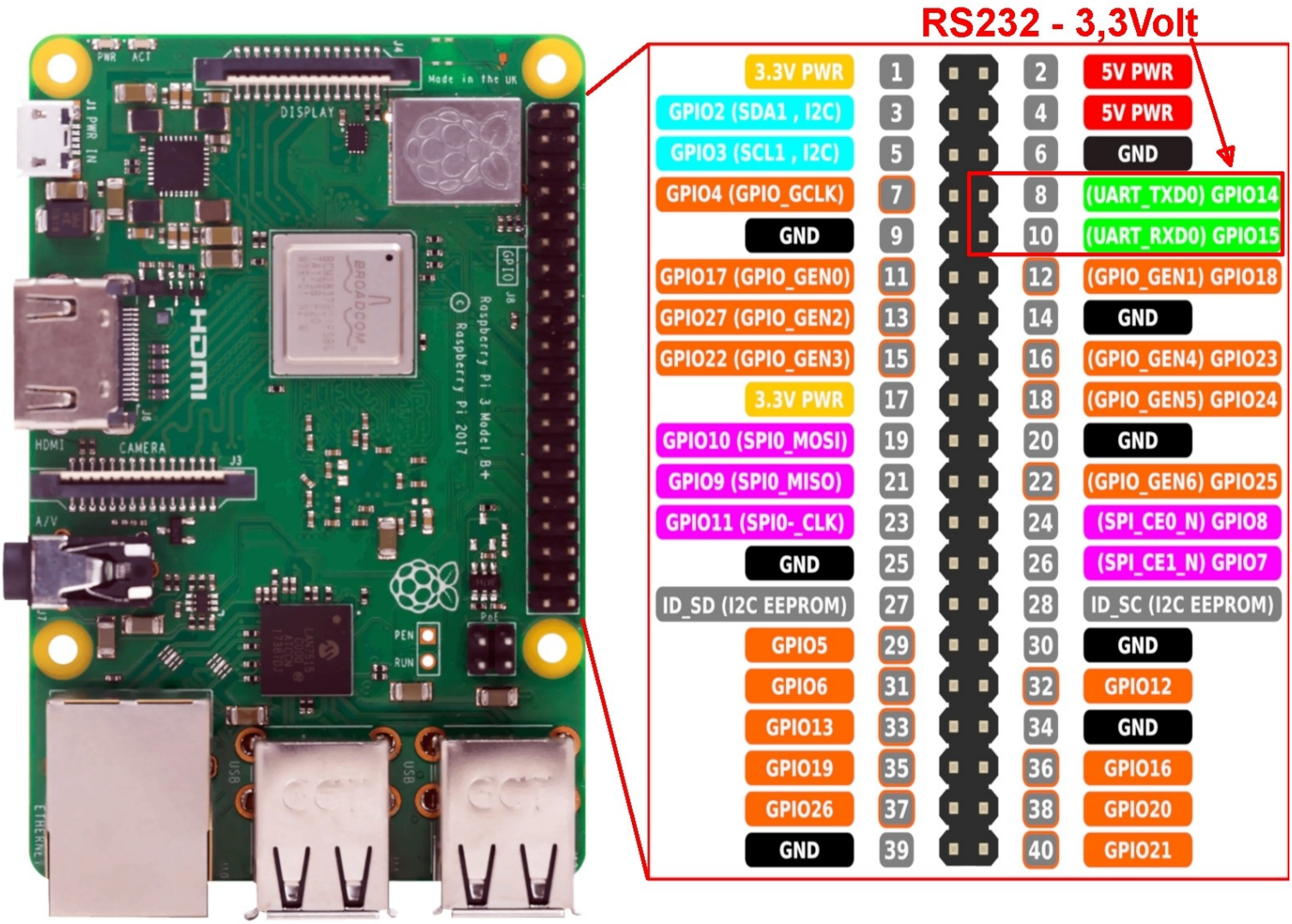
RASPBERRY - PORTA SERIALE RS232

Sul connettore della scheda Rapberry sono presenti i vari GPIO (General Purpouse Input ed Output).

Occorre ricordare che tutti i piedini funzionano con una tensione di 3,3Volt, **pertanto debbono essere collegati obbligatoriamente a dispositivi che funzionano con la stessa tensione per evitare la sicura rottura della scheda.**

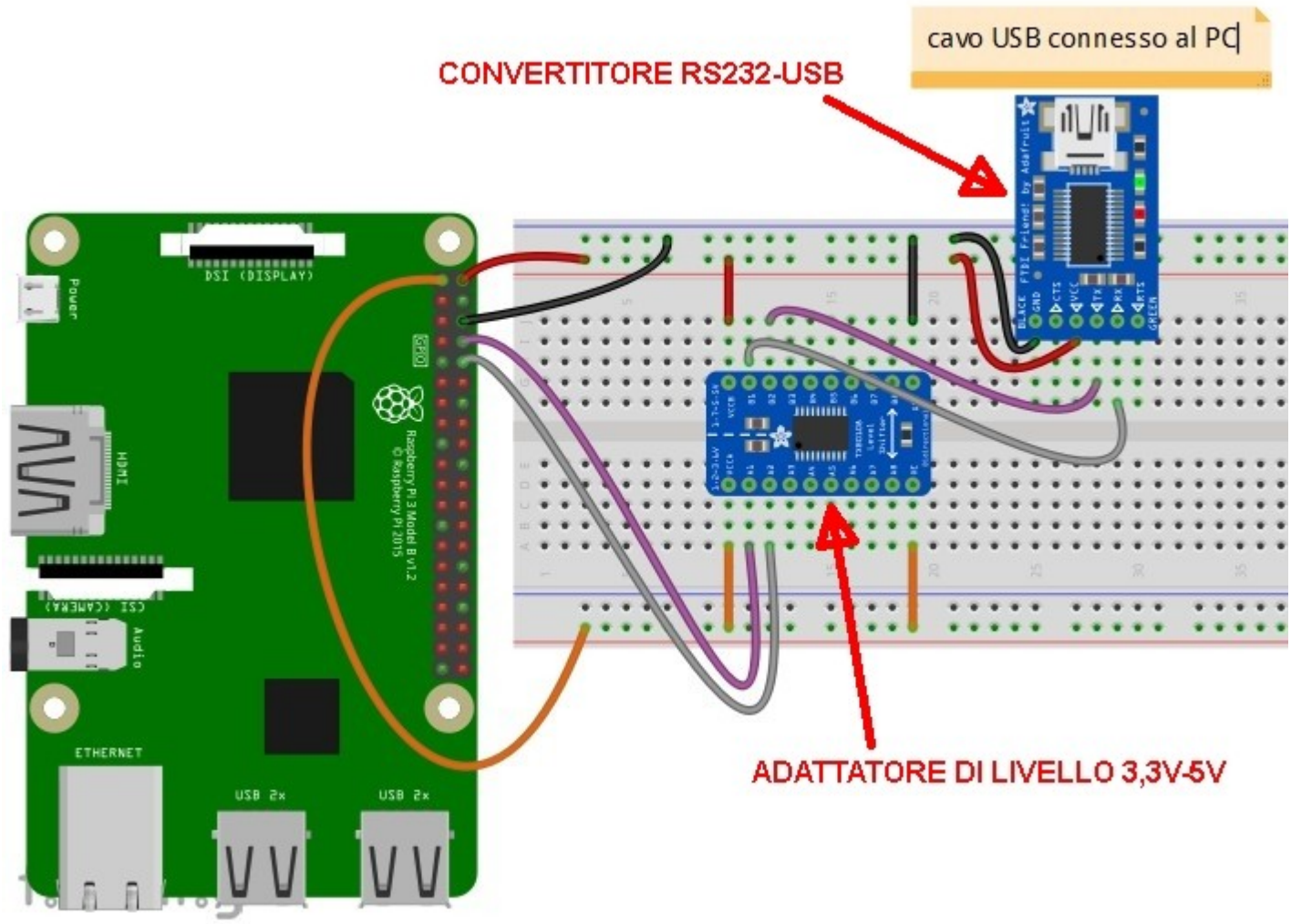
Tutti i piedini sono infatti direttamente collegati al microprocessore della scheda senza alcuna protezione.

Dopo questa doverosa premessa andiamo a vedere in figura quali sono i due piedini TX e RX che sono utilizzati dalla scheda per la comunicazione seriale USART RS232.



Come detto sopra i due piedini funzionano con una tensione di 3,3Volt, pertanto è indipensabile nella maggior parte dei casi, utilizzare un traslatore di livello. Di seguito 3 possibili collegamenti tramite interfaccia RS232 della scheda Raspberry.

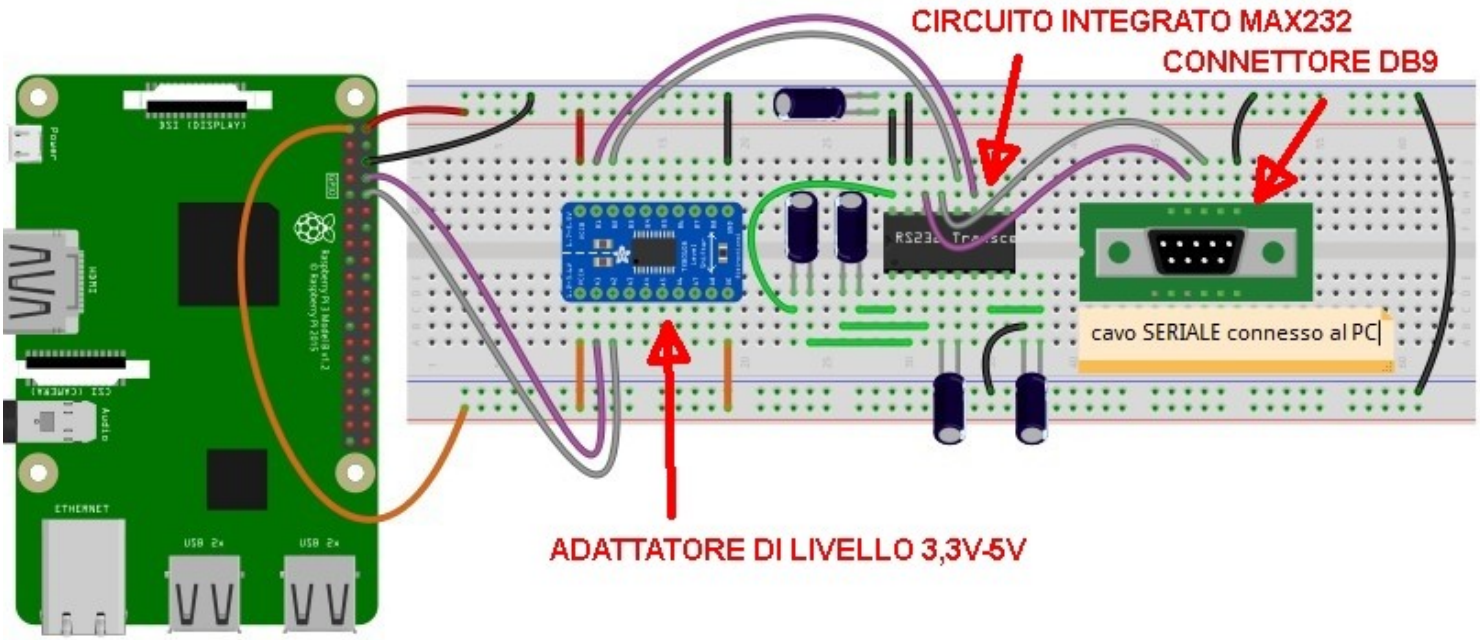
1. CONNESSIONE AD UN PC TRAMITE USB



In questo caso sono necessari due dispositivi, il primo al centro dello schema si occupa di traslare i livelli da 3,3Volt a 5Volt e viceversa, il secondo in alto si occupa di interfacciare la comunicazione RS232 con la USB del PC. Questo dispositivo necessita dell'installazione dei propri driver sul PC, in questo modo nel PC avremo una porta seriale (Indicata con COM) che utilizza il protocollo di comunicazione USART RS232.

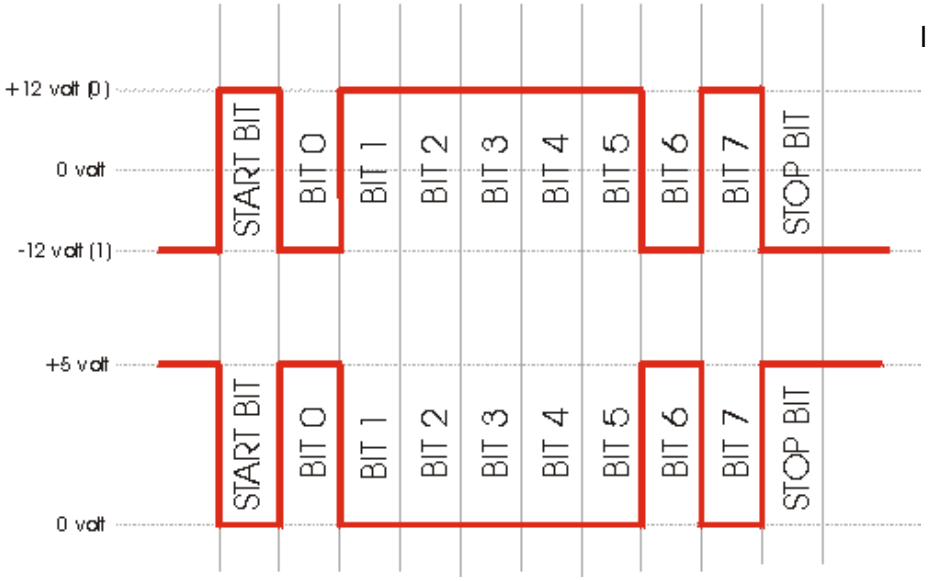
I moderni PC domestici, non montano più le porte seriali RS232, in quanto utilizzate ancora solo in ambito industriale. Per questo motivo si trovano in commercio dei cavi che utilizzano al proprio interno lo stesso circuito indicato sopra. Nella stragande maggioranza dei casi il chip presente nel circuito convertitore è costruito dalla FTDI (Future Technology Devices International) https://www.ftdichip.com/index.html
Una ditta che realizza anche altri convertitori sempre con la connessione USB.

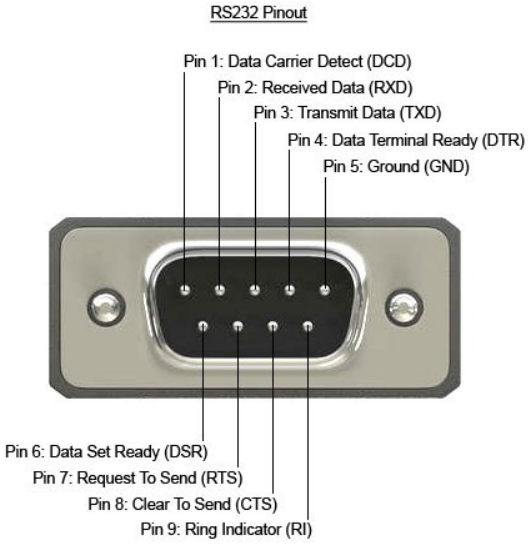
2. CONNESSIONE AD UN PC TRAMITE PORTA SERIALE 9 PIN



In questo caso occorre sempre i circuito adattatore di livello da 3,3V a 5V per connettersi al MAX232, un circuito integrato che converte ulteriormente il segnale dalla logica RS232 TTL alla logica prevista dal protocollo RS232 che utlizza segnali +12Volt e -12Volt..

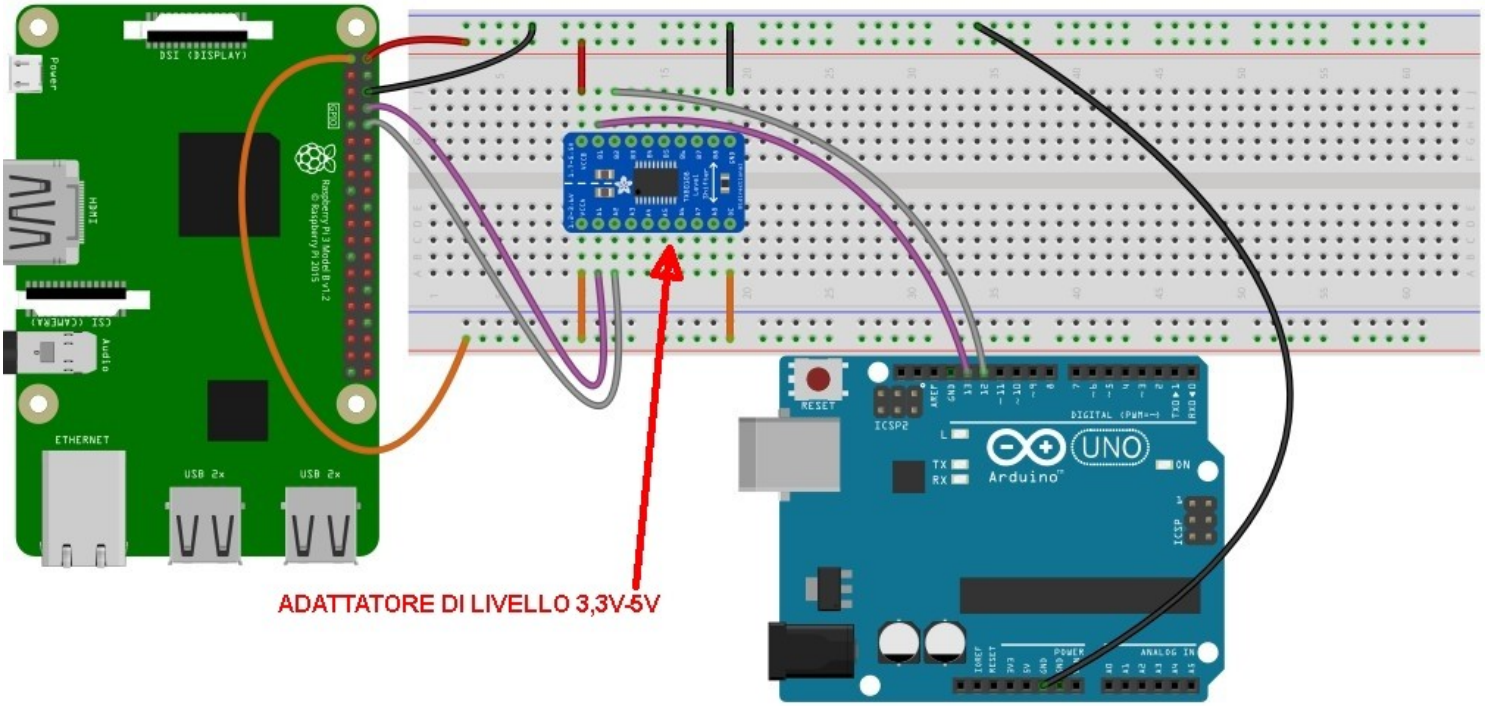
Si trovano in commercio integrati che convertono il segnale dalla logica TTL partendo da un valore di 3,3V in questo caso non occorre l'adattatore di livello.





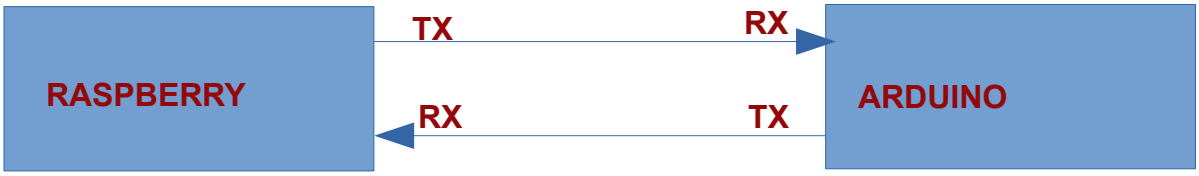
Nell'immagine sopra vediamo in alto il segnale secondo lo standard RS232, ed in basso lo stesso segnale secondo lo stesso protocollo di comunicazione con logica TTL. Sui connettori a vaschetta DB9 come quello in figura troviamo il segnale +12V -12V.

3. CONNESSIONE AD ARDUINO



In questo caso occorre solo adattare il livello del segnale che nella scheda Raspberry è a 3,3Volt e nella scheda Arduino è 5Volt.

Ovviamente il segnale TX della scheda Raspberry andrà collegato al segnale RX di Arduino ed allo stesso modo RX di Raspberry andra con TX di Arduino.



Sul software di Arduino andrà perciò utilizzata la libreria **SoftwareSerial.h** nel seguente modo.

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(13, 12);  //pin13=RX pin12=TX

void setup() {
  mySerial.begin(9600);           //velocità in bps
  mySerial.println("Benvenuto"); //scrittura su seriale
}

void loop() {
  if (mySerial.available()) {       //attesa ricezione
    Serial.write(mySerial.read()); //lettura dato
                                    //e scrittura di quanto
                                    //ricevuto (eco)
  }
}
```

ATTIVAZIONE DELLA SERIALE

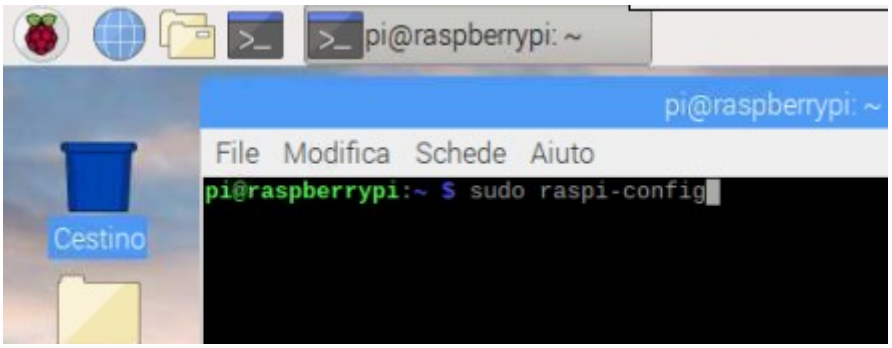
Fino ad ora abbiamo affrontato l'aspetto hardware della questione, ora vediamo invece l'aspetto software della lezione dal lato della scheda Raspberry.

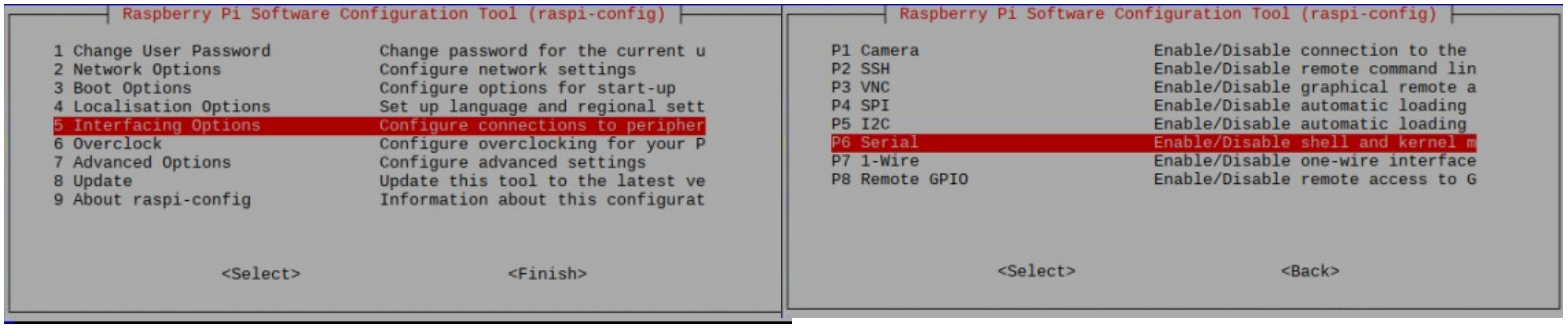
Innanzittutto dobbiamo ricordarci di attivare la seriale sulla scheda, per fare questo dobbiamo accedere al menu di configurazione tramite terminale o tramite interfaccia grafica.

Nel primo modo occorre aprire un temrinale e digitare il comando: ***sudo raspi-config***

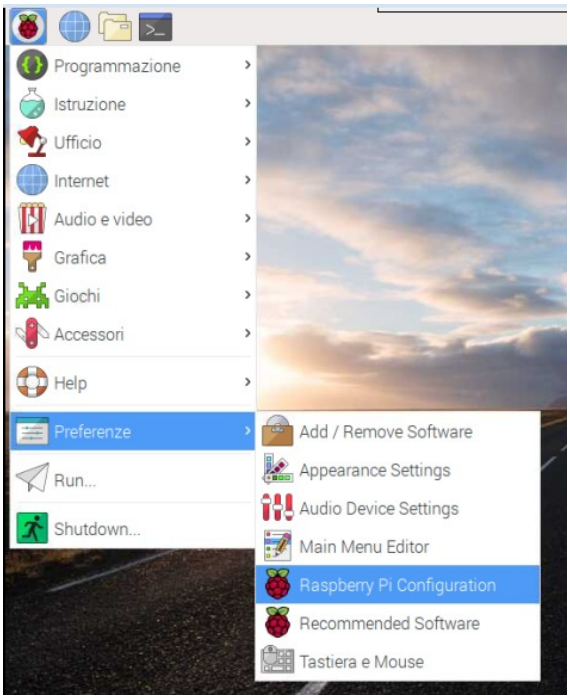
Successivamente accedere alla sezione **Interface Options** e successivamente al punto **P6 Serial.**

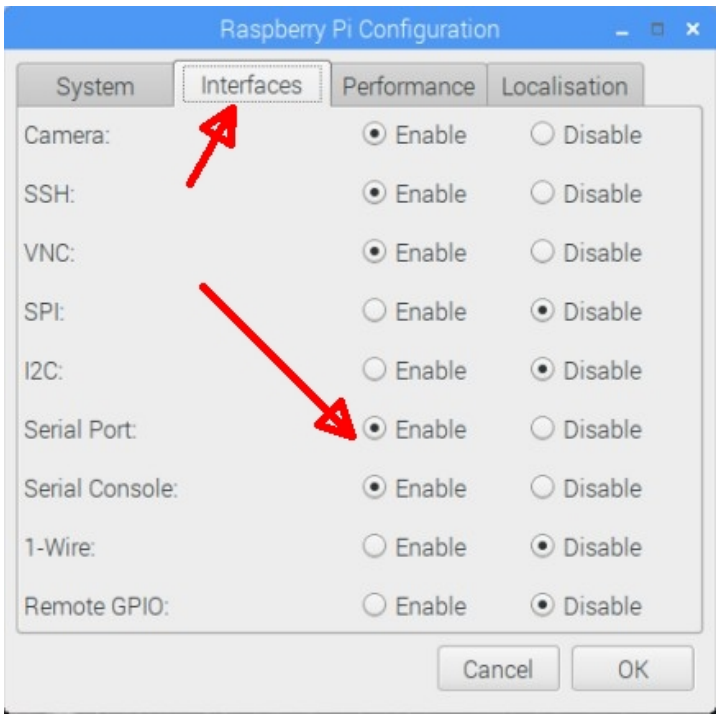
Occorre abilitare la seriale e riavviare.





Si può anche attivare la seriale accedendo alla configurazione della scheda da interfaccia grafica.





Dopo questa operazione avremo a disposizione la seriale RS232 sui due piedini del connettore della Rapberry.

PORTE SERIALI PRESENTI SU RASPBERRY

In realtà nel S.o.C (System On a Chip) della Raspberry sono presenti due periferiche hardware USART.

Una periferica PL011 (periferica hardaware implementata nei procesori ARM) ed una mini UART, entrambi sono periferiche seriali RS232 a 3,3 Volt implementate nel sistema.

PL011 http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0183g/index.html

mini UART https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf

Nella scheda Raspberry Pi 3, la prima viene utilizzata per comunicare con il modulo Bluethoot-Wifi presente sulla scheda, e nel filesystem LINUX, questa periferica viene gestita dal file ***/dev/ttyAMA0***.

La seconda viene invece utilizzata per l'accesso da consolle e per inviare messaggi durante la fase di BOOT, e viene gestita dal file ***/dev/tty/S0***.

Per impostare la velocità o altre carattesitiche delle porte seriali, come anche per verificare le impostazioni date si può utilizzare i comando ***stty***, ad esempio possiamo digitare ***sudo stty -F /dev/ttyS0*** ed ottenere in risposta quanto segue:

```
pi@raspberrypi:~ $ sudo stty -F /dev/ttyS0
speed 115200 baud; line = 0;
min = 1; time = 0;
-brkint -icrnl -imaxbel iutf8
-isig -icanon -iexten -echo -echoe -echok -echoctl -echoke
```

Oppure possiamo digitare ***sudo stty -F /dev/ttyS0 9600*** per assegnare alla periferica una velocità di 9600 bps, dopo la modifica delle impostazioni della seriale occorre sempre riavviare.

Tutte le informazioni sulla sintassi corretta del comando e sui suoi parametri le possiamo avere digitando ***man stty***.

A questo link tutte le informazioni necessarie: https://elinux.org/RPi_Serial_Connection

Come detto sopra nella versione Raspberry Pi 3, la periferica PL011 gestita dal file /*dev/*ttyAMA0 ed è collegata al modulo wifi interno alla scheda, mentre la seconda utilizzata per l'accesso da seriale, è fisicamente collegata ai piedini presenti sul connettore delle GPIO.

Questo discorso non vale per tutte le versioni della scheda Rapberry, in altre versioni la seriale presente sul connettore è invece quella gestita dal file /dev/AMA0.

Essendo le due periferiche seriali differenti (la PL011 è più affidabile in caso di alta velocità di tramissione) potrebbe essere utile modificare la seriale presente sul connettore delle GPIO, ciò è ovviamente possibile, tutte le informazioni sono presenti nel seguente link: https://www.raspberrypi.org/documentation/configuration/uart.md alla voce **UARTs and Device Tree.**

Torniano comunque al nostro caso, volendo utilizzare la seriale presente sul connettore dobbiamo gestire il file ttyS0, ma prima di tutto dobbiamo eseguire tre operazioni fondamentali:

- **ABILITARE LA SERIALE miniUART NEL FILE CONFIG.TXT**
- **ASSEGNARE I PERMESSI AL FILE DELLA SERIALE**
- **DISATTIVARE SERIALE PER LOG DI BOOT E PER ACCESSO CONSOLLE**

ABILITARE LA SERIALE miniUART NEL FILE CONFIG.TXT

Per abilitare la mini UART come scheda primaria del sistema occorre editare il file ***/boot/config.txt*** aggiungendo la riga ***enable_uart=1***

Il baud-rate (velocità) di trasmissione della seriale, dipende dalla frequenza del core della GPU, abilitando la miniUART come primaria, si correggerà il clock del core a 250MHz (se l'opzione force_turbo non è abilitata).

ASSEGNARE I PERMESSI AL FILE DELLA SERIALE

Ogni dispositivo nel sistema operativo Linux viene trattato come fosse un file appartenente al suo filesystem. Allo stesso modo troviamo la seriale sotto la directory **/dev** chiamata con il nome **ttyS0**.

Una delle cose da fare pertanto è dare i permessi all'utente per l'utilizzo di questo file con il comando **chmod.**

Con "**sudo chmod 666 ttyS0"** diamo a tutti gli utenti il permesso di lettura e di scrittura nel file.

Possiamo verificare la corretta assegnazione dei permessi con il comando **"ls -l ttyS0"** in risposta dovremo ottenere "**rw-rw-rw-".**

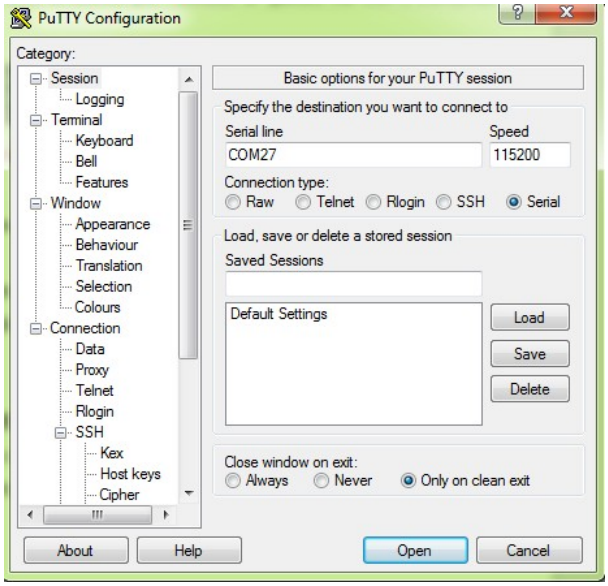
```
pi@raspberrypi:/dev $ sudo chmod 666 ttyS0
pi@raspberrypi:/dev $ ls -l ttyS0
crw-rw-rw- 1 root tty 4, 64 apr  7 22:56 ttyS0
```

PROPRIETARIO GRUPPO UTENTI

Se invece non si vogliono modificare i permessi al file si può agire assegnando l'utente **pi** al gruppo **dialout** ed al gruppo tty con il seguente comando: **sudo usermod -a -G dialout pi**

DISATTIVARE SERIALE PER LOG DI BOOT E PER ACCESSO CONSOLLE

Se proviamo a lanciare putty ed attiviamo una comunicazione seriale con la scheda Raspberry, possiamo accedere al terminale per eseguire qualsiasi funzione.Inoltre durante l'accensione e lo spegnimento della scheda, vengono inviati dei messaggi di log sempre sulla stessa seriale. Per questo motivo per utilizzare la seriale per altre applicazioni, è consigliabile disabilitare sia l'invio dei messaggi di log, che la possibilità di login da seriale.



Per disattivare la possibilità di accesso da seriale si può agire tramite la configurazione ***sudo raspi-config*** sulla sezione **interfacing options** e successivamente **serial.**

Occorre disattivare la prima voce e non la seconda

```
The serial login shell is disabled
The serial interface is enabled
```

Oppure si possono lanciare i seguenti comandi:

sudo systemctl stop serial-gettyòttyS0.service
sudo systemctl disable serial-gettyòttyS0.service

e successivamente editare il file ***sudo nano /boot/cmdline.txt*** togliendo la parte ***console=serial0,115200.***

GESTIONE SERIALE CON PYTHON

Tutte le informazioni necessarie si possono trovare al link: https://www.elinux.org/Serial_port_programming .

Per gestire la porta seriale tramite Python, occorre innanzitutto installare le librerie necessarie con il comando:

sudo apt-get install python-serial

La documentazione per la libreria si può trovare al seguente link: https://pyserial.readthedocs.io/en/latest/pyserial.html

Successivamente si può accedere alla porta seguendo il seguente esempio:

```
import serial

port = serial.Serial("/dev/ttyS0", baudrate=115200, timeout=3.0)

while True:
   port.write("Scrivi qualcosa\r\n")   #invio la stringa sulla seriale
   rcv=port.read(10)                   #leggo dalla seriale
   port.write("Hai scritto:" + repr(rcv)) #invio una stringa con quanto ricevuto
   port.write("\r\n")     #nuova linea
```

Nell'esempio precedente viene dichiarata una seriale chiamata "port" con elocità 115200bps.

Successivamente in un ciclo infinito, viene inviata una stringa con l'istruzione port.write, e viene letto il carattere o la stringa riccevuta con l'istruzione port.read.

Per provare questo programma si possono utilizzare i primi due schemi con un terminale seriale da PC.

Per provare invece la connessione della scheda Raspberry con Arduino (il 3° schema) occorre adattare i due programmi di esempio visti fino ad ora.

GESTIONE SERIALE IN C

Di seguito un esempio di scrittura e lettura tramite seriale RS232, occorre lanciare un monitor seriale da PC (hyperterminal o altri) e mandare in esecuzione il seguente programma di esempio, dopo averlo scritto con il comando ***sudo nano rs232.c*** e salvato con lo stesso nome (CTRL+X) il programma va compilato mediante il comando ***sudo gcc rs232.c -o rs232.out*** e va avviato scrivendo ***./rs232.out.***

A questo punto si può lanciare un monitor seriale da PC impostando la velocità di comunicazione ***115200 bps un bit di stop no parità***, e leggere la stringa seriale provando ad inviare qualcosa.

```
#include <termios.h>
#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#define DATA_RX 50  // buffer di ricezione
#define PORTA_SERIALE "/dev/ttyS0" //percorso porta seriale utilizzata
#define SERIAL_SPEED B115200  //velocità in bps

int flag_seriale;
```

```c
//apertura e configurazione porta seriale
int configura_seriale()
 {
    char esito_config;
    struct termios serial_pi;
    //apertura porta seriale in lettura e scrittura
    // O_RDWR=read and write O_RDONLY=read only  O_WRONLY=write only
    flag_seriale = open (PORTA_SERIALE, O_RDWR);
    if (flag_seriale < 0)
      {
        printf ("Errore apertura seriale %s\n", strerror (errno));
        return (-1);
      }//if (flag_seriale < 0)
    //impostazione della seriale
    cfsetospeed (&serial_pi, SERIAL_SPEED);              // Set output speed in bps
    cfsetispeed (&serial_pi, SERIAL_SPEED);              // Set input speed in bps
    serial_pi.c_cflag = (serial_pi.c_cflag & ~CSIZE) | CS8;     // 8-bit
    serial_pi.c_cc[VTIME] = 10;                          // 1 second read timeout
    serial_pi.c_iflag &= ~(IXON | IXOFF | IXANY);        // No xon/xoff ctrl
    serial_pi.c_cflag |= (CLOCAL | CREAD);               // Ignore modem controls, enable reading
    serial_pi.c_cflag &= ~(PARENB | PARODD);             // No parity
    serial_pi.c_cflag &= ~CSTOPB;                        // Set bit stop
    serial_pi.c_cflag &= ~CRTSCTS;                       // No handshake
    esito_config=tcsetattr (flag_seriale, TCSANOW, &serial_pi);
    if(esito_config!=0)
      {
        printf("Errore configurazione seriale %s\n", strerror(errno));
        return (-2);
      }//if (esito_config)!=0)
    return (0);
}//int configura_seriale()
```

```c
int main()
{
  char lettura_seriale [DATA_RX];  // Data buffer (Rx)
  char esito;
  int cont_invio=0,flag_read;

  printf("Rapsberry Pi seriale RS232 in C\n");

  esito=configura_seriale();

  while(esito==0)
   {
    cont_invio++;
    printf("Numero invio %d\n",cont_invio);
    write (flag_seriale, "Scrivi qualcosa\r\n",20 );
    memset (lettura_seriale, 0, sizeof(lettura_seriale) ); // Vuota il buffer dei dati
    sleep(1);
    flag_read=read(flag_seriale,lettura_seriale,sizeof(lettura_seriale));
    if(flag_read<0)
      {
        printf("Errore lettura! %s\n", strerror(errno));
        esito=1;
      }//if(flag_read<0)
    else   printf("Dato letto [%d bytes]: %s\n\n", flag_read, lettura_seriale);
   }//while(esito=0)

   close (flag_seriale);
}
```