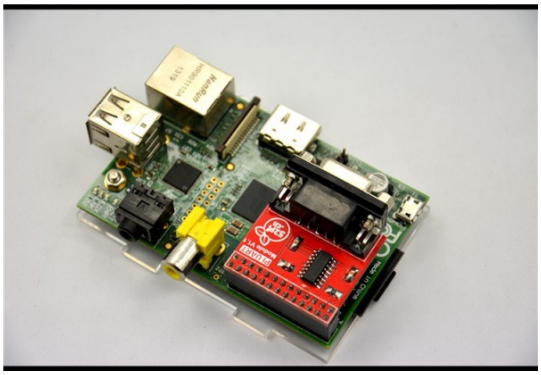
UART - RS232 on the Raspberry Pi



The Raspberry Pis have two built-in UARTs, a PL011 and a mini UART. They are implemented using different hardware blocks, so they have slightly different characteristics. However, both are 3.3V devices, which means extra care must be taken when connecting up to an RS232 or other system that utilises different voltage levels. An adapter must be used to convert the voltage levels between the two protocols. Alternatively, 3.3V USB UART adapters can be purchased for very low prices.

It is important to note that for the **Raspberry Pi 3** and **Raspberry Pi Zero W**, the PL011 UART is connected to the Bluetooth module, while the mini UART is used as the primary UART and will have a Linux console on it. **On all other models, the PL011 is used as the primary UART**.

In Linux device terms, by default, /dev/ttyS0 refers to the mini UART, and /dev/ttyAMA0 refers to the PL011. The primary UART is the one assigned to the Linux console, ***which depends on the Raspberry Pi model as described above.*** There are also symlinks: /dev/serial0, which always refers to the primary UART (if enabled), and /dev/serial1, which similarly always refers to the secondary UART (if enabled).

As we have a Raspberry Pi 3 device, by default the PL011 UART is connected to the Bluetooth module. To use the PL011 as the primary UART, the pi3-miniuart-bt UART Device Tree Overlay should be used instead. This device tree switches the Bluetooth function to use the mini UART (ttyS0), and restores UART0/ttyAMA0 to GPIOs 14 and 15.

Here bellow in the following pages, we explain how to switch and restore the UART0 /dev/ttyAMA0 to GPIOs 14 and 15 respectively, which are pins 8 and 10 on the GPIO header.

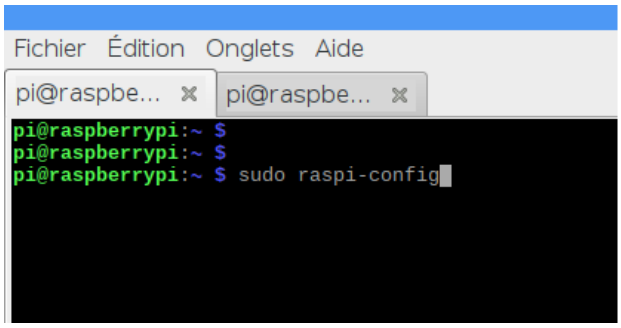
(*) Relevant differences between PL011 and mini UART

The mini UART has smaller FIFOs. Combined with the lack of flow control, this makes it more prone to losing characters at higher baudrates. It is also generally less capable than the PL011, mainly due to its baud rate link to the VPU clock speed.

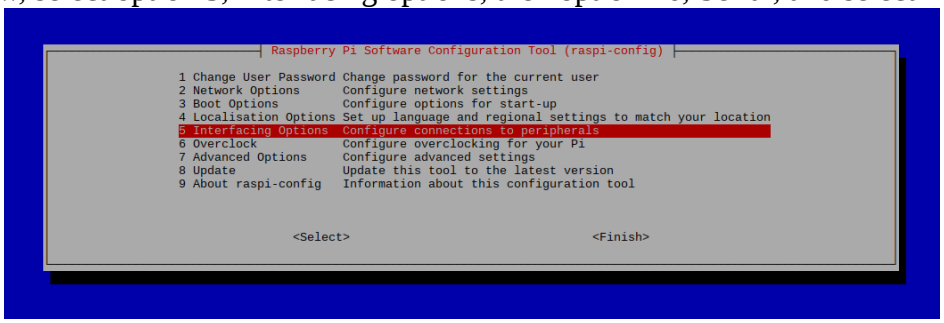
The particular deficiencies of the mini UART compared to the PL011 are :

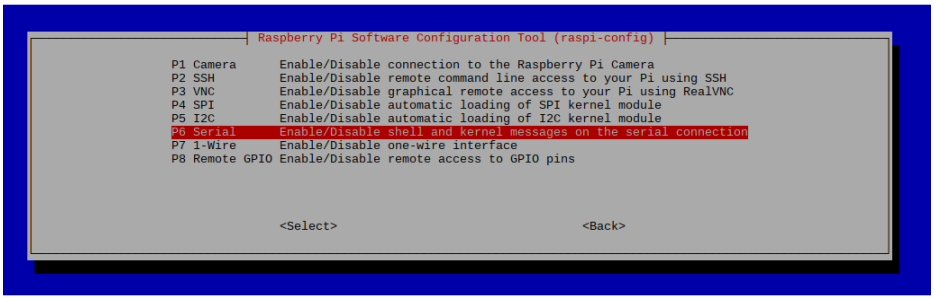
- No break detection
- No framing errors detection
- No parity bit
- No receive timeout interrupt
- No DCD, DSR, DTR or RI signals

First let's open the configuration tool after this, simply run the following from the command line:

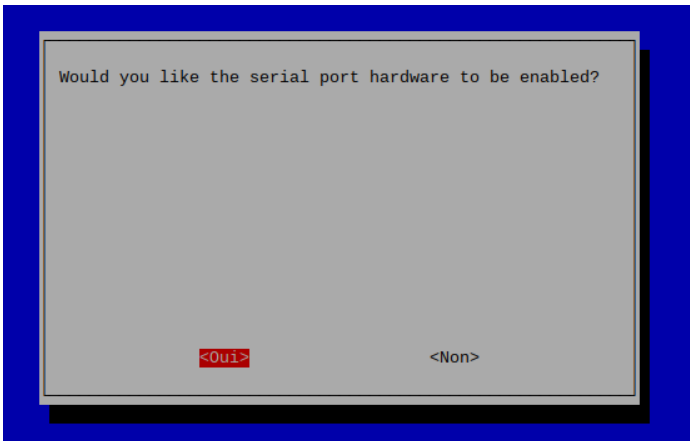


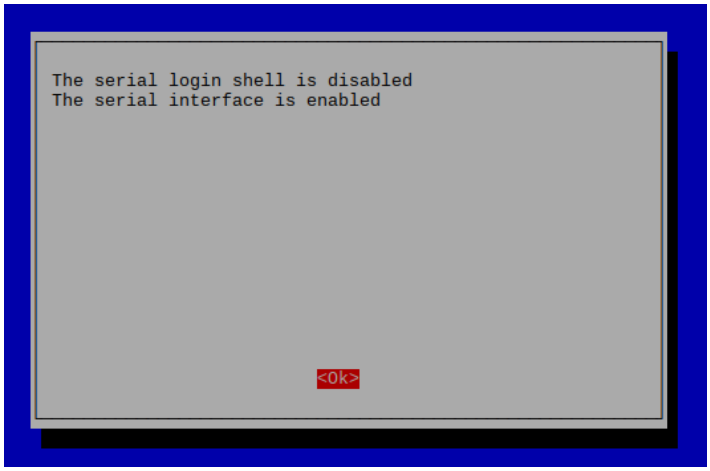
As shown below, select option 5, Interfacing options, then option P6, Serial, and select No.



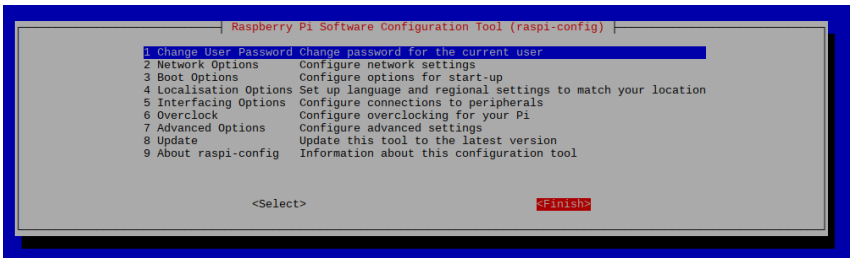


Disable Linux's use of console UART.

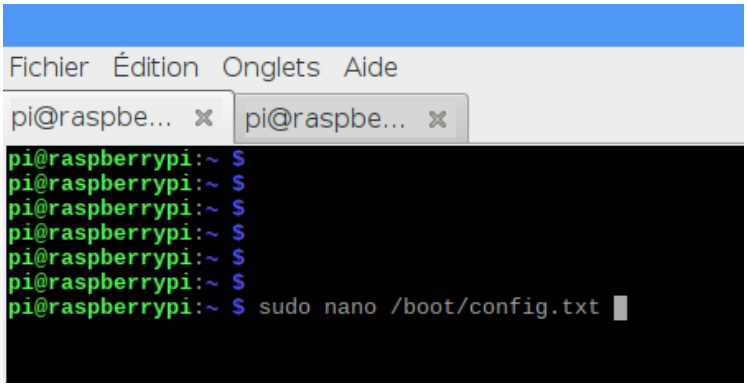


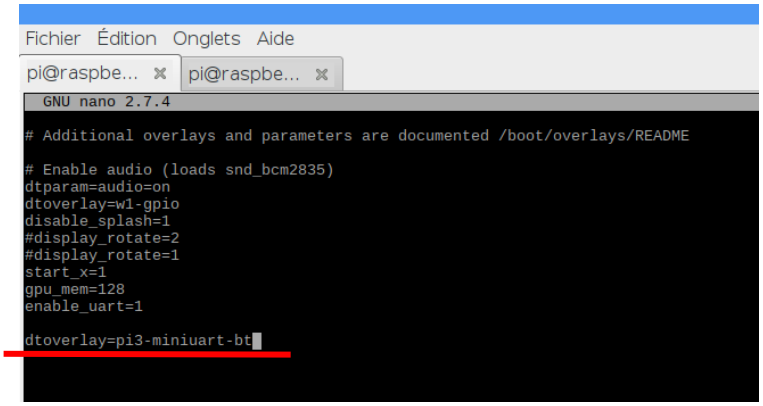


Exit raspi-config.

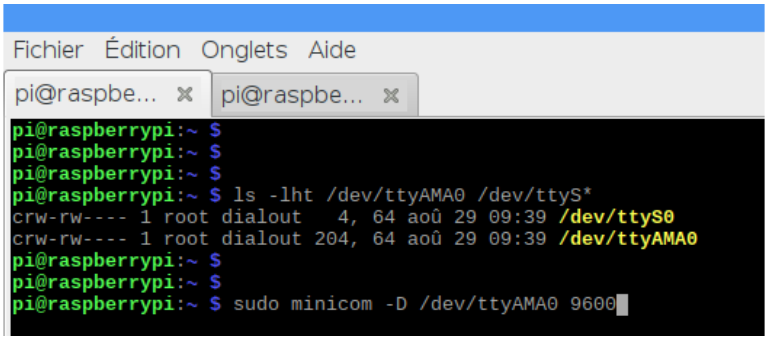


Add a line to the config.txt file to enable Device Tree Overlays as shown below:





Now reboot the raspberry and let's check if we can connect to the principal UART PL011



Symlinks

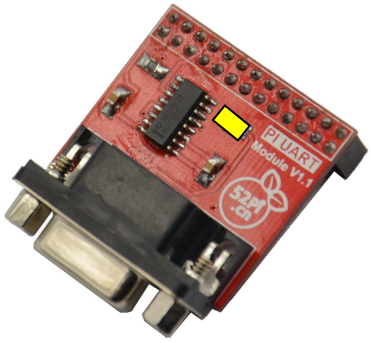
```
Fichier  Édition  Onglets  Aide
pi@raspbe...  pi@raspbe...

pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $ ls -l /dev/serial*
lrwxrwxrwx 1 root root 7 aoû 30 09:51 /dev/serial0 -> ttyAMA0
lrwxrwxrwx 1 root root 5 aoû 30 09:51 /dev/serial1 -> ttyS0
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $
```

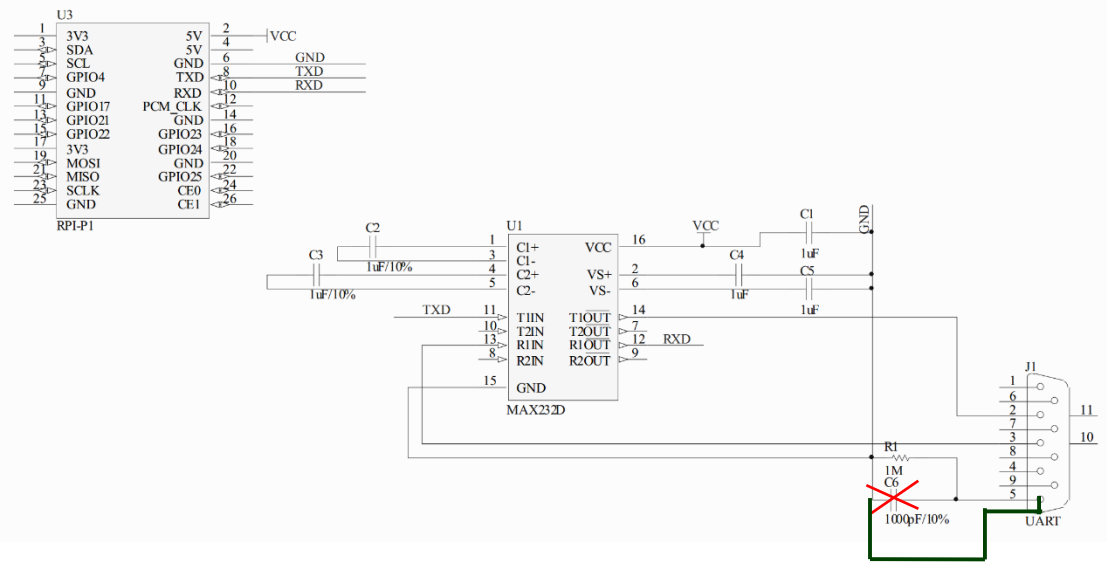
To test our UART-RS232 module, we simply create a loopback on the DB9 connector .
Connect the Tx pin2 to Rx pin3 to have an echo of each character sent.


```
Fichier  Édition  Onglets  Aide
pi@raspbe...  pi@raspbe...

Bienvenue avec minicom 2.7

OPTIONS: I18n
Compilé le Apr 22 2017, 09:14:19.
Port /dev/ttyAMA0, 11:15:04

Tapez CTRL-A Z pour voir l'aide concernant les touches spéciales

aa

bb

cc

dd
```

serial communication program, minicom

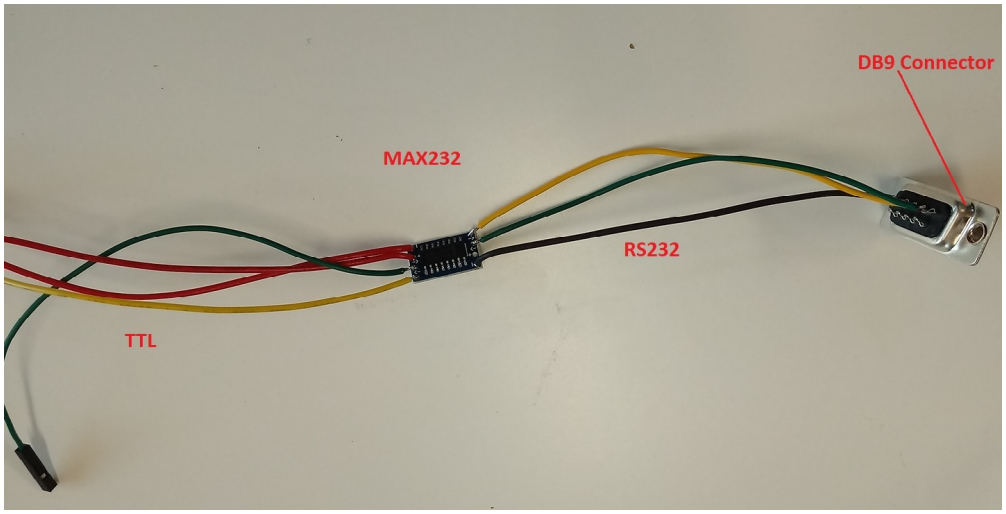
For our test and experiment, the 52PI-RPI-UART expansion module has been used.





We had some issue to communicate properly with our devices due to hardware. This communication problem has been solved by removing the C6 1000pF component and shorting the connection.

We can also create our own uart-RS232 converter using a TTL to RS232 converter as shown below:



Previously I have used the the friendly serial communication program, minicom.

Another elegant way to test the uart-rs232 could be performed by using the pySerial module. This module encapsulate the access for the serial port for python language.

```
Fichier  Édition  Onglets  Aide
pi@raspbe...  pi@raspbe...
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $ sudo python3 rs232-uart_test.py
/dev/ttyAMA0
b'uart-rs232 test\n'
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $ nano rs232-uart_test.py
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $ sudo python3 rs232-uart_test.py
/dev/serial0
b'uart-rs232 test\n'
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $
pi@raspberrypi:~/rs232-uart_module $
```

Measuring RS232 signals' voltage using Oscilloscope