# Bluetooth

Note preliminari:

* Citazione dalla grande guida dei moduli HC-0X :

**"3. Notes. (1) HC-03 and HC-05’s command should end up with “\r\n”. It means when you finish programming, you should add terminator (“ENTER” or “0x0d 0x0a”) to the program. It’s different from HC-04 and HC-06 (They don’t need terminator)."** -cit

*http://cdn.instructables.com/ORIG/FKY/Z0UT/HX7OYY7I/FKYZ0UTHX7OYY7I.pdf*

* Secondo la guide di Arduino e instructables i pin del bluetooth vanno connessi così:

Pin del Micro per RX ———-> TX

Pin del Micro per TX ———-> RX

3.3V ———-> VCC

GND ———-> GND

Librerie necessarie:

#include <SoftwareSerial.h>

Dichiarazioni:

Pre-setup & post-libreria questo :

* SoftwareSerial nomedelsoftwareseriale(esempio:"bluetooth")(PIN RX,PIN TX);

Nel setup:

* Nomedelsoftwareseriale.begin(38400);

**"In modalità AT il modulo HC comunica a 38400 baud**." -cit

**"Gran parte di questo capitolo è preso da** [**questa guida su Instructables**](http://www.instructables.com/id/AT-command-mode-of-HC-05-Bluetooth-module/)**. Per cambiare configurazione al vostro modulo (ad es. cambiare velocità di trasmissione, nome, pin, ecc.) dovete entrare in modalità AT. L'HC-06 dovrebbe essere sempre in questa modalità, mentre l'HC-05 dev'essere acceso tenendo premuto il tasto "KEY" oppure mettendo a +3.3v il pin key sulla board. "** -cit

*http://www.raspibo.org/wiki/index.php?title=HC-05\_e\_HC-06#Differenze\_HC-05\_e\_HC-06*

Funzioni:

* Verifica della connessione :

#Questa funzione ritorna 1 se la software serial è attiva e sta "ascoltando" (streammando il flusso dati) mentre ritorna 0 se non sta "ascoltando" .

**int nomedelsoftwareseriale\_connection\_state()**

**{int a=0;**

**if (nomedelsoftwareseriale.isListeninig())**

**{a=1}**

**return a;**

**}**

* Flush del buffer sul software seriale :

#Questa funzione semplicemente scarta TUTTO quello che c'è nel buffer inserendolo nella variabile flush e sovrascrivendolo ogni volta. **Non c'è modo di invertire questa funzione**.

**void buffer\_flush()**

**{**

**while(nomedelsoftwareseriale.available()>0)**

**{**  
 **char flush = nomedelsoftwareseriale.read();**  
 **}**

**}**

* Trasmissione di un messaggio intero e stringa :

#Questa funzione trasmette sul software seriale inizializzato (qualsiasi dipendentemente dal nomedelsoftwareseriale ) una stringa o un intero passata/o alla funzione, il nome della variabile nella funzione è 'transimission'.

**void message\_string\_transmit (string transmission)**

**{**

**nomedelsoftwareseriale.write(transmission);**

**}**

Piccola variante della funzioni precedenti che permette di inviare variabili di interi.

**void message\_int\_transmit (int transmission)**

**{**

**nomedelsoftwareseriale.write(transmission);**

**}**

* Funzione per la restituzione della lunghezza del buffer del software seriale.

**int buffer\_lenght();**

**{int lenght=0;**

**x=nomedelsoftwareseriale.available();**

**return x;**

**}**

* Verifica dei messaggi :

#Questa funzione ritorna il contenuto del buffer del software seriale **se presente** altrimenti ritorna NULL, se il buffer supera il limite di 2048 elementi il buffer non viene letto e viene flushato poiché illeggibile.

Esiste anche readString ma questo metodo mi sembra più sicuro, non ottimale, **sicuro**.

**char serial\_buffer\_retrieve()**

**{int char\_ind=0, char\_ind2=0, x;**

**x=lenght\_buffer();**

**if (x>0)&&(x<2048)**

**#'x>0' significa che ci sono x dati nel buffer, svuota (buffer\_flush()) il buffer se ci sono più di 2048 elementi, altrimenti si va verso il buffer overflow.**

**{**

**char buffer\_retrieve[x];**

**#inizializzazione buffer\_retrieve[x](sotto)**

**for (char\_ind=0; char\_ind<x; char\_ind++)**

**{**

**buffer\_retrieve[char\_ind]=' ';**

**}**

**#scrive su buffer\_retrieve[x](sotto)**

**for (char\_ind2=0; char\_ind2<x; char\_ind2++)**

**{**

**char\_retrieve[char\_ind2] = nomedelsoftwareseriale.read();**

**}**

**return char\_retrieve[x];**

**}**

**buffer\_flush();**

**return null;**

**}**

* Funzione per l'identificazione (detecting) di messaggi singoli o multipli contenuti nel vettore di caratteri passato alla funzione.

Si utilizza la variabile char end\_message per stabilire una combinazione di caratteri all'interno del messaggio con il significato di 'fine messaggio'.

Si utilizza la variabile int lenght per sapere quanto è lungo il vettore di caratteri essa sarà calcolabile grazie alla funzione buffer\_lenght(), la funzione appena esplicitata andrà posizionata con criterio prima della funzione serial\_buffer\_retrieve;

La funzione ritorna il numero di messaggi (il numero di volte che viene trovato nel vettore 'detecting' il vettore char 'end\_message') sotto forma di intero.

*Esempio :*

*Vettore end\_message = "\*r"*

*Vettore detecting = "ciao \*r come \*r stai? \*r"*

*in questo esempio verrano detectati 3 messaggi (3 volte è presente "\*r").*

**int message\_detect(char detecting[], char end\_message[|2 circa?|], int lenght)**

**{int number\_of\_messages=0, utility\_ind=0;**

**for( utility\_ind=0; utility\_ind<lenght; utility\_ind++)**

**{**

**if( detecting[utility\_ind] == end\_message[1])**

**if(( detecting[utility\_ind+1] == end\_message[2])&&(utility\_ind!=lenght-1))**

**#la riga sovrastante serve ad evitare che venga letto un carattere in più rispetto alla lunghezza del vettore stesso. (esempio utility\_ind==19 allora utility\_ind+1==20 QUESTO NON DEVE ACCADERE) questo problema è anche risolvibile modificando il for ma una soluzione vale l'altra in questo caso.**

**number\_of\_messages++;**

**}**

**return number\_of\_messages;**

**}**