Gestione di Arduino da remoto

Per gestire una scheda Arduino da remoto, per esempio da PC tramite una linea seriale, sono necessarie tre cose:

- un insieme definito di comandi e risposte da scambiarsi tra il PC, che funziona da master (*client*) e la scheda slave Arduino (*server*);
- un programma (*sketch*) su Arduino, che attende i comandi in arrivo sulla linea seriale e li esegue;
- un software sul PC (una finestra *Hyperterminal* o un programma in *Visual Basic* o *C++*) che invia i comandi e visualizza le risposte.

Per facilitare l'elaborazione delle stringhe di comando si può utilizzare la funzionalità *String* di Arduino, mentre la finestra di dialogo sul PC può essere la stessa del monitor seriale a disposizione.

Studio e collaudo della funzionalità *String* di Arduino

Una stringa è un array di elementi char, terminato con il carattere **null** (\0).

Il carattere null ha valore binario 0, notevolmente diverso rispetto al carattere ASCII '0', che ha valore decimale 48 (30_H). Per manipolare le stringhe (copiarle, concatenarle, calcolarne il numero dei caratteri) Arduino, oltre alle funzioni standard, mette a disposizione la funzionalità **String** (con il carattere iniziale 'S' maiuscolo), non compatibile con il linguaggio C standard.

Per comprenderne le caratteristiche, si esegua il programma che segue e si osservi quanto emesso nella finestra di monitor seriale.

```
//String_basic
String text1 = "prima stringa";
String text2 = " con aggiunta";
String text3; //da usare successivamente

void setup()
{
Serial.begin(9600);
Serial.print(" text1= ");
Serial.println( text1);
Serial.print(text1.length());
Serial.println(" caratteri significativi");
Serial.print("anche text2 contiene ");
Serial.print(text2.length());
Serial.println(" caratteri");
text1.concat(text2);
Serial.println("concatenandole, text1 diventa: ");
Serial.println(text1);
}
void loop()
{}
```

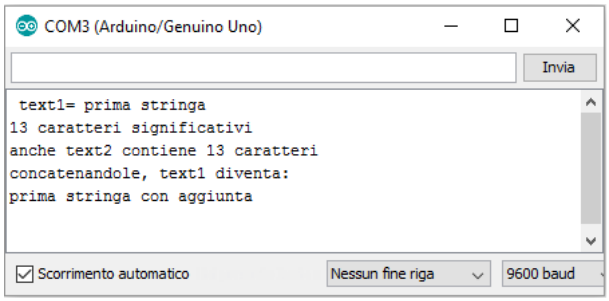

Fig. 1. *Collaudo della funzionalità String.*

Definendo la stringa text1 di tipo String è possibile ottenerne la lunghezza con la:

```
text1.length()
```

e concatenarla con la variabile String text2 mediante la:

```
text1.concat(text2)
```

Per combinare stringhe si può anche utilizzare l'operatore +. Per verificarlo basta aggiungere in coda al codice precedente le due righe seguenti.

```
text3 = text1 + " e altro ancora";
Serial.println(text3);
```

Per elaborare con la funzione String gli array di caratteri definiti in C, bisogna prima convertirli in oggetti String, come indicato di seguito.

```
charoldString[] = "Array di caratteri da convertire in un oggetto String";
StringnewsString = oldString;
```

Gestione da monitor seriale

Sulla scheda Arduino, la presa USB confluisce in un convertitore FT232, le cui uscite TXD e RXD sono incrociate con i segnali della porta seriale del microprocessore e terminano rispettivamente sui pin 0 (RX) e 1 (TX) del connettore Digital.

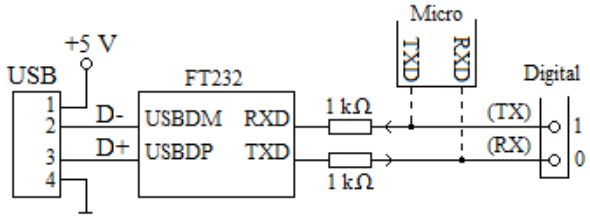

Fig. 2. *Linee seriali interne a Arduino.*

È possibile quindi governare le uscite di Arduino o leggere lo stato degli ingressi tramite monitor seriale, concordando i codici di attivazione da inviare e predisponendo uno sketch che interpreti il contenuto dei messaggi ricevuti dalla seriale.

Lo sketch che segue, per esempio, accende e spegne il LED presente su Arduino, connesso con il pin 13, inviando da monitor rispettivamente i caratteri A (accendi) e S (spegni).

```
/* Monitor_LED */
String comando ="";

void setup()
{
Serial.begin(9600);
pinMode(13,OUTPUT);
}
void loop()
{
while(Serial.available())
comando+=char(Serial.read());

 if(comando!=""){
  if(comando == "A"){
  digitalWrite(13, HIGH);
  Serial.println("Led ON");
  }
  if(comando == "S"){
  digitalWrite(13, LOW);
  Serial.println("Led OFF");
  }
 comando="";
 }
}
```

Ogni volta che il programma acquisisce dalla porta seriale un comando valido, aziona di conseguenza l'uscita 13 e restituisce un messaggio di conferma. Terminata l'elaborazione, la stringa di comando viene svuotata, così da renderla disponibile per una nuova acquisizione.

Possibile continuazione

Predisporre uno sketch che gestisca i messaggi a dimensione fissa riportati in tabella, supponendo che i pin da 2 a 7 siano stati predisposti come ingressi con pull-up e da 8 a 13 come uscite.

Protocollo di gestione da remoto		
Comando	**Azione**	**Valori**
Hnn	Porta alto il pin nn	H08 – H13
Lnn	Porta basso il pin nn	L08 – L13
Rnn	Legge lo stato H/L del pin nn	R02 – R13
An	Legge il livello dell'ingresso analogico n	A0 – A5

Per la soluzione, è utile considerare le due funzioni C standard di manipolazione delle stringhe evidenziate di seguito.

```
char *strstr (char *string1, char *string2)
```

Ricerca la presenza della stringa (dello spezzone) string2 all'interno di string1 e in caso affermativo ritorna un valore diverso da zero (in pratica ritorna il puntatore alla prima lettera corrispondente); per esempio:

```
if (strstr (str,"pin4=on")) digitalWrite(4, HIGH);
```

porta alto il pin 4 se nella stringa str è presente la scritta "pin4=on";

```
int sprintf (char * buf, char *format, arg-list)
```

Compone la stringa buf con le variabili listate al termine, utilizzando il format indicato; per esempio:

```
sprintf(str,"%d%dore",h/10, h%10);
```

compone str con la scritta indicante le decine e le unità delle ore in decimale, seguite dal testo "ore". Alcuni codici di formato sono: %c (singolo carattere), %d (decimale intero), %f (decimale con virgola), %x (esadecimale), %s (stringa di caratteri), ecc., con diverse varianti sul numero di cifre e di decimali da riportare.