Using the KNX-module pyknx for KNX-Integration

The pyknx module is made for python 2.7 and eases integration of KNX-Data in python processes. Once beeing imported the module provides some functions for KNX-Interaction.

1.0 Import module

```
import pyknx as knx
```

Program must be executed as root to have the necessary rights.

1.1 Available functions

1.1.1 Helper funtions

```
a=knx.grp2int("1/2/8")
# grp2int() converts a groupaddress-string into an integer

knx.readreq("1/2/9") or
knx.readreq(1200)
# readreq() sends a read-request to the bus.
# The address can be a group-address-string or an integer value
```

1.1.2 Reading:

```
a=knx.readint("1/0/0")
# Reads a signed integer, with length what ever the last telegram contained
# This can be anything between 1 bit and 32bit
# The address can be a group-address-string or an integer

a=knx.readint(knx.grp2int("1/0/0"))
a=knx.readint(2048)
# This does exactly the same

b=knx.readuint("1/0/0")
# Reads an unsigned integer, with length what ever the last telegram contained
# This can be anything between 1 bit and 32bit
# The address can be a group-address-string or an integer
```


Arcus-EDS GmbH
Rigaer Str. 88, 10247 Berlin
www.arcus-eds.de
sales@arcus-eds.de
Tel.: +49 / (0) 30 / 2593 3914
Fax: +49 / (0) 30 / 2593 3915

```
c=knx.readfloat("1/0/1")
# Reads a float, with length what ever the last telegram contained
# This can be anything between 8 bit and 32bit
# The address can be a group-address-string or an integer

d=knx.readstring("1/0/1")
# Reads a string, with length what ever the last telegram contained
# This can be anything between 1 and 14 byte
# The address can be a group-address-string or an integer
```

These functions return the last values present for this group address. If this group address is not existing an error exception is thrown.

```
Memory error: wrong data len
```

1.1.3 Writing:

```
knx.writeint("1/0/0",length,value)
# Writes length bytes of value to group 1/0/0
# Length of 0 means 1..6 Bit only
# The address can be a group-address-string or an integer

knx.writeuint("1/0/0",length,value)
# Writes length bytes of value (as unsigned )to group 1/0/0
# Length of 0 means 1..6 Bit only
# The address can be a group-address-string or an integer

knx.writefloat("1/0/0",length,value)
# Writes length ( 1,2,4 )bytes of float value to group 1/0/0
# The address can be a group-address-string or an integer

knx.writestring("1/0/0","Hello world")
# Writes string value ( up to 14 byte )to group 1/0/0
# The address can be a group-address-string or an integer
```


Arcus-EDS GmbH
Rigaer Str. 88, 10247 Berlin
www.arcus-eds.de
sales@arcus-eds.de
Tel.: +49 / (0) 30 / 2593 3914
Fax: +49 / (0) 30 / 2593 3915

1.1.4 Waiting for incoming telegrams:

```
knx.wait(1000)

or

def cb(addr):
    print(addr)

knx.wait(1000,cb)
```

the wait instruction waits for ms milliseconds and optionaly calls a callback function. The function returns the group-address of the incoming telegram. The callback is also executed with the actual group-address as argument.

1.1.5 Open and close:

```
import pyknx as knx

knx.open()
do_some_calculations
knx.close()
```

open() and close() prepare the internal shared memory to be available. This shortens the time for reaction on changing object data. Otherwise each read... and write.... function does an open() and close() internaly.

1.2 Examples:

```
import pyknx as knx
in1=knx.grp2int("1/0/0")
in2=knx.grp2int("1/0/1")
out1=knx.grp2int("1/0/2")
try:
    a=knx.readint(in1)
    b=knx.readint(in2)
    knx.writeint(out,1,a+b)
except:
    pass
```

This example must be executed with an on change event of "1/0/0" and/or "1/0/1" and outputs the sum to 1-Byte group address "1/0/2"

Arcus-EDS GmbH
Rigaer Str. 88, 10247 Berlin
www.arcus-eds.de
sales@arcus-eds.de
Tel.: +49 / (0) 30 / 2593 3914
Fax: +49 / (0) 30 / 2593 3915

An other example:

```
#import pyknx as knx
in1=knx.grp2int("1/0/0")
in2=knx.grp2int("1/0/1")
out1=knx.grp2int("1/0/2")
def callback(a):
    if a != in1 and a != in2 :
        try:
                a=knx.readint(in1)
                a=knx.readint(in2)
                knx.writeint(out1,1,a+b)
        except:
                pass

knx.open()

while(1):
    knx.wait(10000,callback)

knx.close()
```

This example must be run at startup in the background and keeps listening for input. It does the same as the previous example

Arcus-EDS GmbH
Rigaer Str. 88, 10247 Berlin
www.arcus-eds.de
sales@arcus-eds.de
Tel.: +49 / (0) 30 / 2593 3914
Fax: +49 / (0) 30 / 2593 3915