.6.3 Changelog

Version 0.4

- Some more configuration options added

Version 0.3

- Sending data is possible via the WebUI

Version 0.2

- Removed python-can contrib again
- Extended the documentation

Version 0.1.1

- Attempt to fix the python-can imports

Version 0.1

- First beta-like release

1.6.4 canberry

canberry package

Submodules

canberry.can_utils module

Implementation of the protocol for the MOVIDRIVE traction converter of SEW EURODRIVE

`class canberry.can_utils.Service`

Bases: `object`

Namespace for convenient and consistent naming

`NO_SERVICE = 'no_service'`

`READ_ATTR = 'attribute'`

`READ_DEFAULT = 'default'`

`READ_MAX = 'maximum'`

`READ_MIN = 'minimum'`

`READ_PARAM = 'parameter'`

`READ_SCALE = 'scale'`

`WRITE_PARAM = 'write_parameter'`

`WRITE_PARAM_VOLATILE = 'write_parameter_volatile'`

`code = {'minimum': 4, 'write_parameter_volatile': 3, 'scale': 7, 'default': 6, 'attribute': 8, 'write_parameter': 2, 'no_ser`

`canberry.can_utils.bytes_to_int(bytes)`

Convert a bytearray to an integer

Parameters `bytes` – bytearray

Returns integer

`canberry.can_utils.make_mgt_byte(service, sync=False)`

Creates the management byte according to the protocol

Parameters

- **service** – Service code as defined in `Service`
- **sync** – boolean if synchronized mode should be used

Returns integer

`canberry.can_utils.make_sdo(recipient, index, service=None, value=None, sync=False)`

Creates a Service Data Object message

Parameters

- **recipient** – the recipient as integer
- **index** – integer for the sensor
- **service** – requested service from `Service`
- **value** – None to read a value otherwise write value
- **sync** – Synchronized protocol

Returns Service Data Object message

canberry.cli module

The command line interface for canberry

`canberry.cli.main(args)`

```
canberry.cli.parse_args(args)
```

Parse command line parameters

Parameters *args* – command line parameters as list of strings

Returns command line parameters as dictionary

```
canberry.cli.run()
```

canberry.logic module

High-level functions to read and write a sensor of the MOVIDRIVE traction converter

```
class canberry.logic.Sensor
```

Bases: `object`

Namespace for convenient and consistent naming

DUMMY1 = 'dummy1'

DUMMY2 = 'dummy2'

SPEED = 'speed'

code = {'dummy2': 65535, 'dummy1': 65535, 'speed': 8318}

classmethod `list_all()`

```
canberry.logic.is_sensor_known(sensor)
```

Check if sensor is known

Parameters *sensor* – sensor as string

Returns boolean

```
canberry.logic.read_sensor(sensor)
```

Retrieve the data from a sensor

Parameters *sensor* – name of a sensor according to `inSensor`

Returns sensor data as dictionary

```
canberry.logic.write_sensor(sensor, value, volatile=False)
```

Write a value to a sensor

Parameters

- **sensor** – name of a sensor according to `inSensor`
- **value** – value to write
- **volatile** – write parameter `volatile` as boolean

canberry.utils module

Additional utilities

```
class canberry.utils.DummySensor(trans=0.0, scale=1.0)
```

Bases: `object`

A dummy sensor for test purposes

read()

set(*freq*)

`canberry.utils.add_timestamp(dct)`

Adds a timestamp attribute in milliseconds to a dictionary

Parameters `dct` – dictionary

`canberry.utils.list_attributes(obj)`

Lists all attributes of an object or class

Parameters `obj` – object or class

Returns dictionary of user-defined attributes

`canberry.utils.read_config()`

Read the configuration files `.canrc`, `can.conf` etc. as defined by python `can` in order to retrieve all settings from the section `[canberry]`. :return: dictionary

`canberry.utils.static_vars(**kwargs)`

Decorator for adding a static variable to a function

Parameters `kwargs` – initializations of the static variables

`canberry.utils.str2bool(txt)`

Convert a string to a boolean

Parameters `txt` – string object

Returns boolean

canberry.views module

Views of the flask application

`canberry.views.handle_dummy1()`

`canberry.views.handle_dummy2()`

`canberry.views.index()`

`canberry.views.list_sensors()`

`canberry.views.read_sensor(sensor)`

`canberry.views.write_sensor(sensor)`

Module contents

1.7 Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

C

- `canberry`, 8
- `canberry.can_utils`, 5
- `canberry.cli`, 6
- `canberry.logic`, 7
- `canberry.utils`, 7
- `canberry.views`, 8

A

add_timestamp() (in module canberry.utils), 8

B

bytes_to_int() (in module canberry.can_utils), 6

C

canberry (module), 8

canberry.can_utils (module), 5

canberry.cli (module), 6

canberry.logic (module), 7

canberry.utils (module), 7

canberry.views (module), 8

code (canberry.can_utils.Service attribute), 6

code (canberry.logic.Sensor attribute), 7

D

DUMMY1 (canberry.logic.Sensor attribute), 7

DUMMY2 (canberry.logic.Sensor attribute), 7

DummySensor (class in canberry.utils), 7

H

handle_dummy1() (in module canberry.views), 8

handle_dummy2() (in module canberry.views), 8

I

index() (in module canberry.views), 8

is_sensor_known() (in module canberry.logic), 7

L

list_all() (canberry.logic.Sensor class method), 7

list_attributes() (in module canberry.utils), 8

list_sensors() (in module canberry.views), 8

M

main() (in module canberry.cli), 6

make_mgt_byte() (in module canberry.can_utils), 6

make_sdo() (in module canberry.can_utils), 6

N

NO_SERVICE (canberry.can_utils.Service attribute), 6

P

parse_args() (in module canberry.cli), 6

R

read() (canberry.utils.DummySensor method), 7

READ_ATTR (canberry.can_utils.Service attribute), 6

read_config() (in module canberry.utils), 8

READ_DEFAULT (canberry.can_utils.Service attribute), 6

READ_MAX (canberry.can_utils.Service attribute), 6

READ_MIN (canberry.can_utils.Service attribute), 6

READ_PARAM (canberry.can_utils.Service attribute), 6

READ_SCALE (canberry.can_utils.Service attribute), 6

read_sensor() (in module canberry.logic), 7

read_sensor() (in module canberry.views), 8

run() (in module canberry.cli), 7

S

Sensor (class in canberry.logic), 7

Service (class in canberry.can_utils), 5

set() (canberry.utils.DummySensor method), 7

SPEED (canberry.logic.Sensor attribute), 7

static_vars() (in module canberry.utils), 8

str2bool() (in module canberry.utils), 8

W

WRITE_PARAM (canberry.can_utils.Service attribute), 6

WRITE_PARAM_VOLATILE (canberry.can_utils.Service attribute), 6

write_sensor() (in module canberry.logic), 7

write_sensor() (in module canberry.views), 8