SEAHU SH017 (PiToDin)
Seriová linka

HARDWEROVÝ POPIS

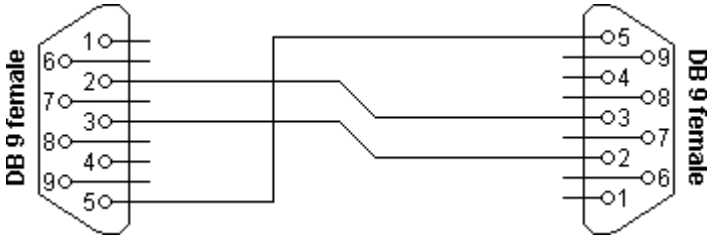
Původní sériová linka z mini počítače raspberryPI pracující na 3,3V logice je v modulu Seahu SH017 rozšířená o převodník logických úrovní RS232 a RS481. Oba tyto převodníky sdílejí stejnou sériovou linku, tj. můžete používat buď RS232 nebo RS481 ne obojí najednou.
Nejedná se o plnohodnotnou sériovou linku se všemi řídicími signály, ale pouze o základní linku tvořenou signály TX,RX,GND.
Převodník RS232 je obousměrný, tj. samostatná linka pro příjem a samostatná linka pro vysílaní dat. Kdežto převodník RS481 je jednosměrný tj. jedna linka se dělí o příjem i vysílaní dat. O směru doku dat rozhoduje pin 11 (GPIO17/GEN0) z mini PC raspberryPI dle následujícítabulky.

Vliv stavu pinu 11 na RS481

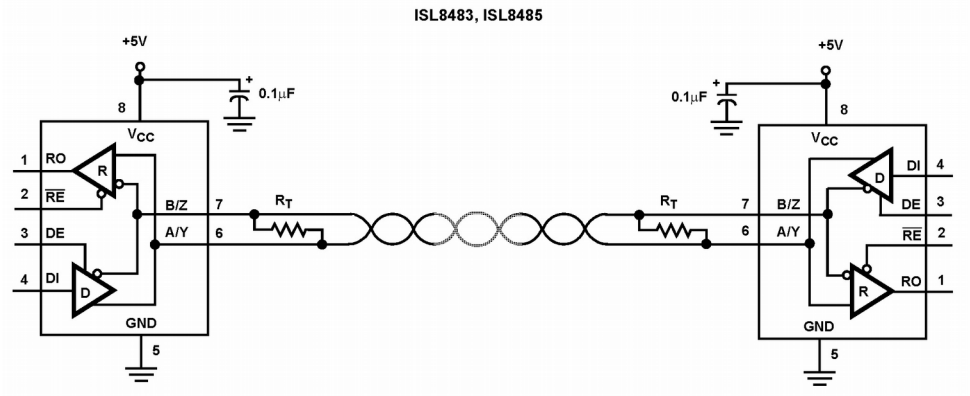
Stav	význam
1	TX-vysílaní dat
0	RX-příjem dat

Signál RX z převodníku RS232 i RX signál z převodníku RS481 je sloučen pomocí funkce AND. Klidový stav obou linek odpovídá logické 1, proto je možné oba převodníky sdílet. Pro používaní převodníku RS232 je vhodné pin11 nastavit na log. 1 (případně ho ponechat jako vstup s vysokou impedanci), tím se převodník RS481 přepne do vysíláni a odpojí svojí linku RX, takže pak nemůže ovlivňovat RX linku z RS232.

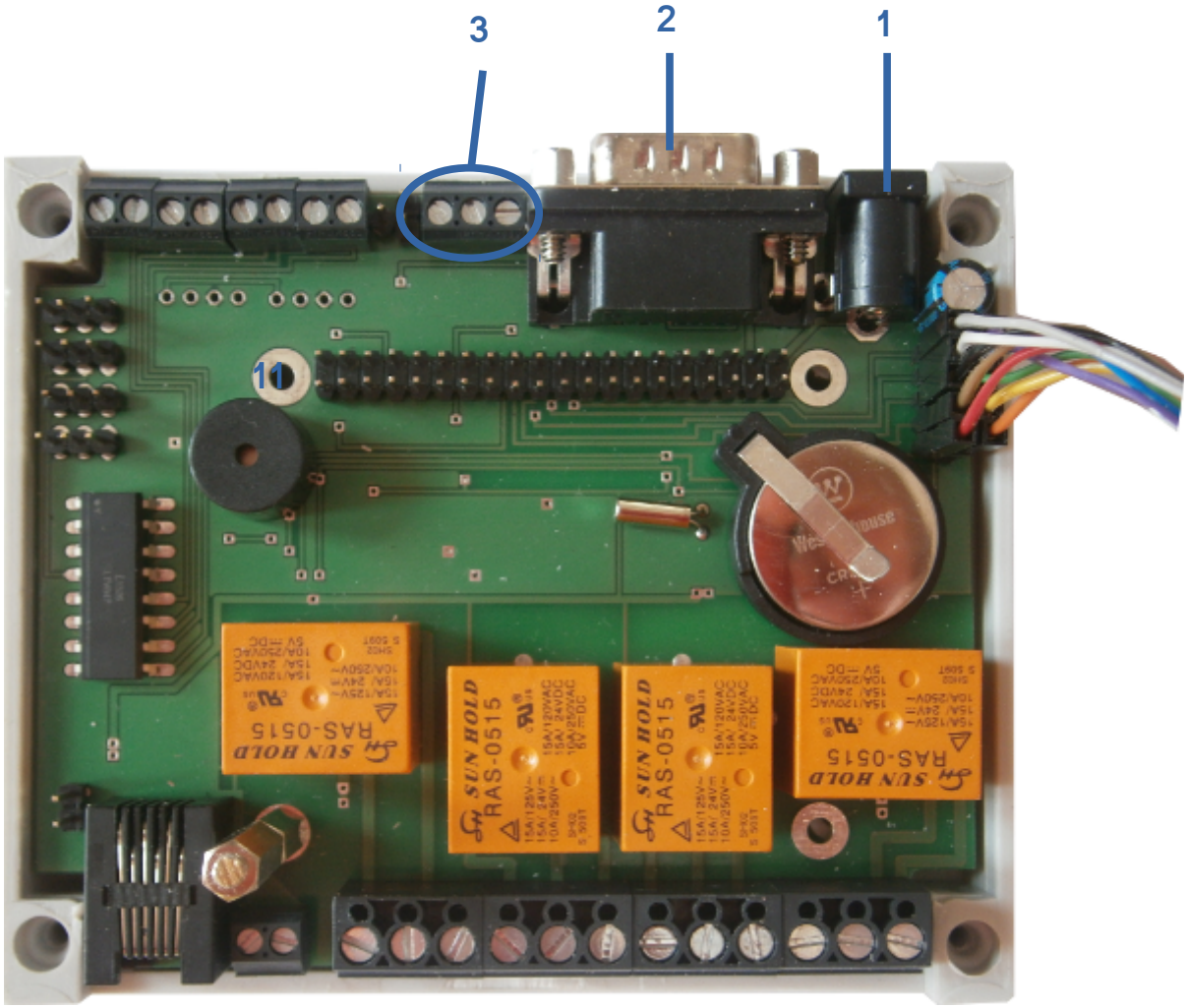
RS232 používá se převážně na propojení dvou zařízení v délce kabelu cca do 5m. Při propojení dvou modulu SH017 nebo propojením tohoto modulu s počítačem je potřeba použít kříženého kabelu (tj. signál TX s jedné strany musí být zapojen do RX na straně druhé a obráceně.). Nejčastěji se pro RS232 používá 9-ti pi-nový CAN konektor.



RS485(481) umožňuje při požití kroucené dvojlinky údajně překlenou vzdálenost až 1,2 km. Rozšířená je hlavně průmyslových zařízeních. Nepoužívá se jen pro spojení dvou bodu, ale také jako sběrnice na které může být najednou až 32 zařízení přičemž jedno s těchto zařízení řídí provoz tzv. Master. V průmyslovém odvětví je tato sběrnice spjatá s protokolem MODBUS, pro který existuje spousta zařízení (čidel, měřicích hodin,..).Tato sběrnice nemá žádný standardizovaný konektor, nejčastěji se používá svorkovnice s označením vývodu A, B, GND. Přičemž GND není pro komunikaci důležitý. Spojují se vždy svorkovnice A s A a B s B. Oba konce vedení je pak zapotřebí opatřit odporem 120Ω tzv. terminátorem, pro tento účel je modul SEAHU SH017 opatřen propojkou jejímž propojením se terminátor zapojí.



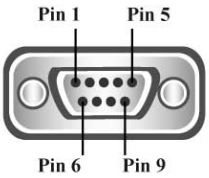
ZÁKLADNÍ DESKA:



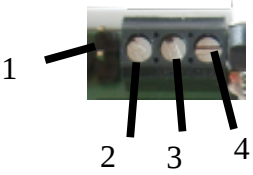
DESCRIPTION:

1. C 2,5mm konektor 5V 2A.

2. Sériový konektor (RS232).
   Pin 2 – RX
   Pin 3 – TX
   Pin 5 – GND



3. RS485
   1 – propojka terminátoru s odporem 120Ω
   2 – B pin
   3 – A pin
   4 – GND



RaspberryPI mini computer

RS232 v Linuxu (přesněji Rasbianu tj. linuxové distribuci speciálně pro mini PC raspberry PI)

Sériový port je v Linuxu zařízení, které se ovládá pomoci souboru /dev/ttyXX. Název tohoto souboru se bohužel mění podle typu portu v každém zařízení trochu jiný. Tak např. U

raspberry PI do v.2	/dev/ttyAMA0
raspberry PI v.3	/dev/ttyS0 (/dev/ttyAMA0 je použit pro bluetooth)
a pokud připojíte USB→serial převodnik	/dev/ttyUSB0, /dev/ttyUSB1,...

Co do tohoto souboru zapíšete se přepošle na sériovou linku, a když s tohoto souboru budete číst tak dostanete to co tam někdo poslal. Avšak sériová komunikace není zas tak úplně jednoduchá. Pomoci IOCT funkci nad tímto souborem, lze nastavit rychlost přenosu, paritu, start a stop bity. Tyto záležitosti je v Linuxu lepší svěřit speciálním programům nebo knihovním funkcím v programovacích jazycích (tedy nejnáročnější je počáteční nastaveni).
V modulu SEAHU SH017 kde je převodník RS232 sdílen s převodníkem RS481 je dobře si pohlídat pin11 z raspberry PI, aby nebyl nastaven na logickou 0 (může být nastaven jako výstup s log. 1 nebo ponechán ve výchozím stavu jako vstup). Tato sériová linka nemá žádné hardwarové řízení komunikace. Takže pokud budete na linku rychle posílat hodně dat, může se stát že dotyčnému zařízení se přeplní buffer a něco s komunikace se ztratí. Totéž se může stát i obráceně, jestliže nebudete dostatečně často zkoušet vyčítat co na sériovou linku dostáváte, opět může dojít k zaplnění tentokrát vašeho bufferu a část komunikace se může ztratit.

Možné použití sériové linky v Linuxu

- spojení s PC
  - lze přenášet terminál (linuxový příkazový řádek) a to jak s raspberry pi to PC nebo i obráceně.
  - Síťové propojení pomocí protokolu PPP (point-to-point)
- propojení s mikrokontroléry nebo jinými externími zařízeními
  - Přímé propojení periférii. I když je sériový port na ústupu stále se najde ve spoustě zařízeni hlavně těch jednodušších a dá se říci ,že není důvod proč by měl vymizet. např.
    - malé pokladní tiskárny
    - zobrazovací panely
    - váhy
    - modemy
    - alarmy
  - Přemostění komunikace po sériové lince přes poč. síť po jiného PC. Informace z periferií připojených k sériové lince se nutně nemusí zpracovávat v tomto počítači. I pro tento účel lze modul SEAHU SH017 použít (i když pro tak jednoduchou úlohu je to trochu škoda).

Ověření funkčnosti sériové komunikace mezi SEHU SH017 a PC s Linuxem.

Propojte PC a modul pomocí kříženého kabelu. Pokud PC nemáte sériový port můžete použít převodník USB→seriál k dostaní je jich spousta většinou jsou i přímo podporované v Linuxu.
Jak v modulu tak na PC spusťte minicom a pokud něco napíšete na jednom počítači tak by se to mělo objevit na druhém a naopak.
Instalace minicomu:

```
Shell:~$ sudo apt-get install minicom
```

Spuštění minicomu (na PC je potřeba vybrat správný port /dev/ttyXX):

```
Shell:~$ minicom -b 115200 -o -D /dev/ttyS0
```

Místo minicomu lze použít jednodušší screen. (115200 je rychlost přenosu dat)

```
Shell:~$ screen /dev/ttyS0 115200
```

Pokud toto nefunguje zkontrolujte ještě stav pinu 11 z raspberry PI viz výše. V případě dlouhého kabelu můžete zkusit snížit přenosovou rychlost, nejnižší používaná je 9600. Běžně používané přenosové rychlosti: 115200, 57600, 38400, 19200, 9600 bps.

Nastavení sériového portu pro výchozí terminál (linuxová příkazová řádka).
Toto se používá většinou u zařízení, které nemají display. U modulu SEAHU SH017 je tato volba standardně vypnutá, protože pokud je na sériovou linku přesměrován výchozí terminál tak je linka zblokovaná pro používaní jiným programem. Avšak pokud neplánujte připojovat k modulu sériová zařízení a pokud nepoužíváte poč. síť a nemate po ruce HDMI monitor a USB klávesnici, ale jen notebook se sériovým portem, tak toto může být jedná z mála možnosti jak modul ovládat.
Spusťte „sudo raspi-config“ a ve volbě „advanced options -> serial“ pokud není povolte přihlašování pomocí sériové linky a změny potvrďte. Následně restartujte modul.
Na notebooku spusťte minicom nebo screen s rychlosti 115200, následně byste měli mít možnost se přihlásit do linuxové příkazové řádky tzv. terminál.

Pokud tuto možnost již nadále nechcete tak přes „sudo raspi-config“ výchozí sériový terminál opět vypněte a v */boot/config.txt* povolte čistě jen sériový port volbou „`enable_uart=1`“

Síťové spojení pomocí PPP (point-to-point) protokolu
Pomocí síťového spojení lze pak přenášet soubory, připojovat síťové disky, spustit terminál (ssh, nebo telnet), atd.
Pro toto spojení je potřeba mít volnou sériovou linku na obou stranách spojení a pote již stačí na obou stranách spustit ppp démona (službu), který se postará o vytvoření virtuální síťovky, přes kterou jsou data tunelovaná sériovou linkou.

Pokud nemáte nainstalovaný program pppd, nainstalujte ho následujícím příkazem:

```
Shell:~$ sudo apt-get install ppp
```

V jednom PC spusťte:

```
Shell:~$ sudo pppd noauth /dev/ttyS0 115200
10.0.0.1:10.0.0.2 passive local maxfail 0 nocrtscts
xonxoff
```

PS: (vše napište na jeden řádek) volba passive znamená, že se program hned neukončí, ale čeká na platný paket od protistrany.
IP adresy jsou na vás, první je lokální a za dvojtečkou je IP protistrany.

Na druhem PC pak spusťte:

```
Shell:~$ sudo pppd noauth /dev/ttyS0 115200
10.0.0.2:10.0.0.1 local maxfail 0 nocrtscts xonxoff
```

v tom okamžiku se vytvoří virtuální síťovky což si můžete ověřit zadáním příkazu:
ifconfig
v jehož výstupu byste měli najít:

```
Shell:~$ ifconfig
...
ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.0.0.2  P-t-P:10.0.0.1  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:314 (314.0 B)  TX bytes:314 (314.0 B)
...
```

také si může vyzkoušet ping, na obě nové adresy by měl projít
ping 10.0.0.1
ping 10.0.0.2

```
Shell:~$ ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.399 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.385 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.375 ms
```

pppd má mnohem více voleb, více naleznete na:
http://elinux.org/RPi_Serial_Connection#Linux_terminal_set_up
http://www.tldp.org/HOWTO/PPP-HOWTO/

Přemostění sériového portu přes síť
Existuje cela řada programu a možnosti jak tohoto docílit. Nejjednodušší mně připadá kombinace programu ser2net na straně serveru (modulu) a programu socat na straně klienta tj. PC kde se port tuneluje.
Instalace programu ser2net a socat:

```
Shell:~$ sudo apt-get install ser2net
Shell:~$ sudo apt-get install socat
```

Server - program ser2net:
Je ho potřeba nejprve nakonfigurovat. Nutno upozornit, že fyzické parametry sériové linky jako rychlost ,parita, start bity se nastavuji dopředu v konfiguraci, na virtuálním portu v cílovém počítači si můžete nastavovat co chcete na nic to už nebude mít vliv. Konfiguraci najdete v */etc/ser2net.conf* stačí do něj umístit následující řádek:

```
3333:raw:0:/dev/ttyS0:115200,8DATABITS,NONE,1STOPBIT
```

Význam je poměrně jasný:
3333 je číslo tcp portu můžete si nastavit svůj
/dev/ttyS0 je sériový port
115200 je rychlost
ostatní volby zde nebudu popisovat, jsou totiž dostatečně popsány přímo v konfiguračním souboru.

Dobré je že jeden a ten samý sériový port může být v konfiguraci uveden víckrát, takže např. na různých síťových portech může být ten samý sériový port pokaždé např. s jinou rychlosti.

Po změně konfigurace restartujte ser2net server:

```
Shell:~$ /etc/init.d/ser2net restart
```

Na klientském PC pak stačí spustit:

```
Shell:~$ sudo socat PTY,link=/dev/YPort TCP:192.168.2.116:3333
```

IP adresa musí být samozřejmě IP vašeho serveru a místo */dev/Yport* může být jakýkoliv jiný soubor, na který se poté budete odkazovat jako na sériový port.

Pokud někomu vadí, že komunikace není šifrovaná, tak si prostudujte ssh a tunelovaní portu a můžete si zřídit šifrovanou linku,případně i socat (rozšířená varianta netcat) má umět šifrovaní, ale nezkoumal jsem.

Více na:
http://techtinkering.com/2013/04/02/connecting-to-a-remote-serial-port-over-tcpip/
http://unix.stackexchange.com/questions/100845/serial-data-over-ethernet-on-a-linux-box
http://www.dest-unreach.org/socat/doc/socat-ttyovertcp.txt
https://blog.philippklaus.de/2011/08/make-rs232-serial-devices-accessible-via-ethernet
Odkazy kde najít ovladače pro OS Windows (nezkoušel jsem):
https://www.remoteqth.com/wiki/index.php?page=Ser2Net#How_To_Connect_to_ser2net_from_Windows
http://www.hw-group.com/products/hw_vsp/index_en.html

RS485(481)
jde také o sériovou komunikaci oproti RS232 která má pro příjem a vysílání samostatnou linku, má RS485 jen jednu linku na které se musí střídat vysílaní a příjem. Na tuto linku se často navěšuje více zařízení, teoreticky co do elektrické zátěže jich snese až 32. S principu fungování a správného střídaní vysílaní a příjmu musí jedno z těchto zařízení provoz řídit a ostatní ho poslouchat (řídicímu se říká master a ostatní jsou slave). V modulu SEAHU SH017 přepínaní příjmu a vysílaní řídí pin11 z raspberry PI (1-vysílá, 0-přijímá). A protože řízení směru se řeší softwarově, nelze RS485 používat stejně jako RS232, na přímé použití pro terminál, nebo přemostění portu můžete zapomenout.
Nejčastější využití RS485 bude spolu s protokolem modbus, ten je v Linuxu nejčastěji používán s knihovnou http://libmodbus.org/ (knihovna funkci v programovacím jazyce C). V této knihovně je myšleno i na alternativní přepínaní vysíláni a příjmu. Pomocí funkce modbus_rtu_set_custom_rts(modbus_t *ctx, void (*set_rts) (modbus_t *ctx, int on)) můžete knihovně vnutit vlastní funkci pro toto přepínaní. Víc jsem to nezkoumal.

Ukázka komunikace RS485 v programovacím jazyce python.

rs485master.py

```
#!/usr/bin/python

import serial
import RPi.GPIO as GPIO
import time

pin_RTS=11
GPIO.setmode(GPIO.BOARD)
#GPIO.setmode(GPIO.BCM)
GPIO.setup(pin_RTS, GPIO.OUT)
GPIO.output(pin_RTS, 1) #defaul value=transmission

ser = serial.Serial(
    port='/dev/ttyAMA0',
    baudrate = 9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=1
    )

print ("Test RS485 Master side:")

ser.flushInput()

counter=0
while True:
    GPIO.output(pin_RTS, 1) #transmission
    ser.write('Write counter: %d \n'%(counter)) #this only very quicky send data to bufer
not wait to send
    time.sleep(0.1) #wait to send date over serial line
    GPIO.output(pin_RTS, 0) #receiver
    receive = ser.read(100)
    if len(receive)==0 : print ("No slave contact.")
    else : print (receive)
    time.sleep(1)
    counter += 1
```

rs485slave.py

```
#!/usr/bin/env python

import time
import serial
import RPi.GPIO as GPIO

pin_RST=11
GPIO.setmode(GPIO.BOARD)
#GPIO.setmode(GPIO.BCM)
GPIO.setup(pin_RST, GPIO.OUT)
GPIO.output(pin_RST, 1) #defaul value=transmission


ser = serial.Serial(
    port='/dev/ttyS0',
    baudrate = 9600,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
    timeout=0.1
    )
print ("Test RS485 Slave side:")

ser.flushInput()

GPIO.output(pin_RST, 0) #receiver
last_time=time.time()
while 1:
    receive = ser.read(100)
    if len(receive)==0 :.
        if (time.time()-last_time)>3 :
            print ("No master contact.")
            last_time=time.time()
    else :.
        last_time=time.time()
        print (receive)
        GPIO.output(pin_RST, 1) #transmission
        ser.write(receive) #this only very quicky send data to bufer not wait to send
        time.sleep(0.1) #wait to send date over serial line
        GPIO.output(pin_RST, 0) #receiver
```

Na jednom počítači (modulu) se spustí program pro master a na druhem program pro slave. Ukázkový master program potom slave programu posílá hlášky a ten je vrací master programu.

Více informací:

- web raspberryPI projektu: https://www.raspberrypi.org/
- web seahu: http://www.seahu.cz

Napsal:

Ing. Ondřej Lyčka leden 2017

Verze dokumentu: 1.00