LE BASI IN MBED.COM

Andare su https://www.mbed.com/ e registrarsi alla piattaforma

Andare su https://os.mbed.com/platforms/ST-Discovery-L475E-IOT01A/ aggiungere la scheda al proprio mbed compiler (nella barra laterale destra)

Importare uno o più esempi nel compilatore tra quelli proposti nella barra laterale destra.

I più interessanti per iniziare sono STM32_Blink_LED e HelloWorld_ST_Sensors.

Ogni programma è scritto in C/C++ e il file main.cpp rispetta la seguente struttura:

```
IMPORTAZIONE LIBRERIE
SET UP VARIABILI GLOBALI
int main() {
   SET UP SENSORI
   while (1) {
      CICLO DI ESECUZIONE INFINITO
   }
}
```

Quando viene caricato un file binario sulla scheda, dopo la prima parte di setup, viene eseguito a ciclo continuo il contenuto del while.

In certi esempi non è esplicito il **while** (1) ma sono presenti funzioni semanticamente identiche:

```
eventQueue.call_every(500, periodicCallback);
eventQueue.dispatch_forever();
```

Per sviluppare su questa piattaforma sono disponibili le risorse:

- Guida C++: http://www.cplusplus.com/doc/tutorial/
  per la sintassi del linguaggio C/C++.
- Documentazione mbed: https://os.mbed.com/docs/mbed-os/v5.14/quick-start/index.html
  per utilizzare le funzioni di interfaccia alla scheda attraverso la libreria mbed.

Oltre a ulteriori guide/risorse/tutorial disponibili in rete.

INVIO DATI TRAMITE SERIALE

La scheda STM32 Discovery, ogni volta che esegue una stampa su standard output, invia i byte stampati sulla porta seriale con cui è collegata.

Invio dati dalla scheda STM32

Ci sono più modi per inviare dati su standard output.

Il metodo più semplice invia array di byte (stringhe) tramite la funzione printf.

```
while(1) {
   printf("stringa su stdout\r\n");
   wait(5); //attendi 5 secondi
}
```

Che invia attraverso la porta seriale 19 byte in notazione esadecimale:

73 74 72 69 6e 67 61 20 73 75 20 73 74 64 6f 75 74 0d 0a

In alternativa è possibile inviare direttamente il contenuto di una variabile con la funzione fwrite oppure un singolo carattere con putc.

```
int number = 4250;
while(1) {
   fwrite( &number, sizeof(int),1, stdout);
   fputc('\n', stdout);
   wait(5); //attendi 5 secondi
}
```

Che invia attraverso la porta seriale 5 byte, di cui: 4 per la variabile intera e 1 per il carattere di 'a capo':

9a 10 00 00 0a

Da notare che 4250_{10}=0x109a ma i byte sono inviati dal meno significativo al più significativo

In questo caso è necessario specificare l'indirizzo della variabile da inviare[1], la dimensione del dato in byte (sizeof(**int**) vale 4), il numero di elementi della dimensione specificata da inviare (nel nostro caso 1), e il flusso di uscita che è stdout.

La funzione putc invece invia il singolo carattere '**\n**' sul flusso stdout.

[1] Vedi puntatori in C per approfondimento

LETTURA SERIALE DA MATLAB

Quando la scheda è collegata a una porta usb, essa è visualizzata come porta seriale. Su Windows in genere è indicata con il prefisso COM, su macOS e Linux con il prefisso dev/ttyS.

1) Collegare la scheda via usb al pc
2) Nella console di Matlab digitare serialportlist oppure seriallist per versioni precedenti alla R2019b
    a. Saranno visibili tutte le porte seriali disponibili, una di esse è la scheda STM32

La lettura della porta COM si esegue tramite la funzione fscanf che permette di leggere una sequenza di byte terminati dal carattere **'\n'** eventualmente eseguendo la conversione dei dati automaticamente tramite il secondo parametro.

Matlab mette a disposizione altre funzioni, come la fread.

Ad esempio, per leggere una sequenza di caratteri inviati sulla porta COM1:

```
s = serial('COM1');
set(s,'BaudRate',9600);
fopen(s);

out = fscanf(s);
disp(out);

fclose(s);
delete(s);
clear s;
```

La prima parte è il setup della connessione alla porta COM1 con baud rate a 9600.

La stringa terminata dal carattere **'\n'** è stampata su console con la funzione disp.

Infine la risorsa s viene rilasciata.

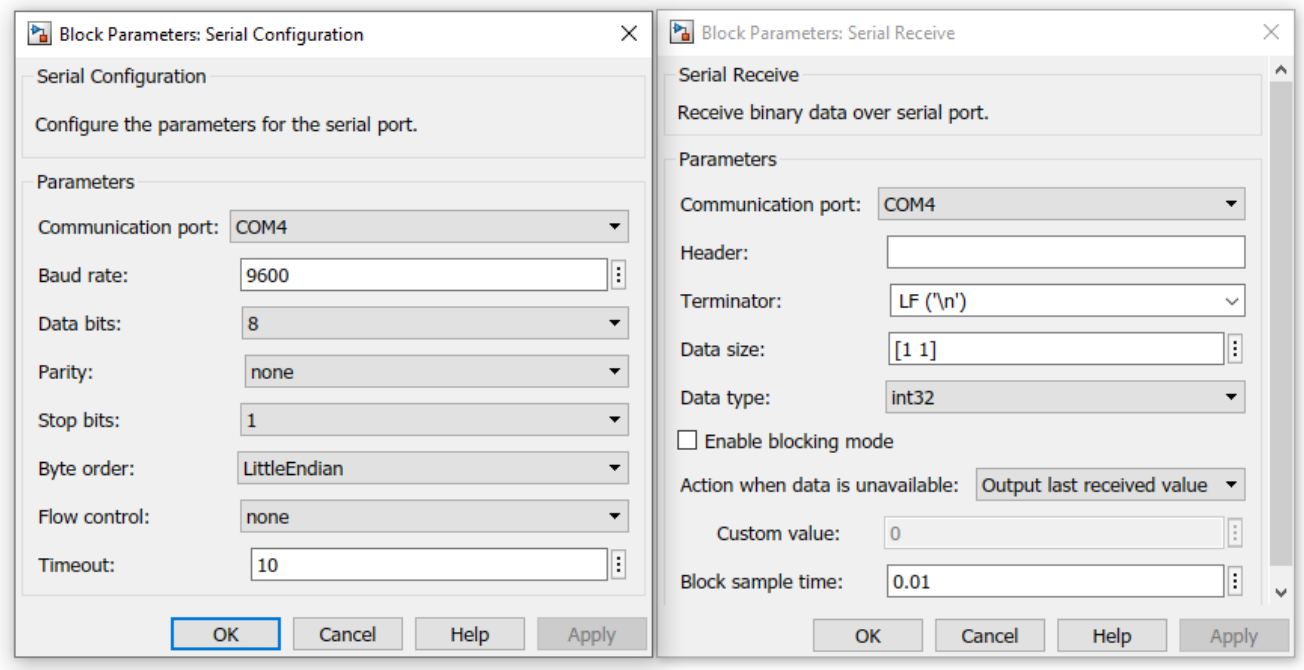
In questo caso la variabile out contiene una stringa di caratteri, se volessimo leggere dallo stream seriale un intero, sarà sufficiente convertirlo usando il secondo parametro:

```
out = fscanf(s, "%d");
```

LETTURA SERIALE DA SIMULINK

È possibile leggere lo stream seriale anche da Simulink. Per farlo è necessario l'add-on "Instrument Control Toolbox" che contiene al suo interno i blocchi Serial Configuration e Serial Receive.

1) Creare un nuovo modello Simulink vuoto
2) Scegliere dalla libreria il blocco Serial Configuration e il blocco Serial Receive, indicare la porta seriale della scheda e configurare i parametri nel modo seguente:



Con questa configurazione è possibile leggere i valori sullo stream dalla scheda inviati con la funzione fwrite.

Infatti il blocco simulink non esegue la conversione da stringa a intero come la fscanf. Se il Data type è int32 con terminator LF, il blocco legge dallo stream 4 byte seguiti da un **'\n'.**

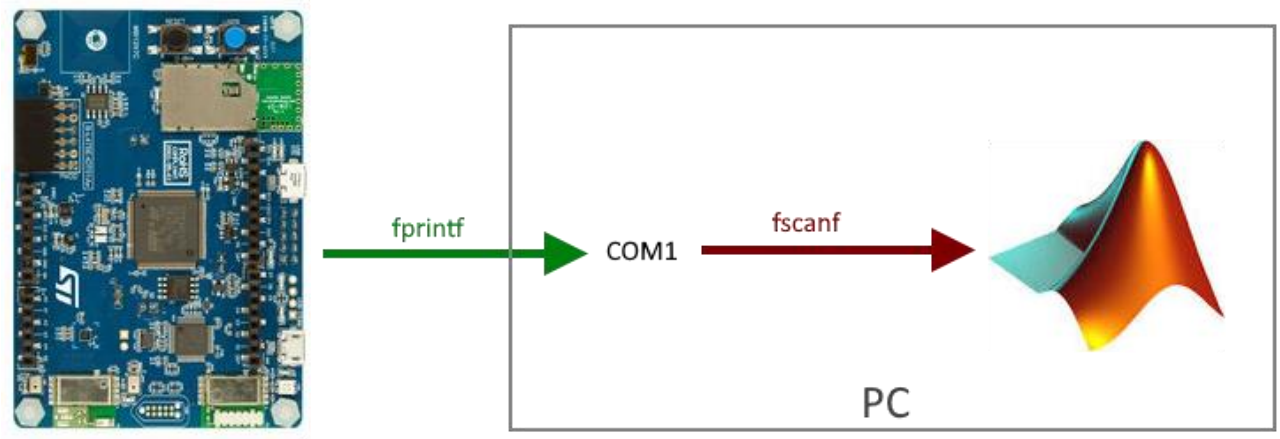
Confermando la configurazione potrebbe essere notificato un errore generico. Ciò accade se matlab/simulink è stato aperto prima di connettere la scheda. Soluzione: chiudere e riaprire matlab.

EMULAZIONE INVIO DA STM32

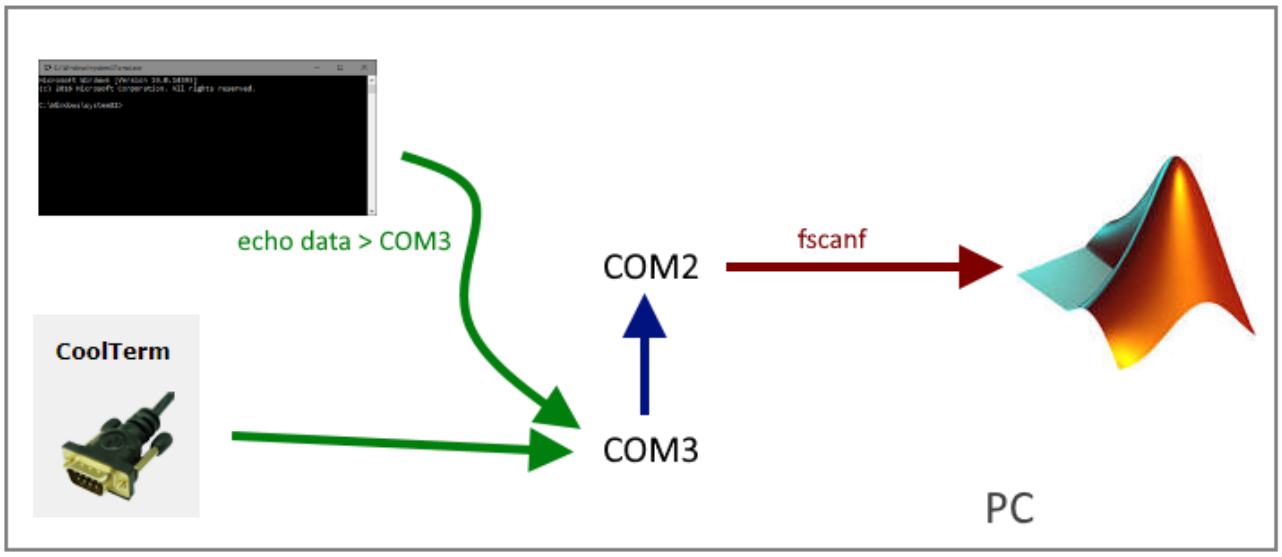
Se non è disponibile la scheda, è possibile utilizzare una porta seriale virtuale da usare come *stub* per sviluppare le applicazioni matlab e simulink.

Per fare ciò, è necessario un modo per creare due porte seriali virtuali collegate tra loro, di cui una sarà la porta di lettura di matlab e l'altra sarà la porta di scrittura della scheda STM32 virtuale.

Normalmente la trasmissione avviene così:



In assenza della scheda l'obiettivo è ottenere il seguente schema:

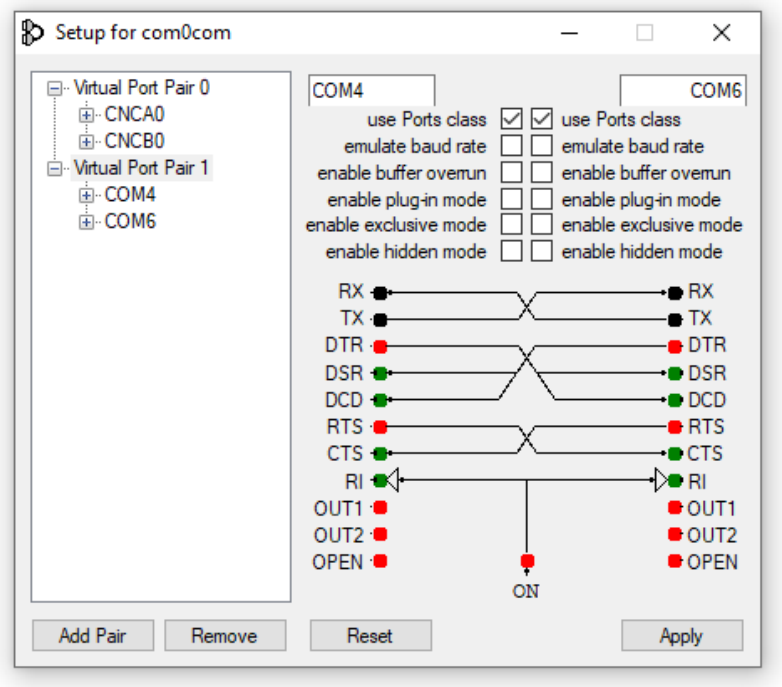


Creazione porte seriali

Windows

Scaricare e installare il programma Null-Modem emulator, conosciuto come com0com:
https://sourceforge.net/projects/com0com/

Eseguire il setup al termine dell'installazione. Automaticamente creerà una coppia di porte virtuali, nella figura successiva COM4 e COM6 che saranno visibili sia in gestione dispositivi sia tramite comando matlab serialportlist



MacOs/Linux

Da terminale è possibile creare le porte ttyS98 e ttyS99 tramite il comando socat. Questo comando mantiene aperte le porte fintanto che il terminale è aperto.

```
sudo socat PTY,link=/dev/ttyS98,raw,echo=0,crnl PTY,link=/dev/ttyS99,raw,echo=0,crnl
```

Per verificare che le porte siano state aperte si può verificare la loro presenza con

```
ls /dev
```

Inoltre è utile cambiare i permessi di lettura/scrittura delle porte aprendo un altro terminale:

```
sudo chmod 666 /dev/ttyS98; sudo chmod 666 /dev/ttyS99
```

Per verificare il funzionamento aprire due terminali distinti. Il primo resta in attesa di messaggi, il secondo li invia.

Terminale 1

```
$ sudo socat PTY,link=/dev/ttyS98,raw,echo=0,crnl PTY,link=/dev/ttyS99,raw,echo=0,crnl
```

Terminale 2

```
$ sudo chmod 666 /dev/ttyS99
$ tail -f /dev/ttyS99
it should work
```

Terminale 3

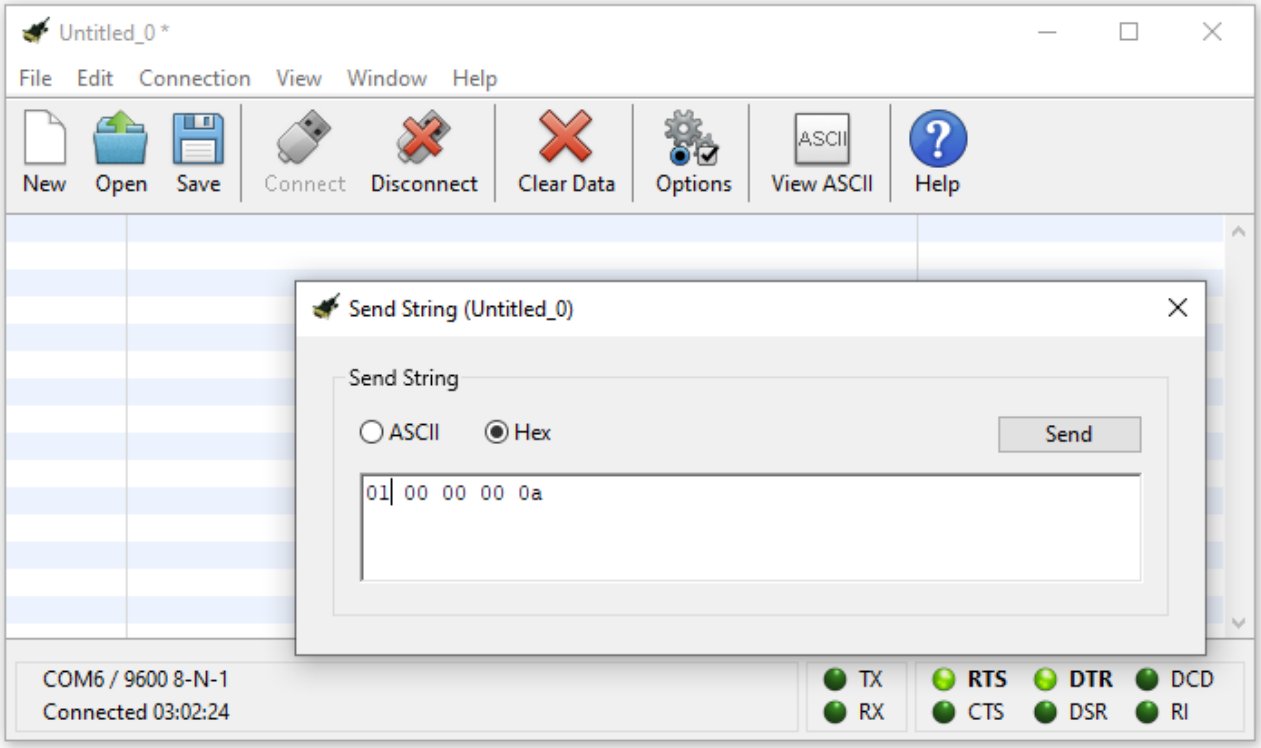
```
$ sudo chmod 666 /dev/ttyS98
$ echo "it should work" > /dev/ttyS98
$
```

Invio Dati sulle porte seriali

Tramite il programma CoolTerm http://freeware.the-meiers.org/ disponibile per Windows/macOs/Linux

Connettersi a una delle due porte e inviare dati tramite connection->send string.

È possibile inviare sia in versione ASCII, come farebbe la printf, oppure come Hex se si vuole simulare l'invio dei byte della fwrite.



In alternativa si può usare qualsiasi comando da terminale redirigendo l'output sulla porta scelta.

```
echo 10 > COM6
```

```
echo 10 > /dev/ttyS98
```