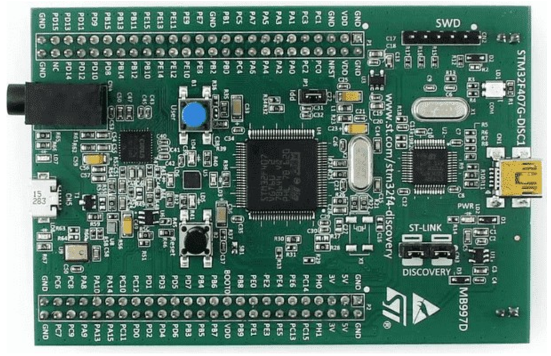
RODOS in Python

Python

Sebastian Kind 2023
sebastian.kind1@studmail.uni-wuerzburg.de
mail@sebastiankind.de



C++

TL;DR

```
import mwinterface as rodos
```

but first, you need to install the library

TL;DR

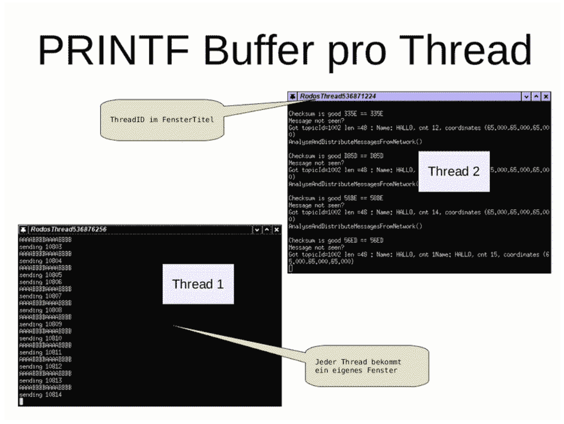
import mwinterface as rodos

- Topics, LinkInterfaces, Gateways, NetworkMessages
  -> **bidirectional communication between 2 Rodos systems**

- Nice feature: PRINTF(...) of every thread is shown in its own terminal window

- Quick python rundown with examples,

  Last slide: troubleshooting and some tips



```
RodosThread536871224

Checksum is good 335E == 335E
Message not seen?
Got topicId=1002 len =48 : Name: HALLO, cnt 12, coordinates (65.000,65.000,65.00
0)
AnalyseAndDistributeMessagesFromNetwork()

Checksum is good D85D == D85D
Message not seen?
Got topicId=1002 len =48 : Name: HALLO,                  5.000,65.000,65.00
0)
AnalyseAndDistributeMessagesFromNetwork(

Checksum is good 58BE == 58BE
Message not seen?
Got topicId=1002 len =48 : Name: HALLO, cnt 14, coordinates (65.000,65.000,65.00
0)
AnalyseAndDistributeMessagesFromNetwork()

Checksum is good 56ED == 56ED
Message not seen?
Got topicId=1002 len =48 : Name: HALLO, cnt 1Name: HALLO, cnt 15, coordinates (6
5.000,65.000,65.000)
```

```
RodosThread536876256
AAAABBBBAAAABBBB
sending 10003
AAAABBBBAAAABBBB
sending 10004
AAAABBBBAAAABBBB
sending 10005
sending 10006
AAAABBBBAAAABBBB
sending 10007
AAAABBBBAAAABBBB
sending 10008
AAAABBBBAAAABBBB
sending 10009
AAAABBBBAAAABBBB
sending 10010
AAAABBBBAAAABBBB
sending 10011
AAAABBBBAAAABBBB
sending 10012
AAAABBBBAAAABBBB
sending 10013
AAAABBBBAAAABBBB
sending 10014
```

install

- ~~pip install rodos and voila <- there is a version on PyPI~~
  - please use the python-middleware from the rodos repos

- today: (copy and paste)

```
git clone https://gitlab.com/rodos/rodos
cd rodos
git checkout python-mw-interface
cd support/support-programs/middleware-python/
./install.sh
```

```
then in Python: import mwinterface as rodos
```

create a topic

```
MyTopic1 = rodos.Topic(1234)
MyTopic2 = rodos.Topic(1001)

<Object> = rodos.Topic(<TopicNo>)

MyTopic1.publish(...)

Callback handler reacts to incoming
messages, you can put your code here
```

Linkinterface

```
LiUdp = rodos.linkinterfaceUDP()
```

```
UDP-Linkinterfaces
are set to broadcast
255.255.255.255

use case: let rodos communicate
 locally on your computer
```

```
LiUart = rodos.linkinterfaceUART("/dev/ttyUSB0")
```

/dev/tyyS0
/dev/rfcomm0 ← For BlueTooth

connect to a µController

Linkinterface+Gateway

```
li = rodos.LinkinterfaceUART("/dev/ttyUSB0")
gw = rodos.Gateway(li)
```

a gateway receives a linkinterface as its parameter

```
gw.forwardTopic(myTopic)
```

forward a topics
= topic will be shared over the gateway via
 the linkinterface

`gw.run()`

important the .run() method,
starts the gateway
in the background, this call returns
immediately

Sending data

You need to import the
struct library to work
on structured data

```
import struct

...

struct.pack("20sIddd", b"HALLO", cnt, 65, 65, 65)
myTopic1.publish(sendMe)
```

Receiving data

```
def topicHandler(data):
    unpacked = struct.unpack("LII", data)
    print("uint64:", unpacked[0], end=' ')
    print("uint32:", unpacked[1], end=' ')
    print("uint32:", unpacked[2], end=' ')


myTopic2.addSubscriber(topicHandler)
```

struct.unpack(...) parses the binary blob as speciefied in the format string (next slides)

this callback function will be called by the topic as soon as new data is received. You need to write your own callback functions and register them to the topic to parse your rodos-messages

Reminder: Do register your handlers with your topics as it is shown here

Pro Python Tip: make sure to handle exceptions in your callback handlers when parsing data!
a try/except a day keeps the doctor away

structs in C

- structs describe the memory layout of composed data types in C
- RODOS uses structs to describe the data type used in a topic
- please use __attribute__ ((packed))

structs in Python

- Python is dynamically typed and has no native concept of structs
- With the help of tools, it is possible to read/write the memory layout of structs

struct pack/unpack

- Python library to format structured types

  a = struct.pack("ccc", 65, 66, 67)

  a contains [65, 66, 67]

  a contains "ABC"

struct pack/unpack

It is possible to read and write all primitive C data types from C structs

```
struct point {
   int x;
   int y;
};
```

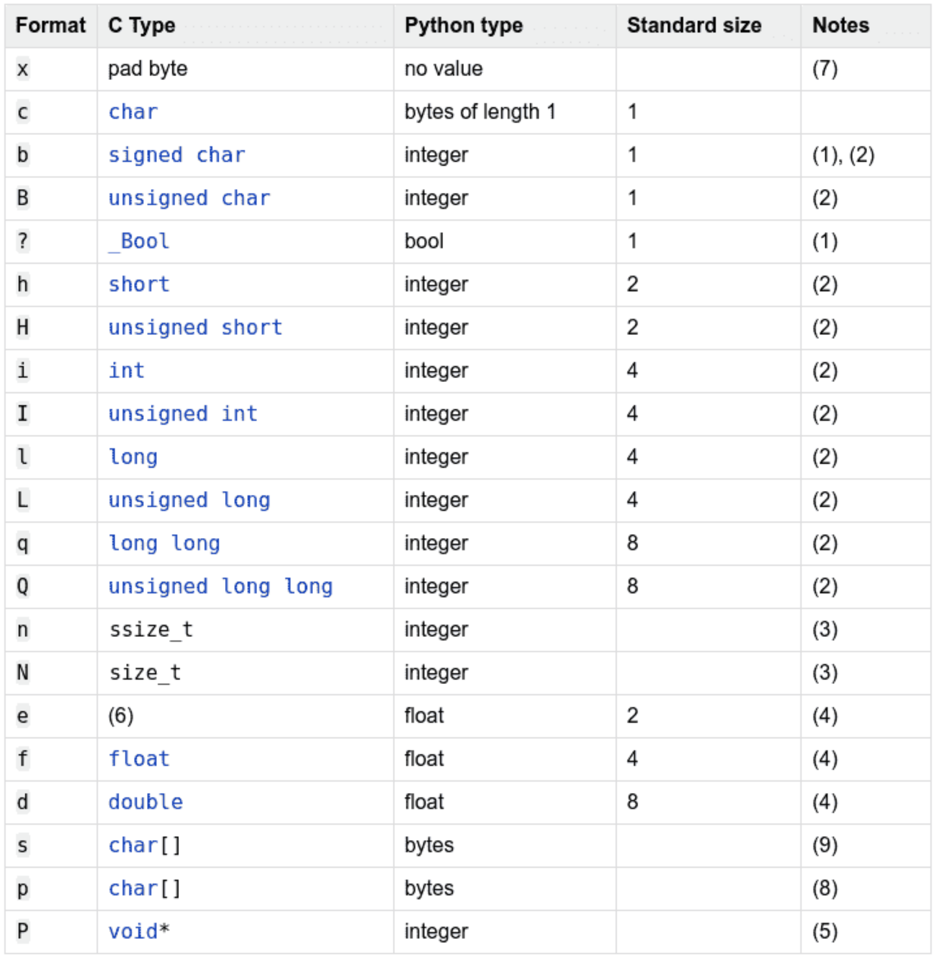
Read
```
values = struct.unpack("ii", point)
x = values[0]
y = values[1]
```

Write
```
point = struct.pack("ii", x, y)
```

python treats `point` as an bytearray `b"values"`

```
struct __attribute__ ((packed)) { short a; int b; }
```

Overview of supported C data types



Format	C Type	Python type	Standard size	Notes
x	pad byte	no value		(7)
c	char	bytes of length 1	1	
b	signed char	integer	1	(1), (2)
B	unsigned char	integer	1	(2)
?	_Bool	bool	1	(1)
h	short	integer	2	(2)
H	unsigned short	integer	2	(2)
i	int	integer	4	(2)
I	unsigned int	integer	4	(2)
l	long	integer	4	(2)
L	unsigned long	integer	4	(2)
q	long long	integer	8	(2)
Q	unsigned long long	integer	8	(2)
n	ssize_t	integer		(3)
N	size_t	integer		(3)
e	(6)	float	2	(4)
f	float	float	4	(4)
d	double	float	8	(4)
s	char[]	bytes		(9)
p	char[]	bytes		(8)
P	void*	integer		(5)

https://docs.python.org/3/library/struct.html <- Examples, Docs

more examples

```
cd /rodos/support/support-programs/middleware-python/rodos
```

- have a look at the tutorials folder
  - there are c++ rodos programs, together with a python program
- there are minimal barebones configurations in, that can be used as a starting point in `/rodos/support/support-programs/middleware-python/rodos/examples`

Setting up Bluetooth on Linux

this might take some seconds
you could also, use the graphical menu of your OS and look for something like "FloatSat"

```
#connect to discovery board

bluetoothctl scan on
bluetoothctl pair 00:0E:EA:CF:6C:54
sudo rfcomm bind 0 00:0E:EA:CF:6C:54

#minicom -D /dev/rfcomm0
```

You need to find the Bluetooth address of your adapter that is connected via UART to your µC

quick way to read view the data stream

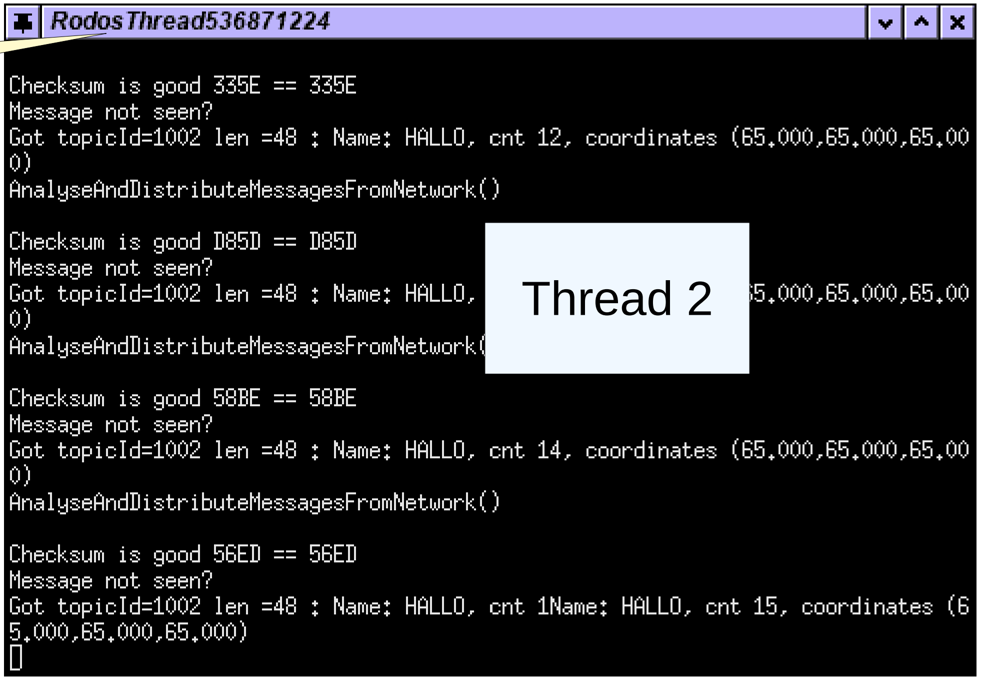
Subsequently, access to the connection can be made using Python with

`LinkinterfaceUART("/dev/rfcomm0")`

Sometimes you need to re-bind your adapter under a different number

PRINTF Buffer per Thread

ThreadID in windowtitle
(actually memory address
of thread)

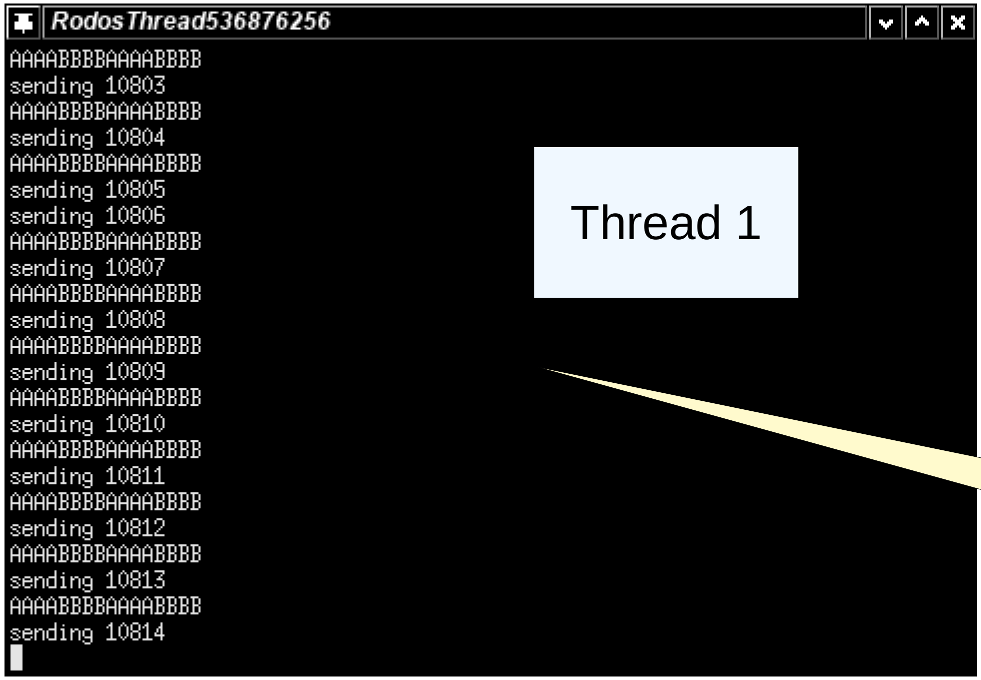


```
Checksum is good 335E == 335E
Message not seen?
Got topicId=1002 len =48 : Name: HALLO, cnt 12, coordinates (65,000,65,000,65,00
0)
AnalyseAndDistributeMessagesFromNetwork()

Checksum is good D85D == D85D
Message not seen?
Got topicId=1002 len =48 : Name: HALLO,                    5,000,65,000,65,00
0)
AnalyseAndDistributeMessagesFromNetwork(

Checksum is good 58BE == 58BE
Message not seen?
Got topicId=1002 len =48 : Name: HALLO, cnt 14, coordinates (65,000,65,000,65,00
0)
AnalyseAndDistributeMessagesFromNetwork()

Checksum is good 56ED == 56ED
Message not seen?
Got topicId=1002 len =48 : Name: HALLO, cnt 1Name: HALLO, cnt 15, coordinates (6
5,000,65,000,65,000)
```

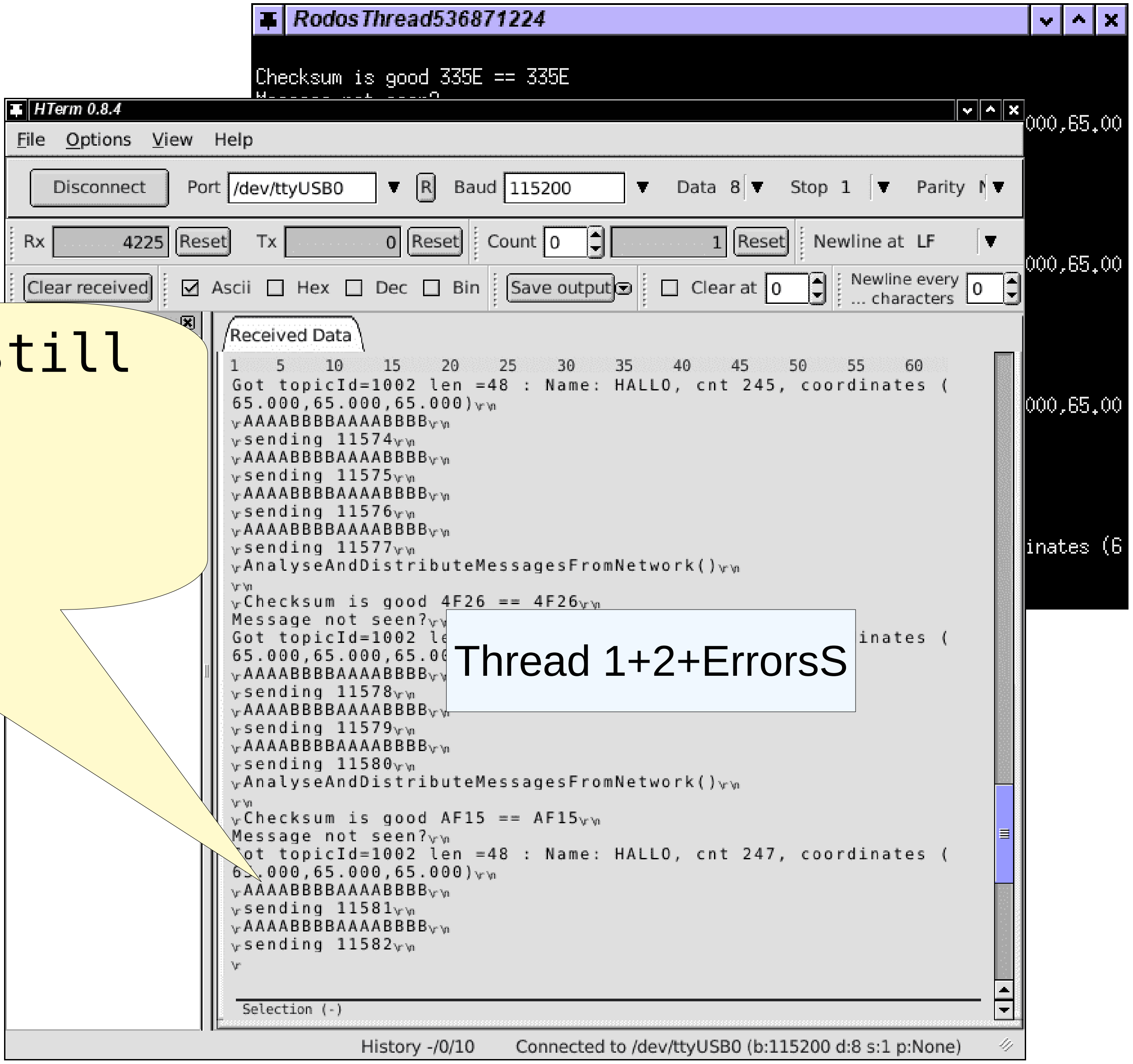


```
AAAABBBBAAAABBBB
sending 10803
AAAABBBBAAAABBBB
sending 10804
AAAABBBBAAAABBBB
sending 10805
sending 10806
AAAABBBBAAAABBBB
sending 10807
AAAABBBBAAAABBBB
sending 10808
AAAABBBBAAAABBBB
sending 10809
AAAABBBBAAAABBBB
sending 10810
AAAABBBBAAAABBBB
sending 10811
AAAABBBBAAAABBBB
sending 10812
AAAABBBBAAAABBBB
sending 10813
AAAABBBBAAAABBBB
sending 10814
```

Use **MW_PRINTF**("<3\n");

Each Thread gets its own
terminal, as soon as the printing
starts

PRINTF buffer per Thread



```
Checksum is good 335E == 335E
000,65,00
000,65,00
000,65,00
inates (6
```

Port /dev/ttyUSB0 · Baud 115200 · Data 8 · Stop 1 · Parity N
Rx 4225 · Tx 0 · Count 0 · 1 · Newline at LF
☑ Ascii ☐ Hex ☐ Dec ☐ Bin · ☐ Clear at 0 · Newline every ... characters 0

```
1    5     10    15    20    25    30    35    40    45    50    55    60
Got topicId=1002 len =48 : Name: HALLO, cnt 245, coordinates (
65.000,65.000,65.000)
AAAABBBBAAAABBBB
sending 11574
AAAABBBBAAAABBBB
sending 11575
AAAABBBBAAAABBBB
sending 11576
AAAABBBBAAAABBBB
sending 11577
AnalyseAndDistributeMessagesFromNetwork()

Checksum is good 4F26 == 4F26
Message not seen?
Got topicId=1002 le                                   inates (
65.000,65.000,65.00
AAAABBBBAAAABBBB
sending 11578
AAAABBBBAAAABBBB
sending 11579
AAAABBBBAAAABBBB
sending 11580
AnalyseAndDistributeMessagesFromNetwork()

Checksum is good AF15 == AF15
Message not seen?
Got topicId=1002 len =48 : Name: HALLO, cnt 247, coordinates (
65.000,65.000,65.000)
AAAABBBBAAAABBBB
sending 11581
AAAABBBBAAAABBBB
sending 11582
```

Selection (-)

History -/0/10   Connected to /dev/ttyUSB0 (b:115200 d:8 s:1 p:None)

Thread 1+2+ErrorsS

Don’t worry you’ll still have access to the serial terminal

trouble shooting, common errors

- Device Not Found/Ressource busy
  - UART/Bluetooth Device not plugged in, wrong name, device appears under different name look at /dev/ttyUSB0, /dev/tty/USB1, /dev/rfcomm0, etc
  - > give it time, wait solid 20 seconds, and restart your programs, if it doesnt help:
  - > re-bind device with bind 1, bind 2, have a look at the bluetooth page
- Everything hangs?
  - Deadlock, when single RODOS terminal gets closed, keep them running
  - > close all terminals, close python progam, reboot the µC, restart python
- struct parsing, values way too big, negative values, wrong size of struct??
  - change Little/Big Endian with ! at the beginning of format string, change unsigned int to int and likewise, take the byte order of your messages into account, recheck the byte offsets, perhaps values are overwritten, you can add pad bytes at any time
  - have a look at the C-types table, earlier in this document, listing all byte sizes
  - "pack expected a buffer of size blabla" structs happen to have different size on µC and PC sometimes, use __attribute__ ((packed))

    -> eg. struct __attribute__ ((packed)) { short a; int b; }