Eelectron Raspberry Pi KNX interface

This handbook is aimed to help you to:

- do your first steps on Raspberry Pi KNX interface

Index

Hardware specification

The KNX interface is connected to the Raspberry Pi board using the expansion header marked P1.
The used pins are:

GPIO 14 (TXD)
GPIO 15 (RXD)
3V3 Power
Ground

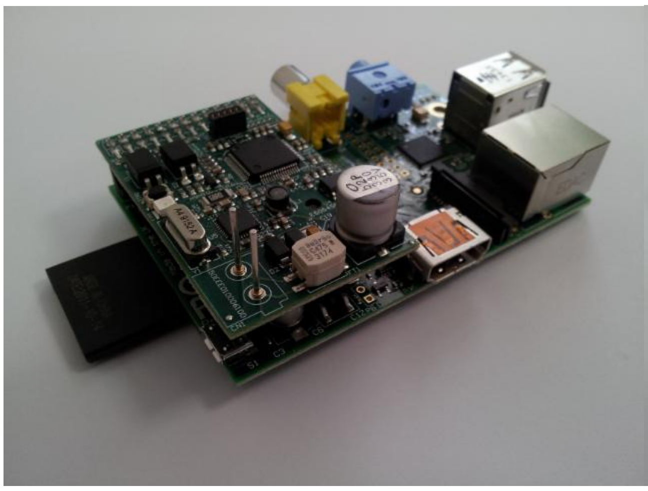
The interface is opto-isolated from the Raspberry Pi board.

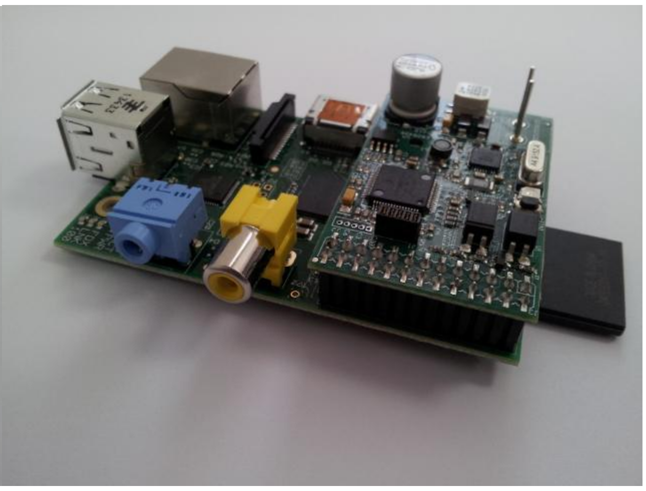
Serial port settings:

19200 baud
8 bits
parity none
1 stop bit

How to install the interface

Just plug the 13x2 pins receptacle to the Raspberry Pi board header marked as P1.





Operating system configuration

If you are using the Raspbian Operating System (Debian Wheezy), by default the serial port is configured to provide boot-up information.
To enable the serial port to communicate with the IC00R01KNX interface you have to free it.
Two files need to be edited, the first one to disable serial port login

/etc/inittab

edit the file, you'll see a line similar to

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Disable it by adding a # character to the beginning and save the file.

```
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

the second one is to disable the bootup info

/boot/cmdline.txt

The contents of the file look like this

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1
root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

Remove all references to ttyAMA0 (which is the name of the serial port). The file will now look like this

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline rootwait
```

for changes to take effects, you need to reboot the Raspberry Pi

Record specifications

Record are identified by a start transmission character <STX> [hex02] and an end transmission character <ETX> [hex03].
Bytes values are reported as two ASCII characters notation [hex2E]
First byte identify the control field
Bytes from 2 to n are the payload
Byte n+1 is the checksum, calculated as xor of bytes from 2 to n

0x02	Control field	Telegram info	Data	Data	Data	Data	Data…	Checksum	0x03
	make xor to calc the checksum								

Control fields:

KNX to Raspberry

0x41	Group monitor Value response	
0x42	Group monitor Value write	
0x4F	ACK or NAK as reply to value read or write	
0x48	KNX BCU status	
0x49	KNX BCU version	
0x45	KNX BCU individual address	

Raspberry to KNX

0x81	Value read	
0x82	Value write	
0x87	read KNX BCU version	
0x85	read individual address KNX BCU	
0x86	Set individual address KNX BCU	

Telegram info:

Bit description

reserved	reserved	RC2	RC1	RC0	P1	P0	bitless

bitless:
set to 1 for telegrams where payload length < 1 byte

telegram priority:

Description	P1	P0
Low	1	1
High	0	1
Alarm	1	0

routing counter:

currently not implemented for transmitting telegrams

KNX BCU to Raspberry

Value Response

Function: every time the KNX BCU listens to a read response telegram on KNX bus, forward it to the Raspberry interface

Field			
0x02	STX		
41	Control field		
??	Telegram info		
??	Source address Hi byte		
??	Source address Lo byte		
??	Target address Hi byte		
??	Target address Lo byte		
??	First payload data		
??			
??	n payload data		
??	Checksum		
0x03	ETX		

Es.
Send, from KNX BCU to Raspberry:
<STX>41xx87460D2001AC<ETX>

Value Write

Function: every time the KNX BCU listens to a write telegram on KNX bus, forward it to the Raspberry interface

Field			
0x02	STX		
42	Control field		
??	Telegram info		
??	Source address Hi byte		
??	Source address Lo byte		
??	Target address Hi byte		
??	Target address Lo byte		
??	First payload data		
??			
??	n payload data		
??	Checksum		
0x03	ETX		

Es.
Send, from KNX BCU to Raspberry:
<STX>42xx87460D2001AF<ETX>

ACK or NAK reply

Function: every time the KNX BCU receive a write, read or set individual address command, reply with a ACK or NACK telegram

Field			
0x02	STX		
4F	Control field		
??	Received command repeated 81, 88, 89, 86		
??	Reply ACK 60 or NAK 51		
??	Checksum		
0x03	ETX		

Es.
ACK sent from KNX BCU to Raspberry:
<STX>4F8860xx<ETX>

NACK sent from KNX BCU to Raspberry:
<STX>4F8851xx<ETX>

KNX BCU version

Function: return the hardware and firmware version of KNX BCU

Field			
0x02	STX		
49	Control field		
??	Hardware version Hi byte		
??	Hardware version Lo byte		
??	Firmware version Hi byte		
??	Firmware version Lo byte		
??	Checksum		
0x03	ETX		

Es.
sent from KNX BCU to Raspberry:
<STX>490100010049<ETX>

KNX BCU read individual address

Function: return the individual address of KNX BCU

Field			
0x02	STX		
46	Control field		
??	Individual address Hi byte		
??	Individual address Lo byte		
??	Checksum		
0x03	ETX		

Es.
sent from KNX BCU to Raspberry:
<STX>8585<ETX>

Raspberry to KNX BCU

Value read

Function: make a read request to KNX bus. An ack or nack telegram must be received from the KNX BCU

Field			
0x02	STX		
81	Control field		
??	Telegram info		
??	Target address Hi byte		
??	Target address Lo byte		
??	Checksum		
0x03	ETX		

Es.
sent from KNX BCU to Raspberry:
<STX>490100010049<ETX>

Value write

Function: make a write to KNX bus. A ack or nack telegram must be received from the KNX BCU

Field			
0x02	STX		
82	Control field		
??	Telegram info		
??	Target address Hi byte		
??	Target address Lo byte		
??	First payload data		
??			
??	n payload data		
??	Checksum		
0x03	ETX		

Es.
sent from KNX BCU to Raspberry:
<STX>490100010049<ETX>

KNX BCU version request

Function: ask for the KNX BCU hardware and firmware version

Field			
0x02	STX		
87	Control field		
87	Checksum		
0x03	ETX		

Es.
sent from KNX BCU to Raspberry:
<STX>8787<ETX>

KNX BCU set individual address

Function: set the individual address of the KNX BCU

Field			
0x02	STX		
86	Control field		
??	Individual address Hi byte		
??	Individual address Lo byte		
??	Checksum		
0x03	ETX		

Es.
sent from KNX BCU to Raspberry:
<STX>86??<ETX>

KNX BCU read individual address

Function: set the individual address of the KNX BCU

Field			
0x02	STX		
85	Control field		
85	Checksum		
0x03	ETX		

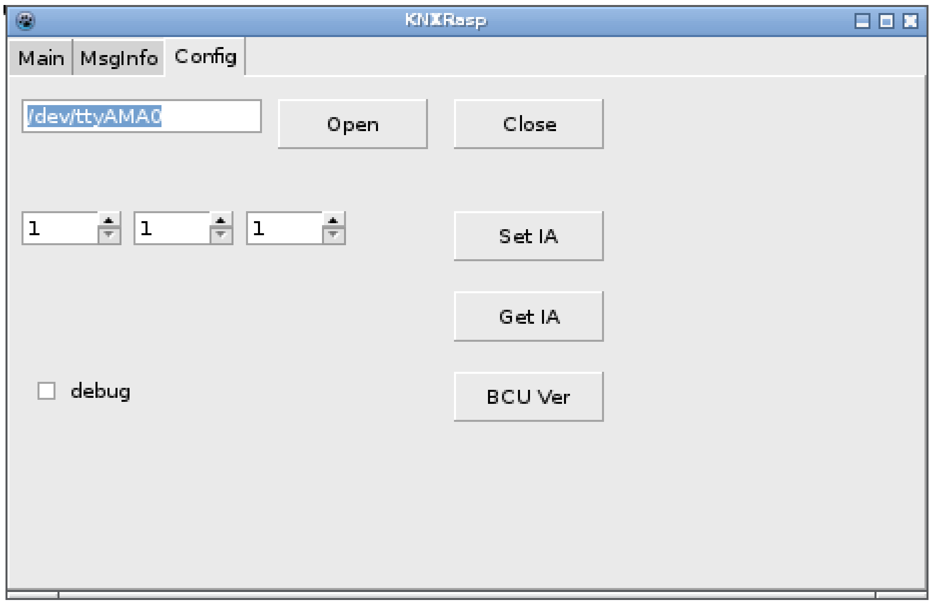
Es.
sent from KNX BCU to Raspberry:
<STX>86??<ETX>

Software example

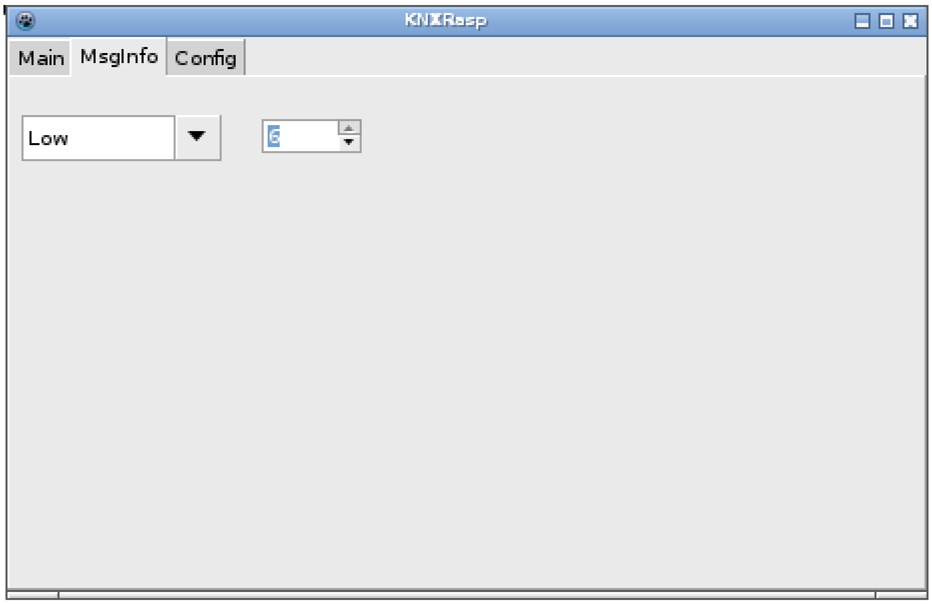
knxRasp is a small software designed to test the functions of the IC00R01KNX interface

Commands present in the **Config** page can be used to

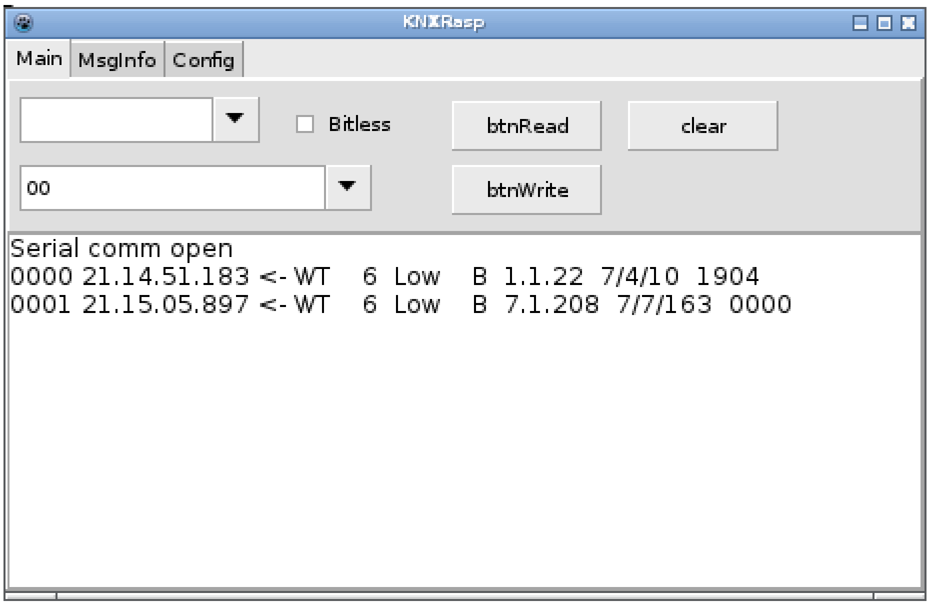
- Open/Close the communication
- Set interface individual address
- Get current individual address
- Get current hardware and firmware version



With **MsgInfo** page you can define message priority and routinc counter value



Main page is divided in two sections, the lower one is a group monitor reporting telegram details, the upper one is used to send write or read request telegrams on the bus



```
Serial comm open
0000 21.14.51.183 <- WT   6  Low   B  1.1.22  7/4/10  1904
0001 21.15.05.897 <- WT   6  Low   B  7.1.208  7/7/163  0000
```

Revision History

Date	Document ref	Comments
28/01/2014	IC00R01KNXSW01010001	First release
29/04/2014	IC00R01KNXSW01020001	Added serial port settings