

Mensile di progettazione elettronica, attualità scientifica, novità tecnologiche.

€ 6,00

Elettronica In

www.elettronica.in.it

oltre l'elettronica

Anno XIX - n. 182
Dicembre 2013
Gennaio 2014



REALIZZA I
CIRCUITI STAMPATI
CON LA 3DRAG

Wi-Fi Shield
per **Arduino**



Plotter
per superfici
sferiche



- **OROLed**, un orologio a LED a lettura analogica
- **UDOO** la scheda italiana "da corsa"
- **myKIT**, misura di temperatura con sonda RTD
- **LEWE**, bracciale biometrico
- **Linux Embedded** con pcDuino
- **Corso mbed**



Dotiamo Arduino di uno shield e una libreria per farlo comunicare con il mondo esterno tramite WiFi.

WI-FI E INTERNET PER ARDUINO

di DANIELE DENARO

La scheda Arduino ha rappresentato una vera rivoluzione in campo elettronico dimostrandosi particolarmente adatta ai non esperti ed allo stesso tempo anche molto versatile, tanto da riuscire a gestire moltissime applicazioni in svariati ambiti. L'abbiamo vista, ad esempio, affrontare compiti impegnativi come la connessione con il mondo esterno, realizzata ora con le semplici interfacce di base (seriale, I²C-Bus), ora con quella che definiamo la connettività vera e propria, ossia le interfacce wireless e la rete Ethernet. In entrambi i casi i limiti di Arduino si manifestano palesemente, perché la scheda del team di Banzi, come si sa, conta su un hardware minimale, mentre

i collegamenti con le reti ed ancor più la connettività Internet, richiedono una gestione asincrona e sostanzialmente multi-task dei protocolli di comunicazione. Per questo motivo, il collegamento di Arduino con Internet è sempre stato un po' difficoltoso, tanto da spingere i suoi creatori a implementare un sistema ibrido come la YÚN, dove il processore Atheros ha lo scopo principale di gestire le comunicazioni. YÚN non è comunque l'unica soluzione, infatti, come dimostreremo in queste pagine, Arduino può disporre della connettività Internet attraverso uno shield equipaggiato con un gestore dello stack TCP/IP, in modo da liberare l'ATmega da alcune incombenze

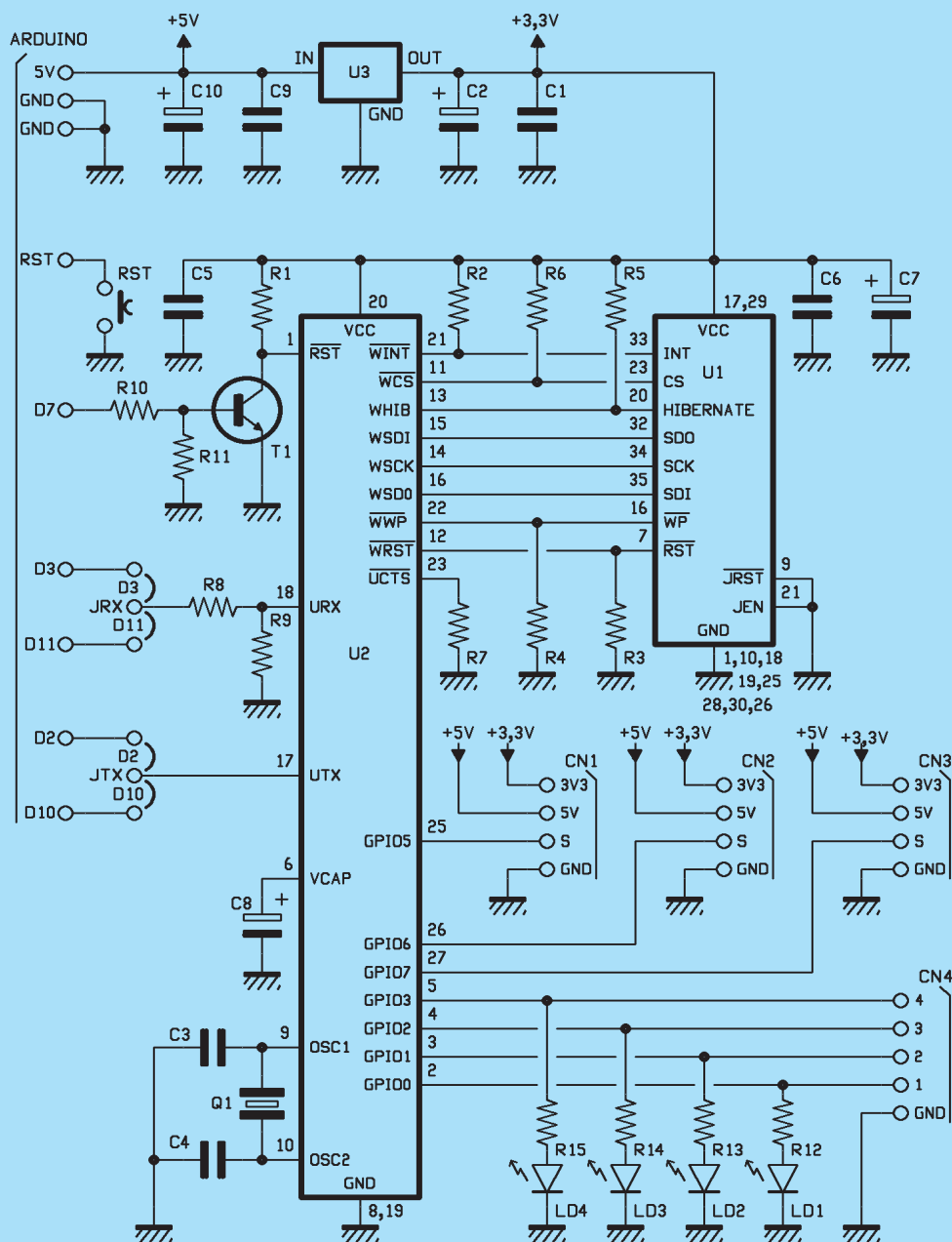
di base. Ma anche con il gestore, bisogna comunque superare diverse difficoltà nella gestione dei collegamenti Internet, sia con semplici socket2 (vedi più avanti), sia con il protocollo Http (quello usato dal mondo Web). Per questo motivo, è fondamentale corredare la scheda di una libreria che, oltre a dialogare con il gestore del

protocollo TCP/IP, semplifichi al massimo la vita di chi vuole programmare Arduino e vuole farlo dialogare con altri computer tramite Internet.

Iniziamo con la descrizione dell'hardware del nuovo shield WiFi e successivamente descriviamo la libreria di corredo ed il suo uso.

LO SHIELD WIFI

Se ricordate, già qualche tempo fa (precisamente, nel fascicolo n° 160) abbiamo pubblicato il progetto di uno shield per dotare Arduino della connettività WiFi; quello qui descritto è una versione migliorata, in quanto, rispetto al precedente, basato su un modulo di Microchip che ora

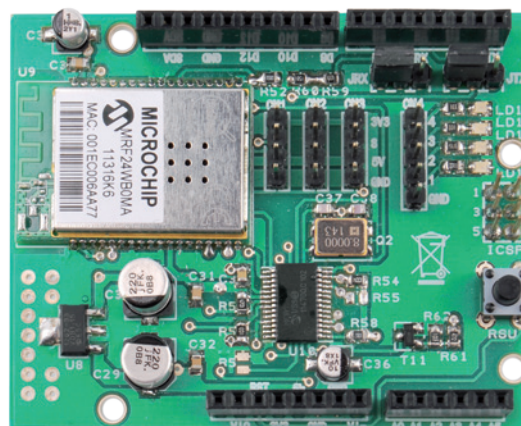
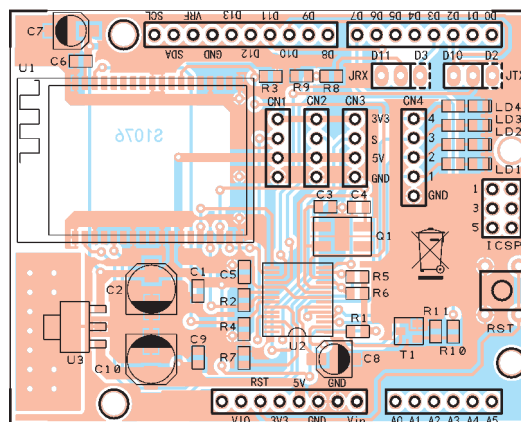


Elenco Componenti:

R1, R4, R11: 10 kohm (0805)
 R2, R10: 4,7 kohm (0805)
 R3: 100 kohm (0805)
 R5: 1 Mohm (0805)
 R6: -
 R7: -
 R8: 1 kohm (0805)
 R9: 1,5 kohm (0805)
 R12÷R15: 330 ohm (0805)
 C1, C5, C6, C9: 100 nF multistrato (0805)
 C2, C10: 220 µF 6,3 VL elettrolitico (D)
 C3, C4: 22 pF ceramico (0805)
 C7, C8: 10 µF 35 VL elettrolitico (B)
 U1: MRF24WB0MA/RM
 U2: MCW1001A
 U3: TC1262-3.3 (SOT-223)
 T1: BC817
 Q1: Quarzo 8 MHz (HCX-7SB) > RS: 675-4703
 RST: Microswitch
 LD1÷LD4: LED verde (0805)

Varie:

- Strip maschio 3 vie (2 pz.)
- Strip maschio 4 vie (3 pz.)
- Strip maschio 5 vie (1 pz.)
- Strip maschio/Femmina 6 vie (1 pz.)
- Strip maschio/Femmina 8 vie (2 pz.)
- Strip maschio/Femmina 10 vie (1 pz.)
- Strip maschio/Femmina 3 vie (2 pz.)
- Jumper (2 pz.)
- Circuito stampato



è uscito di produzione, dispone dell'hardware che permette di gestire da sé il protocollo TCP/IP. Più esattamente, la differenza sostanziale tra il vecchio shield e questo nuovo è che nel precedente era Arduino (tramite un'apposita libreria) a governare il modulo WiFi e gestire il protocollo TCP/IP, mentre in questo nuovo circuito abbiamo inserito un processore di interfaccia che si occupa della gestione del protocollo TCP/IP, scaricando così Arduino da un compito che, dall'esperienza del primo shield, è apparso troppo impegnativo. Ma una scheda di rete diventa veramente utile se è pilotata da un software che semplifica il più possibile il suo utilizzo; per

questo motivo abbiamo corredato la scheda di una libreria che pensiamo finalizzata agli utilizzi di Arduino, con un grado di semplificazione superiore a quello per un hardware più complesso (anche se più flessibile) come quello della YÚN. Venendo alla descrizione dell'hardware, possiamo notare che lo shield (che ha le stesse dimensioni della scheda Arduino Uno) si basa essenzialmente su un modulo WiFi MRF24WB0MA ed un processore MCW1001A, entrambi Microchip. L'alimentazione è a 3,3V (ottenuti mediante un riduttore di tensione alimentato dalla linea 5V di Arduino) ma il processore sopporta anche livelli TTL.

Il modulo radio ha già l'antenna integrata.

Il processore governa il modulo WiFi tramite il bus SPI (linee SDO, SDI, SCK) ed altre linee di controllo come CS, INT, RST, HIBERNATE, WP; dal lato host (cioè Arduino) il processore è collegato con un'interfaccia seriale a due fili in logica TTL (RX,TX). Questo ci ha costretto ad utilizzare una porta seriale simulata da software con una versione personalizzata della libreria SoftwareSerial. La personalizzazione è stata necessaria a causa della modalità con due bit di stop, ma anche per aggiustare la velocità ottimale e rendere trasparente l'utilizzo della libreria. Infatti questa versione della Software-


```
#define RXPIN 3
#define TXPIN 2
```

stack TCP/IP, gestendo il protocollo di base attraverso l'utilizzo della sua RAM interna per i buffer di trasmissione e ricezione, nonché per la memorizzazione dei vari parametri. Va però detto che, non disponendo di memoria non volatile interna, il processore ad ogni riavvio va riconfigurato. Il protocollo di comunicazione tra l'host ed il processore e le sue funzioni sono ben descritti nel datasheet del componente, che trovate alla pagina web: www.microchip.com/wwwproducts/Devices.aspx?dDocName=en555844. In ogni caso, la gestione di basso livello è demandata alla libreria che fornisce un insieme di funzioni per aprire socket o gestire il protocollo HTTP.

una figura di *server* (e listener, ovvero ascoltatore) ed una figura di *chiamante* (client), altrimenti sarebbe impossibile sincronizzare il colloquio.

La libreria copre questi due tipi di funzionalità: ad esempio può essere utilizzata quando vogliamo dare ad Arduino il compito di server, in modo che reagisca e risponda a messaggi di un altro computer. Oppure può essere utilizzata per dare ad Arduino la possibilità di collegarsi di sua iniziativa ad un computer server per spedirgli, per esempio, misure effettuate da sensori.

Il collegamento si stabilisce in TCP/IP tramite delle entità chiamate *socket*, le quali corrispondono al concetto di “presa” cui collegare un cavo di comunicazione. Esiste quindi un socket per ciascuno dei due computer che devono dialogare tra loro. Un socket si crea nell’ambito di un numero di porta. I numeri di porta sono dei sub-indirizzi applicativi all’interno di un indirizzo IP. I numeri inferiori a 1.000 sono riservati ai servizi pubblicamente riconosciuti quali Mail, Ftp, Web (porta 80), e così via.

Bene, giunti a questo punto possiamo descrivere la libreria per Arduino che implementerà la gestione della connessione in TCP/IP. La libreria si può scaricare dal nostro sito www.elettronica.in.it insieme agli altri file del progetto. Ma procediamo con ordine: la MwiFi va posizionata (dopo la decompressione) nella directory "libraries" dell'IDE di Arduino, come le altre librerie; va utilizzata includendo negli sketch il file <MwiFi.h>.

A questo punto va istanziata come oggetto e può essere utilizzata; la prima funzione da richiamare è **begin()**, che serve a

inizializzare la scheda:

```
#include <WiFi.h>
WiFi WiFi;
void setup()
{
  WiFi.begin();
  :
```

Lo start-up della scheda determina l'accensione del primo LED. Proviamo, ora, a collegare Arduino in rete utilizzando un access-point, che potrebbe essere quello del router WiFi casalingo, per esempio assegnando il nome (SSID) "D-Link-casa":

```
WiFi.ConnSetOpen("D-Link-casa");
```

valido nel caso la rete fosse non protetta, oppure:

```
WiFi.ConnSetWPA("D-Link-casa", "miarete");
```

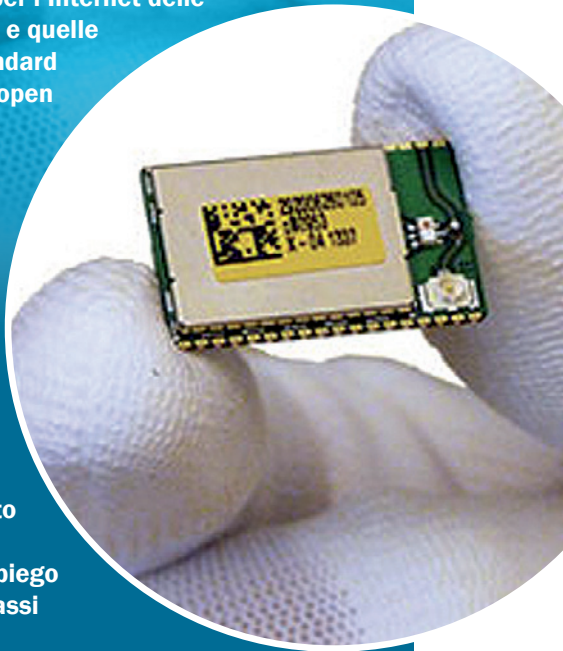
nel caso la rete avesse protezione WPA con password "miarete". Le precedenti funzioni servono a predisporre la scheda alla connessione; ma la vera e propria connessione si ottiene chiamando la funzione:

```
WiFi.Connect();
```

Se la connessione va a buon fine, si accende il secondo LED. Fate però attenzione, perché nel caso di rete protetta, la connessione può richiedere anche più di mezzo minuto, in quanto il controller della scheda deve codificare la chiave tramite la password. L'attuale versione della libreria prevede un reset automatico di Arduino nel caso la connessione venga persa; il reset automatico si verifica anche se vengono rilevati errori dal controller. In questo modo, un eventuale blocco è scongiurato ed il sistema può essere "unattended".

WiFi 5 GHz + Bluetooth

ConnectBlue presenta OWL355, un modulo WLAN capace di funzionare sia nella classica banda a 2,4 GHz, sia nella nuova 5 GHz, oltre che di interfacciarsi con tutti gli standard Bluetooth. Progettato per l'Internet delle Cose, le applicazioni diomedicali e quelle industriali, supera gli attuali standard sia IPC che AQL. Grazie al driver open source Linux, il modulo consente di implementare con facilità una connessione wireless in un qualsiasi dispositivo elettronico fornendo hardware modulare omologato, certificazione EMC, qualificazione Bluetooth, un'ampia gamma di antenne. Tra le altre caratteristiche, da segnalare la tecnologia Castellation Package che utilizza dei rilievi di metallo ai lati del modulo che rendono molto più agevole la saldatura (anche manualmente) consentendo l'impiego del modulo in applicazioni con bassi volumi di produzione.



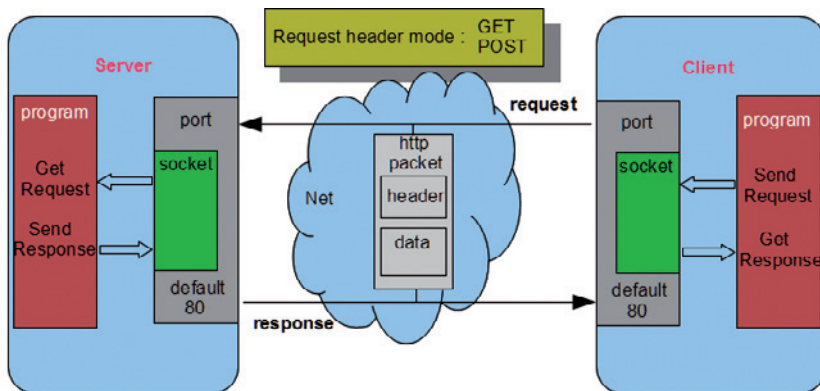
Per comodità sono state aggiunte anche funzioni dirette di connessione, che preparano la connessione e la eseguono in un unico step. Inoltre è stata aggiunta la possibilità di generare la chiave numerica dalla password, in modo da utilizzare successivamente questa al posto della password; infatti l'accesso con password, prevedendo ogni volta l'elaborazione della chiave numerica, può richiedere anche un minuto, mentre l'accesso con chiave è velocissimo. Ad Arduino viene assegnato, dal router, un indirizzo IP dinamico, perché il comportamento predefinito della scheda è questo; ma volendo si può imporre un indirizzo fisso. L'indirizzo assegnato può essere richiesto da una funzione della libreria. A questo punto scegliamo di far

funzionare Arduino da server: per prima cosa chiamiamo la funzione **openServerTCP()**, che crea l'ascoltatore su una certa porta (per esempio 5000), e poi mettiamo in loop la ricezione di una eventuale richiesta di link:

```
int ssocket=WiFi.openServerTCP(5000);
void loop()
{
  int csocket=WiFi.pollingAccept(ssocket);
  :
```

Le variabili intere *ssocket* e *csocket* sono dei riferimenti (handle) rispettivamente al server e al socket di collegamento con l'eventuale computer che vuole effettuare il link. La funzione **pollingAccept()** ritorna un numero minore di 255 se il collegamento

Fig. 2 - Comunicazione HTTP.



è stato richiesto, o 255 se non c'è alcuna richiesta di collegamento in arrivo.

Nel caso il collegamento sia stato stabilito potremo spedire o ricevere messaggi facendo riferimento a questo *csocket*. Per esempio per ricevere un record, cioè una stringa terminante con un line-feed possiamo utilizzare la funzione:

```
char *line=WIFI.  
readDataLn(csocket);
```

Ci verrà restituita una "null-terminated string" ma senza il line-feed. In questo caso, la libreria utilizza un buffer predefinito di 81 caratteri (ma la sua lunghezza può essere modificata nel suo define) perciò non abbiamo bisogno di fornirli noi. Sono comunque previste altre possibilità. Se invece vogliamo rispondere possiamo utilizzare la funzione:

```
WIFI.writeDataLn(csocket,answer);
```

La variabile che abbiamo chiamato *answer* corrisponde ad un buffer di char, però deve essere una "null terminated string". Utilizzare una "null-terminated string" vuol dire che non dobbiamo fornire la lunghezza dei caratteri utili nell'array, perché la funzione la calcola automaticamente, potendo basarsi sul carattere terminale nullo. La maggior parte delle funzioni in C gestisce

e produce questo tipo di stringa, da non confondere con l'oggetto String presente anche nel *language reference* di Arduino.

In questo semplice modo abbiamo stabilito e utilizzato un collegamento WiFi con un computer remoto. Nella libreria, tra gli esempi riportati, c'è un esempio di server chiamato *CommandServer*, che permette di comandare Arduino utilizzando un programma tipo *telnet* sul computer remoto. Per semplificare i test è stato aggiunto un programma Java che opera come *telnet*.

Se invece volessimo far agire Arduino come un "client" che si collega ad un server, la situazione sarebbe ancora più semplice, perché dovremmo creare solo un socket di collegamento, così:

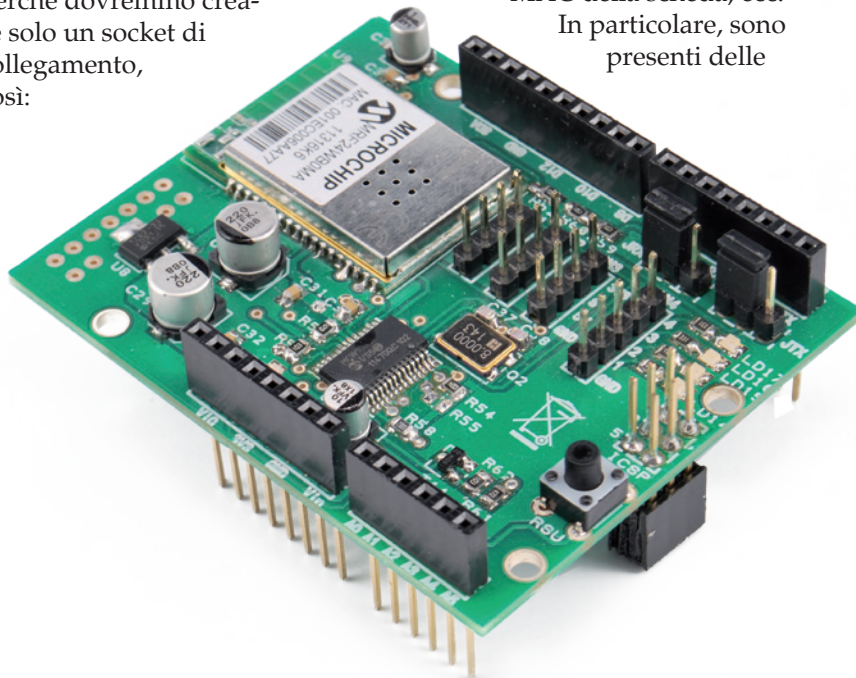
```
int csocket=WIFI.openSocketTCP("192.168.1.2",5000);
```

E se *csocket* è valida (minore di 255), vuol dire che il collegamento, con il computer all'indirizzo 192.168.1.2 sulla porta 5000, è stato stabilito.

A questo punto possiamo utilizzare le funzioni di scrittura e lettura viste precedentemente. Tra gli esempi ce n'è uno (**SendData**) che si collega ad un server remoto per spedirgli ad intervalli regolari letture di sensori; in questo caso è necessario un programma server installato sul computer remoto. Per facilitare i test, insieme alla libreria è stato fornito un programma Java che riceve i dati e li scarica su un file aggiungendo un time-stamp (per ovviare alla mancanza di un RTC, Real Time Clock, su Arduino). Oltre a queste funzionalità di base, nella libreria sono presenti tutte le funzioni necessarie per definire diversi parametri quali l'indirizzo di mascheramento della rete (255.255.0.0 è il predefinito), l'eventuale indirizzo di

Gateway, la lettura del codice MAC della scheda, ecc.

In particolare, sono presenti delle



funzioni per rilevare access-point presenti e visibili nell'ambiente. Ad esempio, per rilevare tutte le reti presenti si usa l'istruzione:

```
int nn=WIFI.scanNets();
```

La variabile intera *nn* conterrà il numero di reti rilevate. Invece la funzione:

```
char* net=WIFI.getNetScanned(i);
```

restituirà le caratteristiche (sotto forma di record) della "i-esima" rete rilevata. Infine, per comodità, è stata prevista una funzione che restituisce il nome della rete non protetta che fornisce il segnale più potente.

Rimandiamo alla documentazione della libreria la descrizione di tutte le funzionalità disponibili. La libreria contiene un help (in inglese) ed è documentata nei file di codice (in particolare i file .h). Ma la libreria non si esaurisce nella sola gestione di connessione e socket; infatti in essa è inclusa una classe derivata (e quindi specializzata) che gestisce il protocollo HTTP. Quest'ultimo è un protocollo che prevede sempre una richiesta ed una risposta; sia la richiesta che la risposta fanno viaggiare in rete pacchetti formati da alcune intestazioni

Listato 1

```
WEBRES rs[8]=
{
  {"index",pindex},
  {"Analog",panalog},
  {"RDigital",rdigital},
  {"Wdigital",wdigital},
  {"wdig",wdig},
  {"Pwm",pwpmpage},
  {"PwmSet",pwmset},
  {"End",sessend}
};

void loop()
{
  WIFI.getRequest(csocket,8,rs);
  :
```

Listato 2

```
prog_char pagerdigital[] PROGMEM=
:
"<tr>"
"<td><div align='center'>@</div></td>"
"<td><div align='center'>@</div></td>"
"<td><div align='center'>@</div></td></tr>"
:

void rdigital(char *query)
{
  char *val[3];
  if(digitalRead(4)) val[0]=ON;else val[0]=OFF;
  if(digitalRead(5)) val[1]=ON;else val[1]=OFF;
  if(digitalRead(12))val[2]=ON;else val[2]=OFF;
  WIFI.sendDynResponse(csocket,pagerdigital,3,val);
}
```

(header) e dati veri e propri (come le pagine html, immagini, video o anche semplice testo). Per scaricare l'utente da tutta questa problematica, la libreria per HTTP si preoccupa di formare questi pacchetti utilizzando, inoltre, in modo spinto la modalità PROGMEM cioè la possibilità di porre in area Flash di programma le costanti ed in particolare i testi. Facciamo notare che il protocollo HTTP è un protocollo testuale: usa solo caratteri. L'uso della memoria Flash per i testi permette di risparmiare spazio nell'esigua RAM di Arduino.

LA LIBRERIA HTTP

Essendo una classe derivata di *MWiFi*, la libreria HTTP eredita tutte le funzioni di *MWiFi*; tuttavia se si vogliono utilizzare le nuove funzioni bisogna includere il file *HTTPlib.h* (al posto di *MWiFi.h*) ed instanziare un oggetto HTTP:

```
#include <HTTPlib.h>
HTTP WIFI;
```

Riguardo alla connessione con un access-point e alla gestione dei socket, tutto rimane come prima (forse ora sceglieremo la porta 80); ma anche questa volta dobbiamo decidere se far fare ad Arduino le veci di un server (questa volta Web Server) o quelle di un client che accede ad un Application Web Server (come Tomcat,

GlassFish, Jboss, PHP, ecc.). Supponiamo di voler realizzare un Web Server da interrogare con un qualunque browser: per far funzionare Arduino come Web Server dobbiamo predisporre le risorse che esso può mettere a disposizione, cioè le pagine html di risposta. Queste pagine saranno memorizzate in aree PROGMEM per i motivi detti sopra. Per esempio:

```
prog_char pageindex[] PROGMEM=
"<html><head>"
"<title>Index</title>"
:
```

A questo punto si tratta di collegare questi buffer in memoria con i nomi delle risorse da richiamare tramite browser. I nomi delle risorse sono la parte di path locale della URL (o URI), ovvero, in sostanza, il nome del file in una URL completa (Es.: *http://www.mio.it/miapag.html*). In questo ambito minimale, le risorse da richiamare saranno identificate dal solo nome senza estensione. Per cui si tratta di associare una pagina memorizzata con il suo nome Internet corrispondente (per esempio *"/indice"*). In realtà questa pagina non si spedisce da sola e quindi bisogna collegare il nome della risorsa con una funzione che si occuperà di spedirla. Per rendere il meccanismo più automatico possibile, è stata predisposta una struttura o

Listato 3

```
void pwmset(char *query)
{
    char *pwmval;
    pwmval=WIFI.getParameter(query,strlen(query),"PWM10");
    if (pwmval!=NULL)
    {int pv;sscanf(pwmval,"%d",&pv);
      analogWrite(10,pv);d10=pv;}
    pwmval=WIFI.getParameter(query,strlen(query),"PWM11");
    if (pwmval!=NULL)
    {int pv;sscanf(pwmval,"%d",&pv);
      analogWrite(11,pv);d11=pv;}
    pwwpage(query);
}
```

più esattamente un *typedef* chiamato WEBRES; questa struttura è formata dall'accoppiata di due campi: il nome della risorsa ed il nome della funzione (che in C corrisponde ad un indirizzo). Si tratta, quindi, di formare tante coppie nome-funzione da passare alla funzione **getRequest()**, che si occuperà di lanciare la funzione corretta (call-back function), o spedire un messaggio standard "Not Found" di risposta, nel caso il nome non combaci con alcuno di quelli predisposti.

Nel **Listato 1** è presentato l'esempio della costruzione di un array di 8 strutture WEBRES ed il suo inserimento nella chiamata alla funzione **getRequest()**. Ogni secondo campo delle strutture corrisponde alla call-back function che **getRequest()** lancerà. La call-back function si dovrà preoccupare di spedire il buffer corrispondente alla pagina scelta ed il suo prototipo prevede che sia di tipo void (cioè che ritorni nulla) ed abbia un solo argomento; in pratica, un puntatore ad una null-terminated string fornita dal chiamante (si veda più avanti):

```
void pindex(char *query)
{
    WIFI.sendResponse(csocket,pageindex);
}
```

Riassumendo: ponendo **getRequest()** nel loop, essa si occuperà di tutta la gestione della richiesta

e rileverà la modalità utilizzata GET o POST, comportandosi di conseguenza (i dati sono contenuti in modo diverso) e lancerà la funzione corrispondente alla richiesta (o un messaggio "Not Found"). La call-back function **pindex()** sopra descritta, però non fa altro che spedire in risposta una pagina html statica, cioè definita in modo fisso.

Arduino Web Server definito così non è molto utile, perché si presuppone di poterlo utilizzare per leggere valori forniti da sensori (**Fig. 4**) o attivare uscite (**Fig. 5**); per ottenere ciò, la pagina html di risposta deve essere costruita sul momento, inserendo i valori che si vogliono leggere. Deve, insomma, essere una pagina dinamica. Ma sarebbe troppo dispendioso costruire, all'interno delle call-back function, la pagina tutta intera; per semplificare il compito si è previsto di definire la pagina "una tantum" come una pagina statica, potendo però inserire al suo interno dei tag (etichette segna-posto) nella posizione che si vuole completare al momento. Per questo scopo esiste, in alternativa a **sendResponse()**, la funzione **sendDynResponse()** che si preoccupa di rintracciare e sostituire i tag mentre spedisce la pagina. La sostituzione avviene sequenzialmente andando a scorrere un array di stringhe predisposto sul momento; quindi il primo tag incontrato viene sostituito con la prima stringa dell'array e così via. Il

tag utilizzato è il carattere @; ne va usato uno solo a prescindere dalla lunghezza della stringa che lo dovrà sostituire. Nell'esempio che trovate nel **Listato 2**, tre tag saranno sostituiti da altrettante stringhe che rappresentano i valori di tre input digitali. Alla funzione **sendDynResponse()** vanno passati l'array delle stringhe e la relativa dimensione. Per far sì che Arduino agisca in seguito

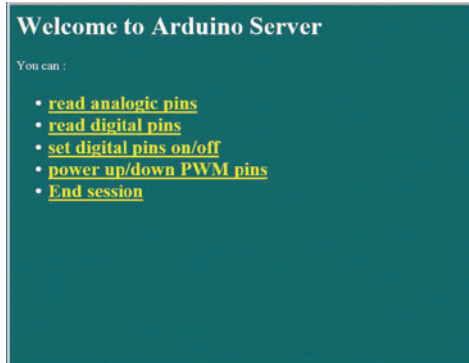


Fig. 3 - Home page del Web Server.

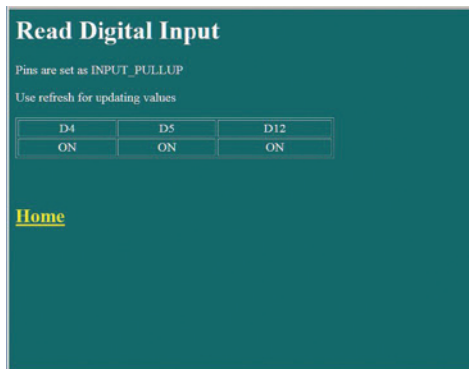


Fig. 4

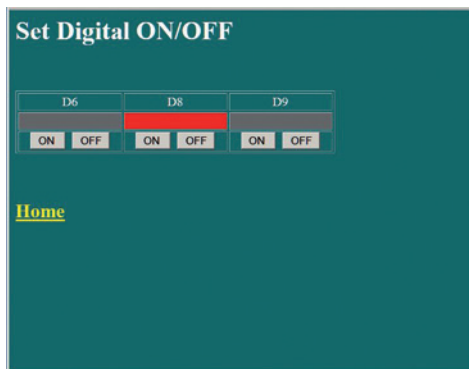


Fig. 5

Listato 4

```
#include <WiFi.h>
WiFi WiFi;
setup()
{
    WiFi.begin();
    WiFi.ConnectWPAWithPSK("MioAcp", "pippo");
    server=WiFi.openServerTCP(5000);
}
loop()
{
    if (!OpenCom)
        socket=WiFi.pollingAccept(server);
    if(socket<255) OpenCom=true;
    if (OpenCom)
        record=WiFi.readDataLn(socket);
    WiFi.writeDataLn(socket, ".....");
}
```

a comandi lanciati dal browser (per esempio mediante pulsanti di form), bisogna leggere i dati inviati dalla Request insieme al nome della risorsa. La situazione è diversa se la richiesta è arrivata sotto forma di GET piuttosto che sotto forma di POST (i due metodi cardine del protocollo HTTP). Si tratta, in ogni caso, di utilizzare finalmente quell'argomento passato alla call-back function da **getRequest()**.

Nel primo caso, i dati sono rappresentati da coppie nome-valore che identificano un parametro. I parametri sono accodati al nome della risorsa in un formato che codifica spazi e caratteri speciali e che possiamo chiamare query-string. La query-string è fornita sempre (anche se di lunghezza zero) alla call-back function (infatti fa parte del suo prototipo). Allora possiamo utilizzare la funzione **getParameter()** per recuperare il valore (sempre sotto forma di stringa) del parametro con un certo nome (**Listato 3**).

Nel secondo caso, invece, la query-string conterrà valori che possono essere in formato query-string (come in genere fanno le form) oppure in un formato qualunque. Bisogna, comunque, tener presente che il buffer che contiene la query-string è fornito dalla libreria ed ha lunghezza di 64 caratteri (ma ridefinibile con

define su *HTTPLib.h*). I dati in eccesso vengono perduti. Negli esempi è contenuto un completo Web Server che permette di leggere valori analogici e valori digitali, di attivare e disattivare output digitali e infine di regolare due uscite PWM. Lo sketch è particolarmente compatto (la metà è costituita dalle pagine html in PROG MEM) grazie all'automazione prodotta dalle funzioni **getRequest()** e **sendDynResponse()**.

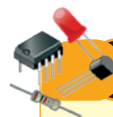
Se, invece, Arduino vuole essere usato come client di un Web Application Server (o di un più semplice CGI), utilizzeremo le funzioni **sendRequest()** e **getResponse()**. **SendRequest()** è in effetti costituita da due separate funzioni in base alla modalità che si vuole utilizzare: GET o POST. Nel caso utilizzassimo **sendRequestGET()** forniremo sia il nome della risorsa sia i parametri in un'unica query-string. Nel caso utilizzassimo **sendRequestPOST()** forniremo separatamente il nome della risorsa e i dati collocati in un buffer di tipo null-terminated string, in questo modo:

```
WiFi.addParameter(query, 128, "/
TestClient", NULL);
WiFi.addParameter(query, 128, "A1",
sal);
:
WiFi.sendRequestGET(csocket, query);
```

Oppure così:

```
sprintf(rec, "%d %d %d
%d", an1, an2, d1, d2);
WiFi.sendRequestPOST(csocket, "/
TestClient", rec);
```

Nel caso dell'utilizzo di **sendRequestGET()**, la query-string è stata formata con l'aiuto della funzione **addParameter()**. La prima volta inizializzando la query-string con il nome della risorsa (valore null), e poi con le coppie nome-valore per i singoli parametri. La funzione **getResponse()**, è quella da utilizzare per recuperare la risposta dal server. Questa può essere anche formata da numerosi dati in vari formati: dalla pagina HTML, a dati in formato XML, JSON o csv (comma separated values). Nel caso i dati non possano essere contenuti in un unico buffer, è possibile richiamare la funzione **getNextResponseBuffer()** in un loop fino a che non ritorni 0. Nel **Listato 4** trovate un esempio elementare di connessione per ricevere e spedire record. ■

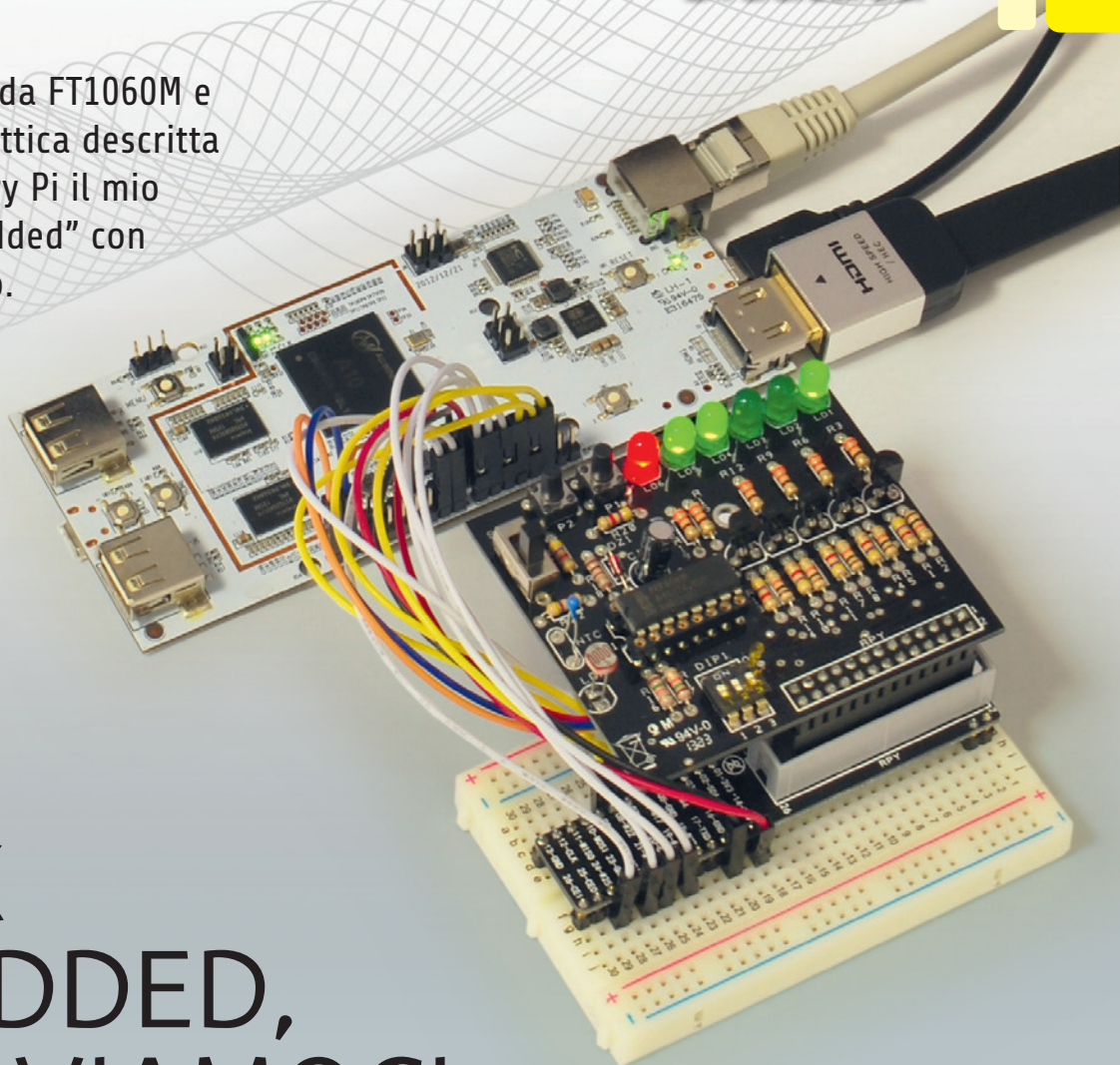


per il MATERIALE

Con le informazioni contenute nel presente articolo, chiunque può realizzare questo progetto. Tuttavia abbiamo ritenuto di mettere a disposizione questo WiFi Shield già montato e collaudato in modo da offrire a tutti l'opportunità di dotare di connessione WiFi la propria board Arduino. Lo shield già montato costa 56,00 Euro IVA compresa ed è contraddistinto dal codice FT1076M.

Il materiale va richiesto a:
Futura Elettronica, Via Adige 11,
21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
<http://www.futurashop.it>

Utilizziamo la scheda FT1060M e l'applicazione didattica descritta nel libro "Raspberry Pi il mio primo Linux Embedded" con il micro PC pcDuino. Un esercizio di "porting" di un'applicazione tra due sistemi con l'analisi e la risoluzione delle differenze.



LINUX EMBEDDED, RIPROVIAMOCI CON **pcDuino**

di MARCO MAGAGNIN

Chi ha seguito la serie di articoli sulla rivista dedicati a Raspberry Pi si ricorderà che avevamo previsto la crescita e la diffusione dei dispositivi in miniatura ARM RISC che possono funzionare con GNU/Linux. Quello che non avevamo previsto è stato il fatto che, più che di crescita, possiamo parlare di esplosione, cioè dell'arrivo sul mercato di dispositivi eterogenei, con caratteristiche diverse e che richiedono

diverse procedure per "addomesticarli" all'utilizzo del sistema operativo GNU/Linux. Nell'articolo sul numero 177 abbiamo presentato una panoramica delle schede disponibili, con particolare attenzione alla loro utilizzabilità in ambito didattico e alla presenza o meno di espansioni GPIO, che le rendono appetibili per un utilizzo "embedded".

Tra le schede presentate ci siamo soffermati su

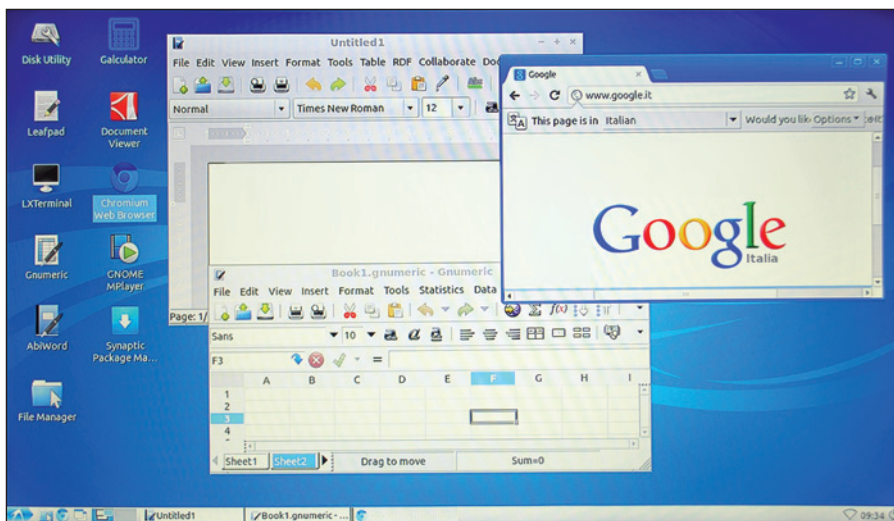


Fig. 2

pcDuino, una soluzione “ponte” tra il mondo GNU/Linux ed il mondo Arduino, e quindi particolarmente adatta a chi, avendo già acquisito esperienza in quest’ultimo ambiente, voglia affrontare l’ulteriore impegno di approfondire il mondo GNU/Linux. Infatti pcDuino ospita la distribuzione “Linaro” di GNU/Linux, derivata da Ubuntu e dotata di interfaccia grafica desktop come visibile in **Fig. 1**. In queste pagine vi proponiamo di familiarizzare con l’utilizzo “embedded” di questa scheda utilizzando la scheda didattica FT1060M ed i contenuti del libro “Raspberry Pi - Il mio primo Linux embedded”. Il libro propone un percorso di familiarizzazione con il mondo GNU/Linux embedded che si concretizza

nella realizzazione di un progetto web che permette di gestire funzionalità con sensori, interruttori e messaggi vocali, sia in locale che in remoto. La videata del progetto è visibile in **Fig. 2** e l’applicazione si basa su strumenti professionali come il web server Apache2, il database MySQL, il server “emoncms” ed i linguaggi di programmazione HTML, PHP, Python e javascript. Sono descritte anche le tecniche di comunicazione TCP/IP AJAX e JSON. A tal fine abbiamo preparato un documento di “porting”, scaricabile gratuitamente dal blog della rivista (<http://blog.elettronica.in>.

Fig. 1

it) che descrive le possibili soluzioni per sopperire alle differenze tra le due schede ed i listati dei programmi adattati a funzionare su pcDuino. Tutti i listati sono comunque scaricabili dallo stesso sito.

Per facilitare il compito a chi si vuole realmente “cimentare” in questa esperienza è stato realizzato il bundle “7300-PCDUINOKIT”. Questo kit contiene i componenti visibili in **Fig. 3**, che permettono di adattare la scheda FT1060M alla piedinatura dei pin di pcDuino. In concreto abbiamo collegato alla scheda FT1060M, un’ulteriore scheda di espansione, la FT1072K che riporta tutti i pin del connettore GPIO su un connettore adatto ad essere innestato su una breadboard di sperimentazione. Tutti i collegamenti necessari tra i pin ex-Raspberry Pi ed i connettori di pcDuino sono realizzati utilizzando i cavetti con terminali maschio-femmina. Il risultato è quello visibile nella foto di **Fig. 4** e lo schema dei “collegamenti” è riportato in **Fig. 5**.

Ci siamo “affezionati” alla sche-

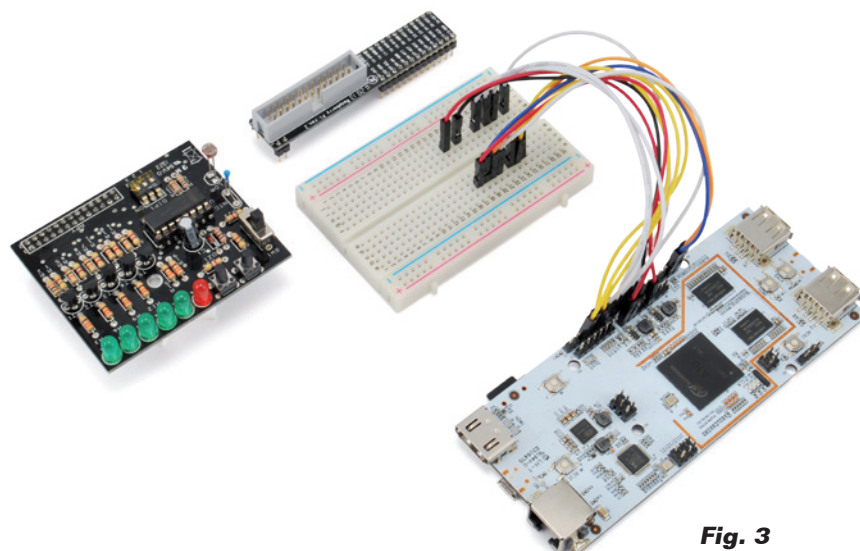


Fig. 3

da pcDuino, perchè offre una serie di funzionalità che non sono presenti nella scheda Raspberry Pi e che ci permettono di risolvere applicazioni particolari. Parliamo specificatamente della presenza di pin in grado di accettare ingressi analogici e di produrre in uscita segnali PWM. La presenza di un SoC Allwinner A10, dotato di CPU ARM Cortex A8 da 1 GHz, ci permette anche di avere a disposizione un po' di potenza in più per le applicazioni che la richiedono. Fatte tutte le considerazioni, incluse quelle economiche, ci troviamo in presenza di una scheda che vale la pena di approfondire. Nell'implementazione di pcDuino i file di device dei pin del GPIO si trovano nelle cartelle:

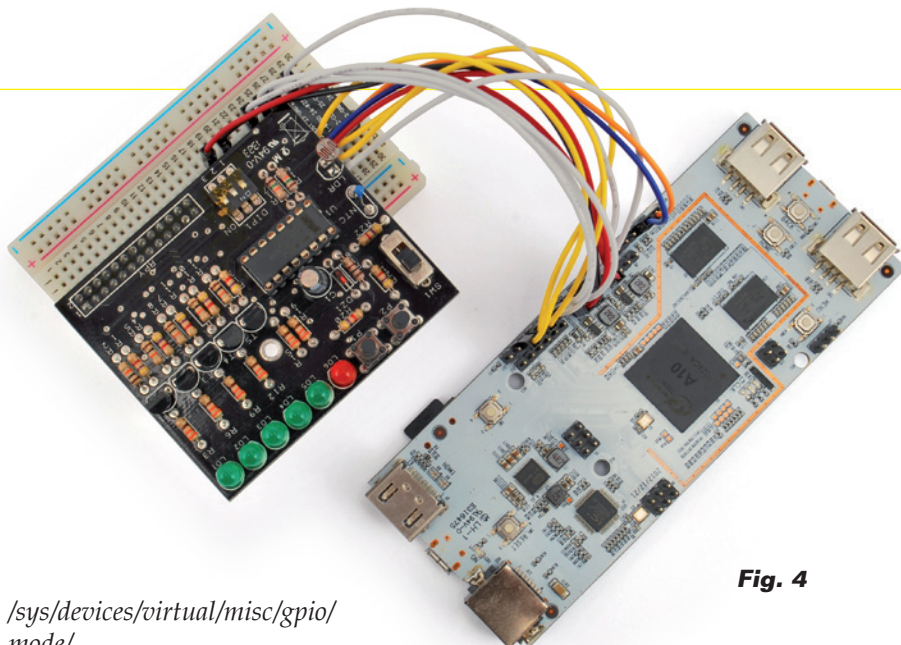


Fig. 4

`/sys/devices/virtual/misc/gpio/mode/`
`/sys/devices/virtual/misc/gpio/pin/`

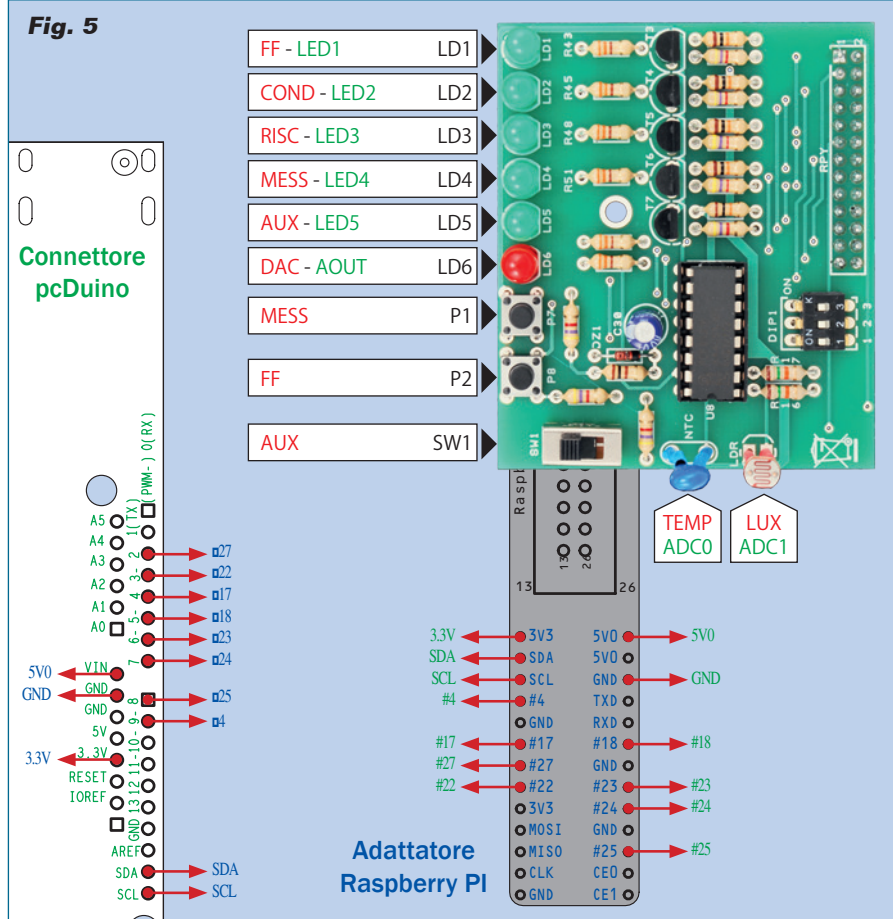
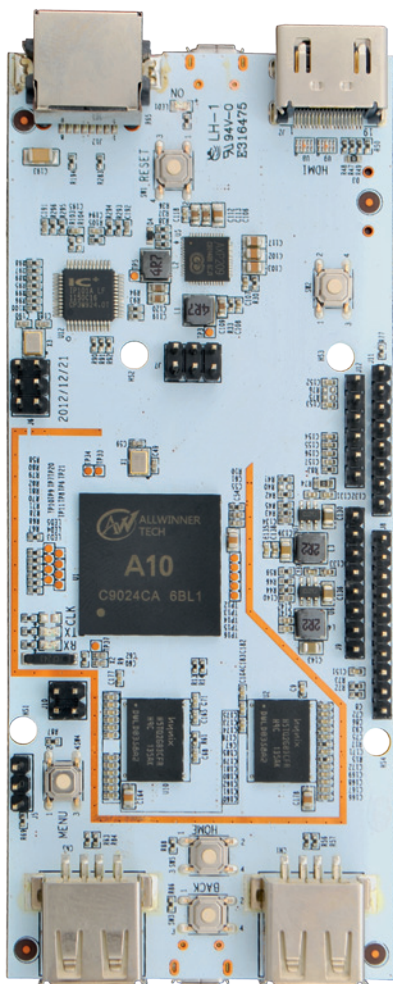
Nella cartella visibile in **Fig. 6**:

`/sys/devices/virtual/misc/gpio/mode/`

si trovano i file di device con i quali è possibile impostare le

modalità di utilizzo di ciascun pin; sono previsti diversi valori, i principali sono:

- '0' - imposta il pin di I/O come INPUT;
- '1' - imposta il pin di I/O come OUTPUT;



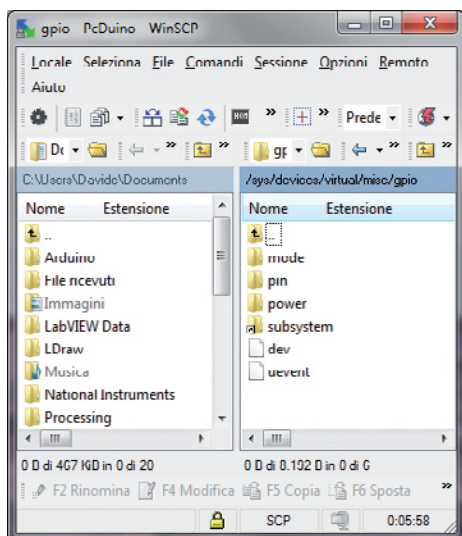


Fig. 6

- '8' - imposta il pin di I/O come INPUT, attivando la resistenza di pull up interna.

I valori dei pin di I/O vengono letti e scritti nei file di device corrispondenti nella cartella (Fig. 7): `/sys/devices/virtual/misc/gpio/pin/`. Ricordiamo che il livello alto dei pin corrisponde a 3,3 V e che la corrente fornita da ciascun pin non è paragonabile a quella fornita dai pin di Arduino, ma è comunque sufficiente a mantenere acceso un LED.

Se vogliamo utilizzare i pin che possono essere utilizzati come convertitori ADC possiamo leggere i loro valori nei file di device, contenuti nella cartella `"/proc/"`, con nome `adcx` (Fig. 8) dove "x" indica il pin ADC desiderato, da 0 a 5.

In Fig. 9 possiamo vedere il risultato delle letture degli ingressi ADC dove è possibile verificare che ADC0 e ADC1 hanno una risoluzione a 6 bit mentre i rimanenti hanno una risoluzione a 12 bit. Infine possiamo impostare i valori dei pin che forniscono funzionalità PWM, utilizzando i file di device nelle cartelle:

`/sys/class/leds/pwm/brightness` (Fig. 10), dove anche in questo caso "x" indica il pin PWM desiderato, da 0 a 5.

Per impostare il valore PWM si scrive il valore desiderato nel file di device "brightness", nella cartella corrispondente al PWM richiesto (Fig. 11); per ciascun pin si possono impostare valori che vanno da 0 al valore contenuto nel file:

`/sys/class/leds/pwm/max_brightness`

diverso per ogni uscita (Fig. 12). Per quanto riguarda l'utilizzo del bus I2C, utilizzando il linguaggio Python vale quanto descritto negli articoli dedicati a Raspberry Pi, con l'utilizzo della libreria "smbus". Per la comunicazione SPI, sempre in Python, è disponibile la libreria SPI-Py.

Molte informazioni, manuali, guide (in inglese) e file da scaricare si trovano sul sito dedicato a pcDuino all'indirizzo:

<http://www.pcdduino.com>

Descriviamo ora le differenze tra

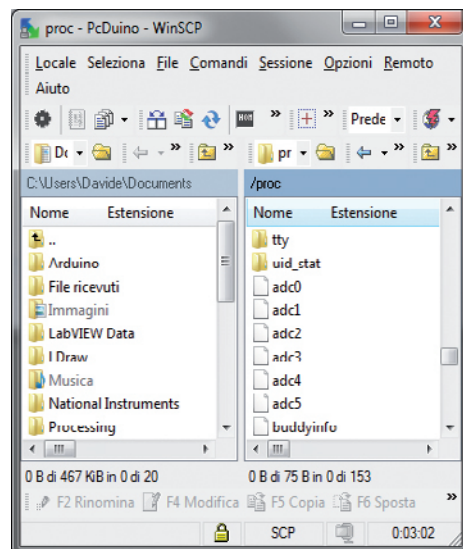


Fig. 8

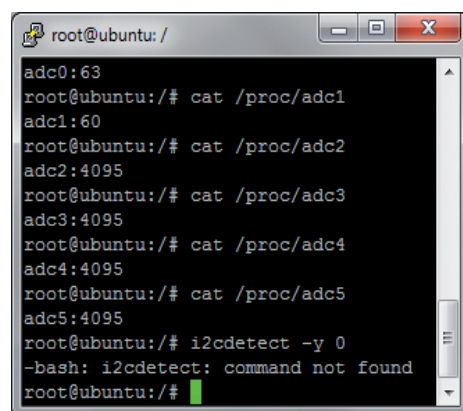


Fig. 9

la scheda Raspberry Pi e pcDuino che hanno richiesto interventi di adattamento della nostra applicazione "didattica" per portarla a funzionare su quest'ultima scheda. La più evidente è la differente dislocazione fisica dei pin di I/O della scheda pcDuino rispetto a Raspberry Pi. Questa differenza ha richiesto l'adozione del cablaggio intermedio per collegare i pin di pcDuino alla scheda FT1060M. La seconda differenza, un po' meno evidente, è la nomenclatura dei pin di I/O di pcDuino rispetto al connettore di Raspberry Pi. Come potete verificare nel documento di "porting", questo ha richiesto la modifica della nomenclatura dei pin all'interno di tutti i programmi che ne fanno uso e la modifica dei rife-

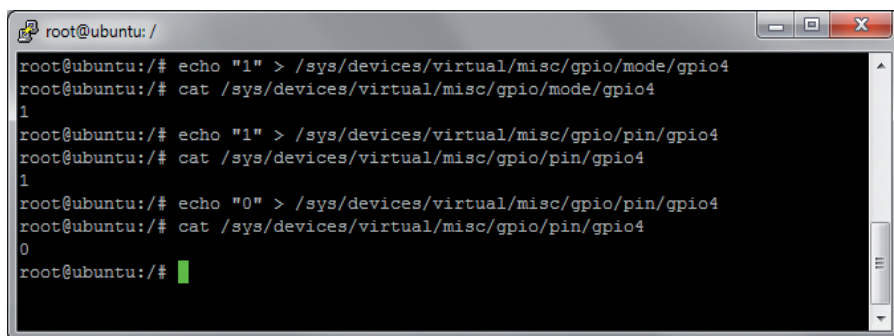


Fig. 7

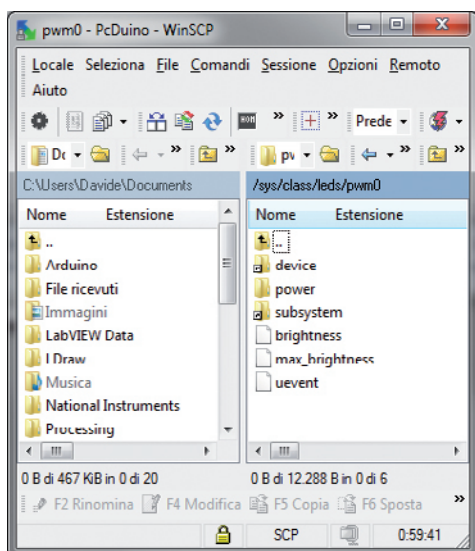


Fig. 10

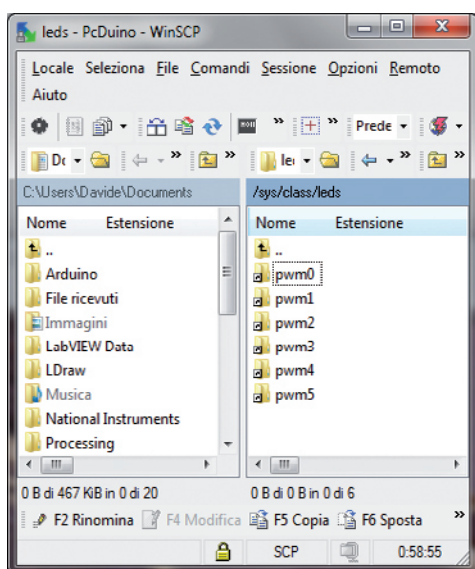


Fig. 11

rimenti ai pin stessi all'interno delle tabelle del database MySQL. Per rimanere nell'ambito della gestione dei pin di I/O, abbiamo dovuto affrontare altre due differenze. Il device driver di gestione del GPIO non prevede la gestione di eventi come il cambio di livello di un pin a seguito della pressione di un pulsante. A conferma di questo, nei file di device dei pin di GPIO, in `/sys/devices/virtual/misc/gpio/` non troviamo l'equivalente del file `"edge"` che troviamo in `/sys/class/gpio` in Raspberry Pi. Questa situazione

preclude l'utilizzo della tecnica di polling, assai efficiente quando si devono gestire eventi esterni come la pressione di pulsanti ed il cambio di stato di interruttori. Abbiamo sostituito questa gestione con dei loop che interrogano ciclicamente lo stato dei pin ai quali sono collegati i pulsanti e l'interruttore. Si ottiene all'incirca lo stesso risultato con un maggior "consumo" di CPU. Non è necessario installare la libreria `"python-rpi.gpio"` per la gestione del GPIO con i programmi Python, in quanto la gestione del GPIO nel caso di pcDuino è diversa rispetto alla gestione in Raspberry Pi. Anche il nome del pacchetto `"python-rpi.gpio"` indica che è stato chiaramente scritto per Raspberry Pi. Il driver di gestione del GPIO per pcDuino, utilizza unicamente i file di device, non ha la possibilità di gestire gli interrupt ed è stato compilato come modulo non permanente, pertanto deve essere caricato con il comando:

modprobe gpio

Un'ulteriore notevole differenza è data dalla mancanza di un'uscita audio per la gestione della funzionalità text to speech. Nel

documento di porting, reperibile nel blog di *Elettronica In*, abbiamo risolto questo problema indirizzando l'audio verso l'uscita HDMI. Lo svantaggio è che per far funzionare la messaggistica vocale è necessario collegare il monitor HDMI ed eseguire il login come utente `"ubuntu"`. A questo proposito anche tutta l'applicazione deve essere eseguita come utente `"ubuntu"` in quanto la gestione del GPIO, contrariamente a quanto avviene nel caso di Raspberry Pi, richiede espressamente un utente diverso da root. Riguardo l'uscita audio, in questo articolo vi proponiamo una soluzione differente, che evita di dover utilizzare il monitor unicamente per l'uscita audio. La soluzione che vi proponiamo prevede l'utilizzo di un dongle `"Audio controller"` con interfaccia USB. Noi abbiamo provato il modello `"3D Sound"` reperibile presso Futura Elettronica con il codice `8220-PCUSBSND`, e visibile in Fig. 13. Lo inseriremo in una presa USB libera di pcDuino, e colleghiamo all'uscita audio il jack di un altoparlante autoalimentato come il modello `8220-PCMPSP`, oppure di una cuffia stereo. La `"configurazione"` finale dell'assemblaggio è visibile

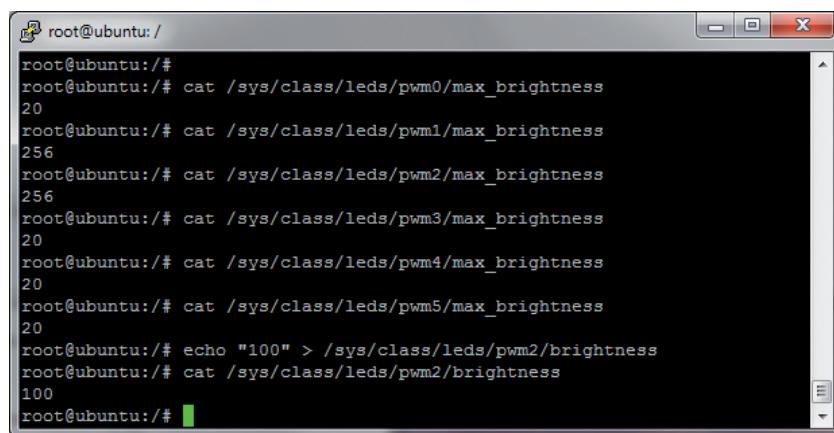


Fig. 12



Fig. 13

in Fig. 14. Una volta inserito il dongle USB, dovrebbe essere immediatamente riconosciuto dal sistema operativo di pcDuino. Per sincerarcene apriamo una finestra di terminale utilizzando lo strumento Putty e digitiamo il comando

alsamixer

che ci presenta la finestra di dialogo visibile in Fig. 15. ALSA (Advanced Linux Sound Architecture) è il driver a livello kernel che gestisce il suono nel sistema operativo di pcDuino. La gestione del suono in GNU/Linux, richiede strati di driver e configurazioni a volte decisamente complessi e dipendenti dalle periferiche e dai programmi utilizzati. Questo per mantenere compatibilità con i sistemi di controllo audio di generazioni e "filosofie" diverse. Nel nostro caso basta quanto abbiamo installato sinora. Torniamo alla pagina di gestione dei dispositivi audio. Per

verificare se il nostro dongle è stato correttamente riconosciuto premiamo il tasto "F6" che ci presenta la lista dei dispositivi audio riconosciuti dal sistema (Fig. 16). Selezioniamo il dongle USB, nel nostro caso "C-Media USB Headphone Set" e ci troveremo nella pagina di gestione del dispositivo (Fig. 17).

In questa pagina possiamo attivare o disattivare l'altoparlante od il microfono, ed impostare il volume. Con i tasti freccia a destra e a sinistra ci si sposta da un canale all'altro, rispettivamente altoparlante, microfono e capture. Il canale attivo ha la descrizione in rosso. Con il tasto "M" si attiva e disattiva il canale, la scritta "OO" in basso indica che il canale è attivo, la scritta "MM" indica che il canale è disattivo. Infine con i tasti freccia su e freccia giù si alza o si abbassa il volume. Con il tasto ESC si esce dalla pagina di configurazione.

Ora proviamo il funzionamento dell'altoparlante. Per poter funzionare, il canale audio richiede che i comandi vengano eseguiti come utente root. Se abbiamo eseguito il "login" come utente "ubuntu" dobbiamo far precedere ciascun comando da "sudo" (Superuser Do), che ci permette di eseguire alcuni comandi come

utente root, anche se lanciati da un utente comune. Equivale a selezionare, in ambiente Windows, l'opzione "Esegui come amministratore". Digitiamo il comando (fig. 18):

sudo speaker-test -c1 -t wav

Se tutto funziona sentiremo una voce che si ripete in continuazione fino a quando non la interrompiamo con "CTRL"+"C". Ora possiamo eseguire le poche modifiche ai programmi della nostra applicazione per renderle compatibili con la nuova configurazione audio. Per esercizio modifichiamo il programma "parla.py", reperibile tra quelli previsti negli esercizi del libro e del porting, nel modo visibile nel **Listato 1**. In pratica abbiamo aggiunto il comando "sudo" prima del comando "espeak". Per provarlo entriamo nella cartella /home/LR con il comando:

cd /home/LR

e digitiamo:

python parla.py

Per rendere compatibile il progetto con l'utilizzo del dongle audio USB, dobbiamo aggiungere il comando "sudo" ovunque venga richiamato il programma "espeak". Nella fattispecie, devono essere modificati i programmi GPIOInt_MESS.py e LEDServer.py, nella cartella /home/LR.

SERVER EMONCMS

Approfittiamo di questo articolo per dare un'indicazione valida anche per chi segue la realizzazione proposta nel libro "Raspberry Pi - Il mio primo Linux embedded". Rispetto alla pubblicazione del libro, il server "emoncms" ha subito degli aggiornamenti (at-

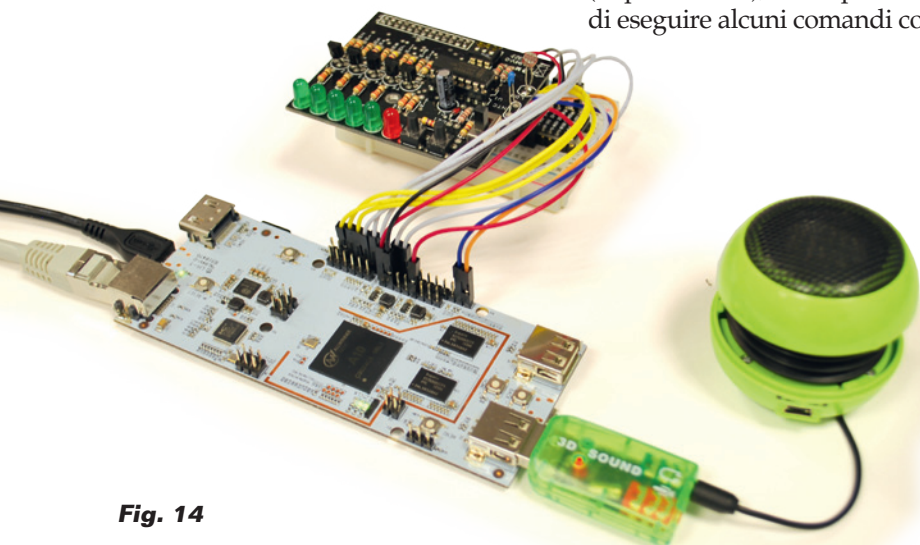


Fig. 14

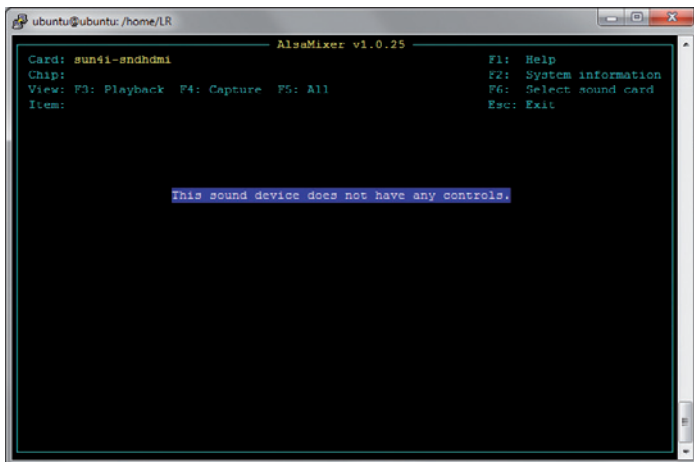


Fig. 15

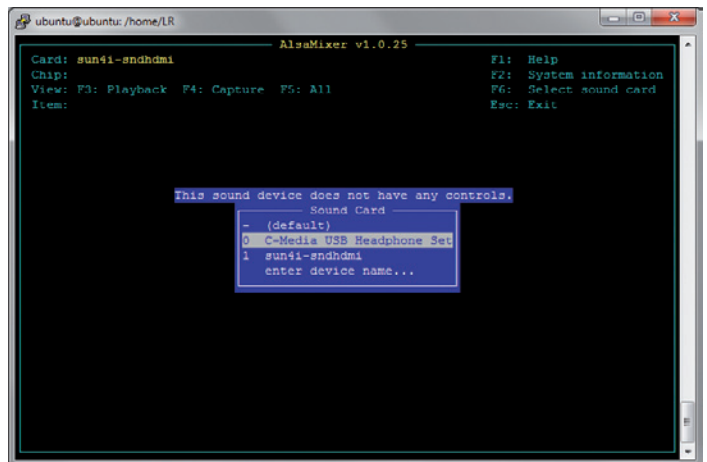


Fig. 16

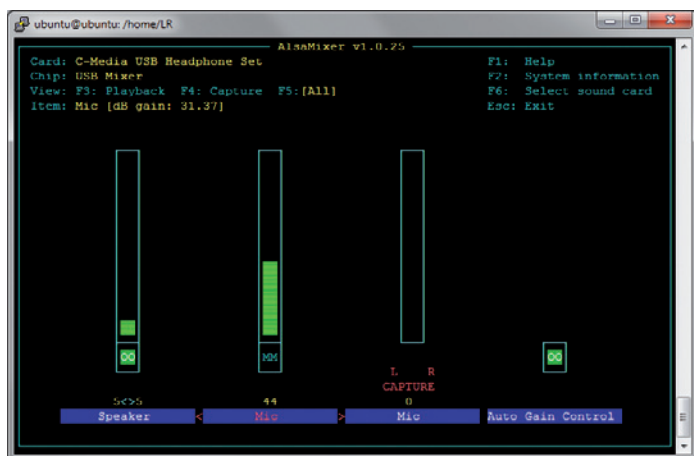


Fig. 17

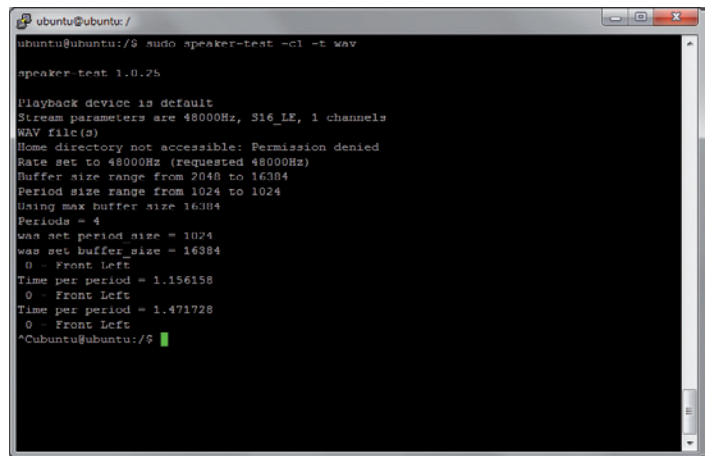


Fig. 18

tualmente è alla versione 6) che hanno interessato in particolare la memorizzazione e gestione dei "feed". In questa versione i "feed", nella configurazione predefinita, non sono memorizzati in tabelle MySQL, ma utilizzano il pacchetto di archiviazione "Timestore", che contrasta con la configurazione descritta nel libro. Timestore è un gestore database "NoSQL" particolarmente adatto a gestire ed elaborare serie

storiche di dati. Per continuare a utilizzare le tabelle MySQL per la memorizzazione dei "feed" è necessario modificare il file di configurazione: *settings.php* in */var/www/emoncms*. Il parametro da modificare è:
`$default_engine = Engine::TSTORE;`
da modificare in:
`$default_engine = Engine::MYSQL;`

Salvare e chiudere il file ed eseguire un reboot, per sicurezza.

ACCENDIAMO TUTTO

A questo punto possiamo lanciare tutta l'applicazione come descritto nel documento di porting. Un'ultima avvertenza, ricordatevi di armonizzare l'indirizzo IP di pc-Duino con quelli inseriti nei vari programmi per il funzionamento delle chiamate AJAX e JSON. La raccomandazione è di configu-

Listato 1

```
#!/usr/bin/python

import os
parla = "ciao a tutti"
os.system('sudo espeak "' + parla + '" -v it -p 50 -s 160 > /dev/null 2> /dev/null')
```

35^a MOSTRA REGIONALE DELL'ELETTRONICA SCANDIANO (REGGIO EMILIA) 15/16 FEBBRAIO CENTRO FIERISTICO 2014



**MERCATINO DELLE PULCI
RADIOAMATORIALI
TELEFONIA - COMPONENTISTICA
COMPUTER - HI-FI CAR - RADIANTISMO
CB e OM - VIDEOREGISTRAZIONE**

ORARI

**Sabato 15 Febbraio dalle 9 alle 18,30
Domenica 16 Febbraio dalle 9 alle 18**



**INGRESSO euro 7,50 RIDOTTO euro 5
GRATUITO per disabili e ragazzi fino a 14 anni**

**INFO: 0522/764302-857436
www.fierascandiano.it**

rare pcDuino con un indirizzo IP statico, se possibile quello che abbiamo utilizzato nella nostra applicazione ("192.168.0.43"). Per farlo, come indicato nel documento di porting, utilizzando WinSCP, entriamo nella cartella:

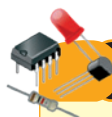
/etc/network

apriamo il file "interfaces" e configuriamolo come segue, ovviamente utilizzando le configurazioni IP compatibili con le caratteristiche della rete gestita dal nostro router.

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
```

```
auto eth0
iface eth0 inet static
    address 192.168.0.43
    netmask 255.255.255.0
    network 192.168.0.0
    broadcast 192.168.0.255
    gateway 192.168.0.254
```

Salviamo e chiudiamo il file, probabilmente ci verrà richiesta la password di root. Bene, abbiamo terminato. Potete provare tutte le funzionalità della nostra applicazione didattica, secondo quanto indicato nel libro, e apportarvi tutte le modifiche che desiderate. Buona sperimentazione e buon divertimento. ■



per il MATERIALE

Il set (cod. 7300-PCDUINOKIT) costa 129,00 Euro IVA compresa. La confezione comprende: la board pcDuino, il libro "Raspberry Pi il mio primo Linux Embedded", il fascicolo "Esperimenti con pcDuino", la scheda di sperimentazione analogico/digitale FT1060M, l'adattatore Breadboard per Raspberry, una Breadboard 40 contatti autoadesiva, una confezione di 70 jumper maschio/maschio, una micro SDCard HC da 4 GB contenente il software, un alimentatore switching con uscita USB 5 Vdc - 2 A, cavo HDMI lungo 1,5 metri, cavo USB da M(A) a Micro(B) lungo 0,7 metri e cavo FTP CAT5E lungo 0,75 metri.



Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287 - www.futurashop.it

UN PLOTTER PER PALLINE DI NATALE

di BORIS LANDONI



Il Natale è alle porte e chi più e chi meno ha rispolverato il presepe o l'albero e l'ha addobbato; sicuramente un po' tutti cerchiamo di rendere nuovo e diverso il nostro simbolo di Natale andando a comperare nuovi addobbi e palline di colore diverso. Se state cercando qualcosa di originale, valutate l'idea di farvi palline personalizzate, con disegni e scritte a piacimento. Come fare? Beh, se siete amanti del bricolage potete prendere pennarelli, polveri e colle e dare sfogo alla vostra creatività, ma siccome siete elettronici, siamo certi che sarete più stimolati dall'idea di decorare le palline con qualcosa di tecnologico. Ecco quindi per voi una

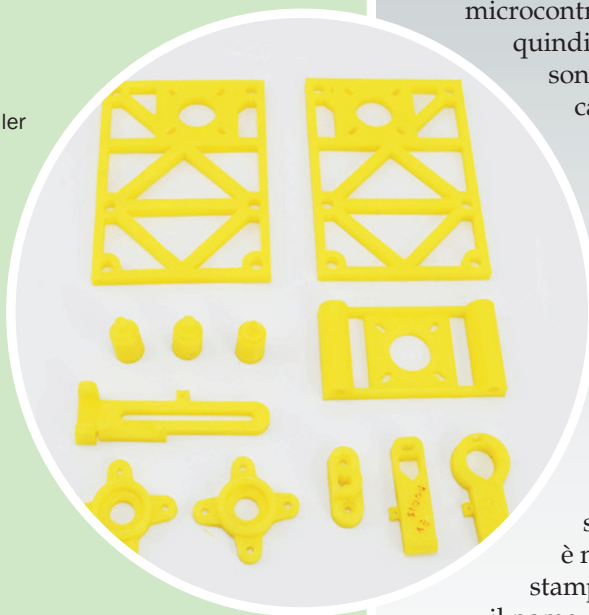
macchinetta ad hoc: una stampante o, meglio dire un plotter, che tratterà a pennarello su sfere e uova, disegni e scritte a piacere, derivati da modelli in file STL trasformati in file G-Code. La macchina si realizza facilmente con pochi pezzi ottenibili anche dalla stampa 3D (allo scopo, in un'apposita pagine di Thingiverse forniamo i relativi modelli) e una scheda di controllo che è poi la stessa proposta per la 3Drag e pubblicata nel fascicolo n° 175 di Eletttronica In; si collega ad un computer, nel quale bisogna far girare l'apposito software di stampa.

Realizziamo una stampante per decorare a pennarello le palline dell'albero di Natale con scritte e disegni adatti al periodo.

Cosa vi occorre?

La macchina si realizza con particolari meccanici, alcuni dei quali si trovano in commercio (viti, dadi, cuscinetti ecc.) mentre altri vanno autocostituiti: la soluzione migliore è farseli con una stampante 3D utilizzando l'ABS o il PLA.

- Pz. 1 Controller per motori passo-passo (cod. 3DCONTROLLER)
- Pz. 2 Driver per motori passo-passo (cod. 3DDRIVER) da montare sul controller
- Pz. 2 Strip M/F
- Pz. 1 Guaina termorestringente Ø 4,2 mm L = 500 mm
- Pz. 1 Set parti plastiche stampabili con la 3Drag;
- Pz. 18 Dado M8
- Pz. 8 Dado M4 (spessore 3,2 mm)
- Pz. 1 Dado M4 autobloccante
- Pz. 18 Rondella piana M8
- Pz. 2 Rondella piana M4 x 16
- Pz. 8 Rondella piana M3
- Pz. 8 Rondella piana M4
- Pz. 1 Vite TCEI M4 x 40
- Pz. 4 Vite TCEI M4 x 25
- Pz. 1 Vite TCEI M4 x 10
- Pz. 3 Vite TCEI M4 x 15
- Pz. 4 Vite TCEI M3 x 8
- Pz. 4 Vite TCEI M3 x 12
- Pz. 2 Vite autofilettante 2 x 7 mm (incluse nel set del servo)
- Pz. 2 Cuscinetto 8 x 22 x 7 mm
- Pz. 1 Molla a compressione Ø filo 0,7 mm, Ø ext 9,2 mm, L = 35 mm
- Pz. 1 Molla a trazione Ø filo 0,6 mm, Ø ext 7 mm, L = 35 mm
- Pz. 2 Motore passo-passo bipolare NEMA-17, 2,5 A (cod. STEPMOT3DRAG)
- Pz. 1 Micro servo, 9 g (cod. SERVO206) completo di leva da 18 mm
- Pz. 3 Barra filettata M8 da L = 230 mm
- Pz. 1 Barra filettata M8 da L = 140 mm



microcontrollore Microchip, quindi molti makers si sono ingegnati cercando di realizzare qualcosa di simile, ma basato su Arduino, in quanto seppure buono, l'hardware Microchip non può contare sul nutrito scambio di informazioni e sul firmware open di cui si giova da tempo la scheda di Banzi. Ne è nata una versione di stampante che ha preso il nome di Spherebot e che sostanzialmente condivide la struttura meccanica, ma si differenzia per l'elettronica.

Purtroppo la documentazione di questa creazione è abbastanza frammentata, a volte incompleta e spesso non aggiornata; ad esempio, il firmware disponibile in Rete non funzionava con le ultime versioni dell'IDE di Arduino. Questo è il motivo che ci ha spinto a rivedere l'intero progetto, cercando di realizzare delle linee guida che permettessero a chiunque di realizzare la propria Spherebot.

Sia quest'ultima che la Eggbot, si basano su una meccanica che prevede un sistema di presa della pallina in grado di far ruotare quest'ultima e su un servomotore montato su un carrello, azionato da un secondo stepper-motor: il servomotore alza la penna quando non deve scrivere e la abbassa quando invece bisogna disegnare. Il sistema di presa conta su una sorta di morsetto che blocca la sfera mediante due ventose in gomma e la fa ruotare (tramite un motore passo-passo) a velocità costante sincronizzandosi con il movimento della penna, guida-

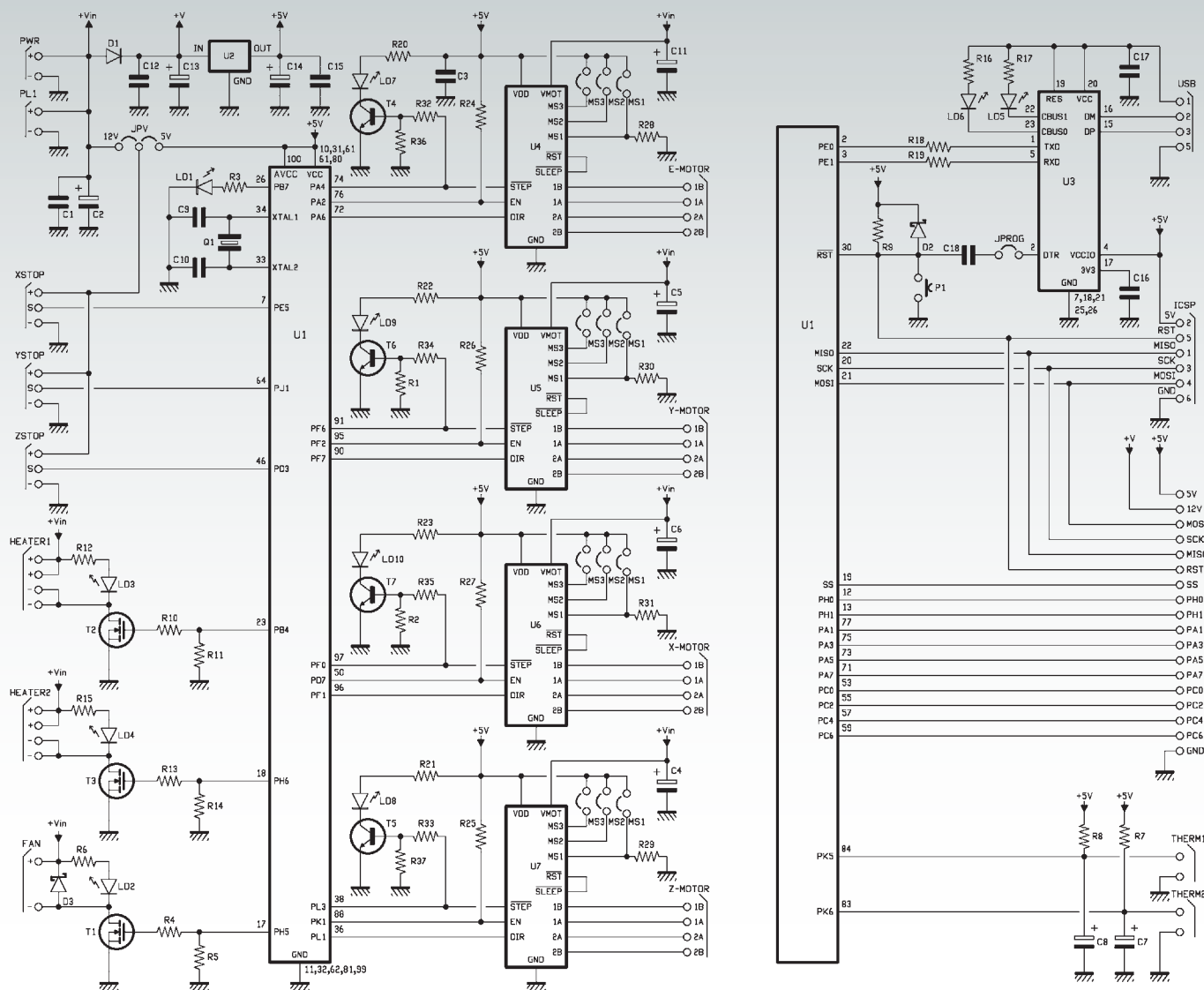


UN PO' DI STORIA

La macchina che proponiamo di realizzare è conosciuta nel mondo Internet come Eggbot (prende il nome dal fatto che nacque per disegnare e scrivere sulle uova...) o Spherebot: la prima è la famosa

produzione della Evil Mad Scientist, la quale, in ottica makers, ha reso tutto il progetto open source (vedere <http://egg-bot.com/>); la scheda di controllo (www.schmalzhaus.com/EBB/) proposta dal produttore è basata su un

[schema elettrico della **SCHEDA BASE**]

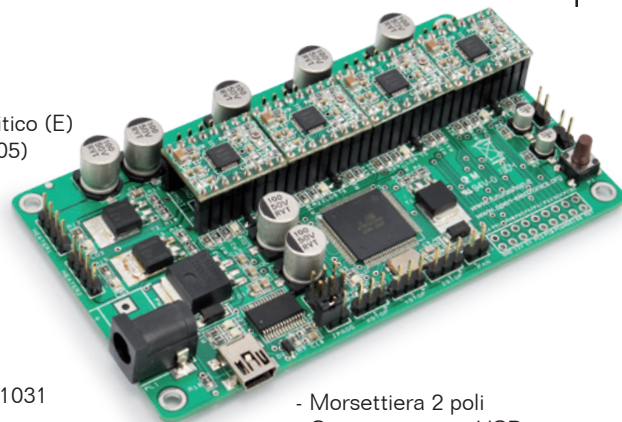


Elenco Componenti:

R1; R2: 10 kohm (0805)
 R3: 330 ohm (0805)
 R4, R10, R13: 10 ohm (0805)
 R5, R11, R14: 100 kohm (0805)
 R6: 1,8 kohm (0805)
 R7÷R9: 4,7 kohm (0805)
 R12, R15: 1,8 kohm (0805)
 R16, R17: 470 ohm (0805)
 R18, R19: 10 ohm (0805)
 R20÷R23: 330 ohm (0805)
 R24÷R27: 10 kohm (0805)
 R28÷R31: 100 kohm (0805)
 R32÷R35: 4,7 kohm (0805)
 R36, R37: 10 kohm (0805)
 C1, C3: 100 nF ceramico (0805)
 C2, C4÷C6: 100 μ F 25 VL elettrolitico (E)
 C7, C8: 10 μ F 35 VL elettrolitico (B)
 C9, C10: 22 pF ceramico (0805)

C11, C13, C14: 100 μ F 25 VL elettrolitico (E)
 C12, C15÷C18: 100 nF ceramico (0805)
 D1: GF1M
 D2: BAT42W
 D3: MBRA140TRPBF
 T1÷T3: BUK6215-75C
 T4÷T7: BC817
 U1: ATMEGA2560-16AU
 U2: MC7805ABD2T (D2PAK)
 U3: FT232RL
 U4÷U7: Driver motori cod. MD09/FT1031
 LD1: LED verde (0805)
 LD2÷LD10: LED rosso (0805)
 P1: Microswitch
 Q1: Quarzo 16 MHz (C7S)

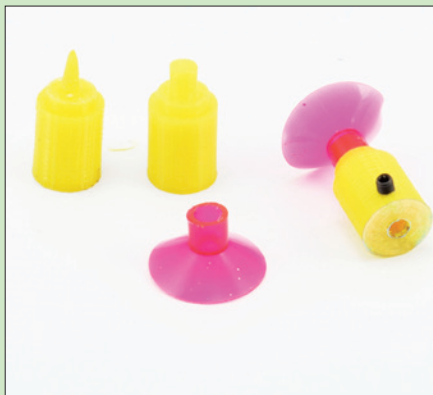
Varie:
 - Plug alimentazione



- Morsettiera 2 poli
- Connettore mini-USB
- Strip maschio 2 poli (4 pz.)
- Strip maschio 3 poli (5 pz.)
- Strip maschio 4 poli (6 pz.)
- Strip femmina 32 poli (2 pz.)
- Jumper
- Circuito stampato

L'assemblaggio della **meccanica**

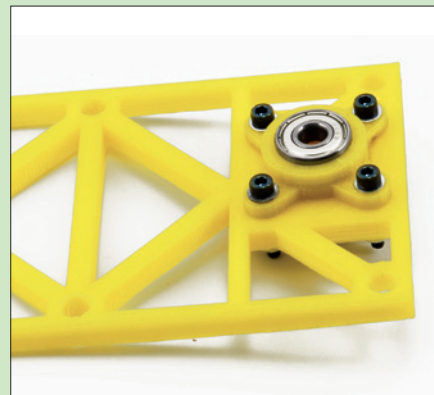
In questo riquadro vedete alcuni momenti del montaggio della parte meccanica del nostro plotter per le palline di Natale; tutti i dettagli meccanici, i rendering delle parti componenti, i file STL corrispondenti, si trovano in Internet su Thingiverse in un'apposita pagina web da noi creata



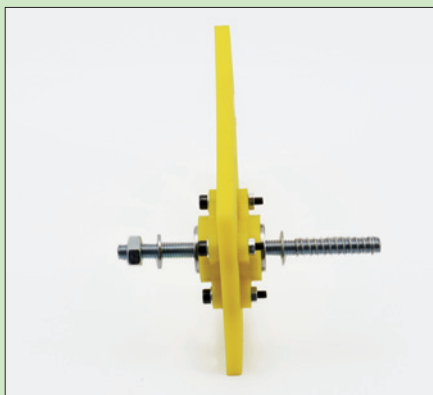
Stampati i supporti per le ventose, applicate le ventose e, a quello che va sull'albero dello stepper-motor, il grano per il fissaggio.



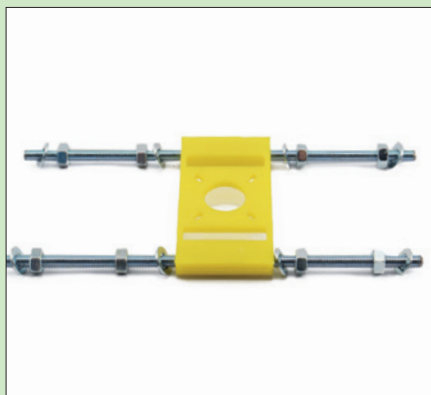
Avvitare alla barra filettata il supporto, con filettatura femmina, per la ventosa (quello che non prevede il grano).



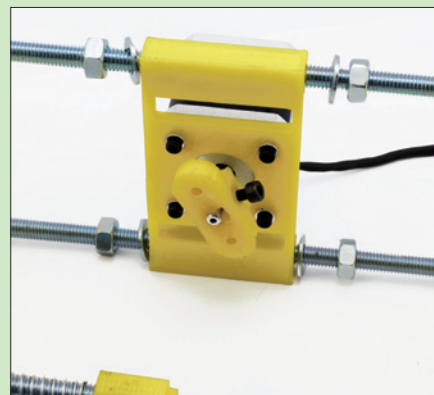
Montare i cuscinetti sulle apposite flange e avvitare queste ultime sul fianco opposto a quello dove prenderà posto lo stepper.



Inserire la barra filettata e montare all'esterno il bullone di fine corsa e la molla che premerà la palla da stampare.



Montare le barre filettate con bulloni e rondelle sul carrello portapenna, che avviterete ai fianchi della macchina.



Avvitare il motore passo-passo al carrello e fissare la staffa porta-servo al suo albero. Avvitare la struttura ai fianchi della macchina.

to da un secondo stepper-motor. Per tracciare scritte e disegni, la penna, una volta che il servomotore l'ha fatta scendere fino a che la punta tocca la pallina, si sposta lateralmente, mentre il sistema di presa ruota in un verso o nell'altro, in modo da creare il tratto verticale.

LA NOSTRA MACCHINA

La meccanica del plotter che vi proponiamo di realizzare è simile a quella delle macchine citate ed è composta da pezzi realizzati con una stampante 3D (ne trovate

i modelli in STL nel nostro sito www.elettronica.in e in un'apposita pagina di Thingiverse, accessibile da www.thingiverse.com/thing:201052), da alcune barre filettate e da un paio di cuscinetti a sfera; in questa pagina trovate la lista completa delle parti che la compongono. Rispetto alle versioni disponibili in Internet, la nostra SpheroBot presenta "un'area di stampa" maggiore, in modo da poter disegnare anche su palline da albero di Natale con un diametro di 6÷7 cm; quindi è più grande. Chi dispone di una stampante 3D

può utilizzare i file STL, aprendoli con Repetier Host (per esempio) e avviando la stampa; chi invece non può procedere alla stampa in 3D, può semplicemente fare una stampa tradizionale in scala dell'STL dall'apposito programma, poi lavorarsi le parti con altri materiali, come compensato, plastiche ecc. Chi ha una piccola CNC può procedere anche con quella, a patto di riuscire a generare i file G-Code.

IL SOFTWARE

Come la maggior parte delle

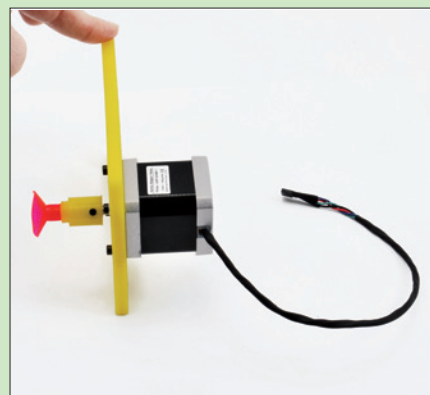
e accessibile all'indirizzo www.thingiverse.com/thing:201052. Una volta completata la struttura, collegate i cavi dei motori passo-passo e del servocomando agli appositi pin-strip della scheda controller per 3Drag.



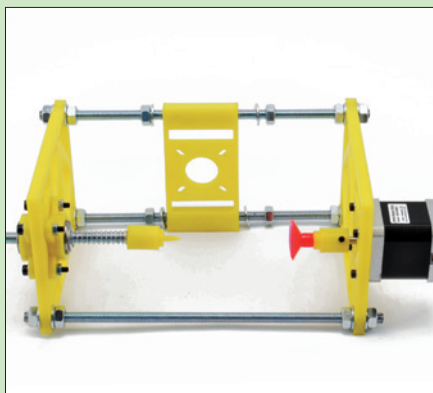
Montate il servo sull'apposito braccio, assemblato mediante l'apposita vite con dado. Il tutto si applicherà alla staffa porta-servo.



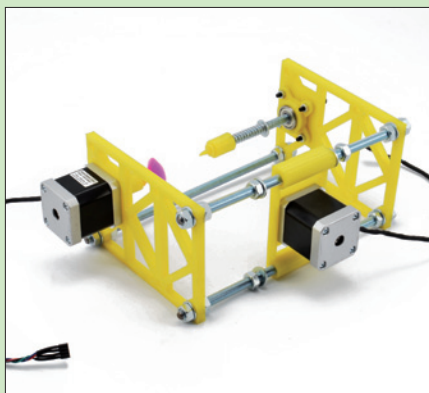
Dotate il braccio del servo della molla di richiamo e poi avvitate il bullone sul braccio opposto a quello che porta il servo.



Fissate a vite il motore-passo-passo sul fianco destro della macchina e montate sull'albero il supporto con la ventosa.



Il supporto con la ventosa si applica all'albero dello stepper-motor stringendo l'apposito grano. Ora la meccanica è assemblata.



Avvitate posteriormente al carrello il motore passo-passo che muoverà il gruppo con il servocomando e il portapenna.



Montate sul carrello il gruppo con il servocomando e il portapenna, fissandolo con le viti alla flangia del motore passo-passo.

macchine a controllo numerico, anche la nostra SpheroBot è in grado di gestire i codici G-code; per poterglieli passare è però necessario utilizzare un apposito software. Questo è stato scritto in Processing (rimandiamo al corso pubblicato su *Elettronica In* dal n° 159 al n° 162), lo stesso linguaggio con cui è scritto l'IDE di Arduino. La versione disponibile in Rete è stata adattata per renderlo pienamente compatibile con lo sketch presente sulla scheda di controllo. Una volta lanciato il software, bisogna scegliere la

COM virtuale cui colleghiamo la nostra scheda controller (e perciò la nostra SpheroBot) e aprire il file GCode che si vuole mandare

in "stampa"; immediatamente, aperto il file, la board piloterà i motori in base ai codici ricevuti (Fig. 1). Chi invece non volesse

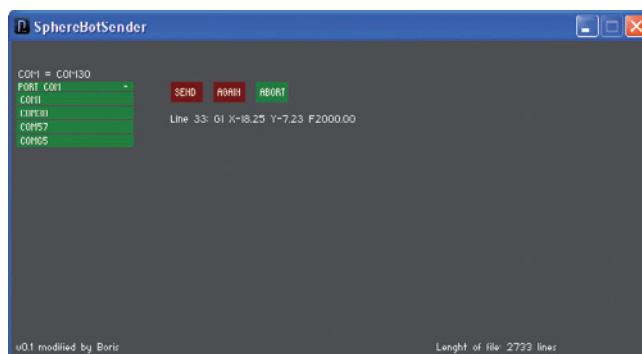
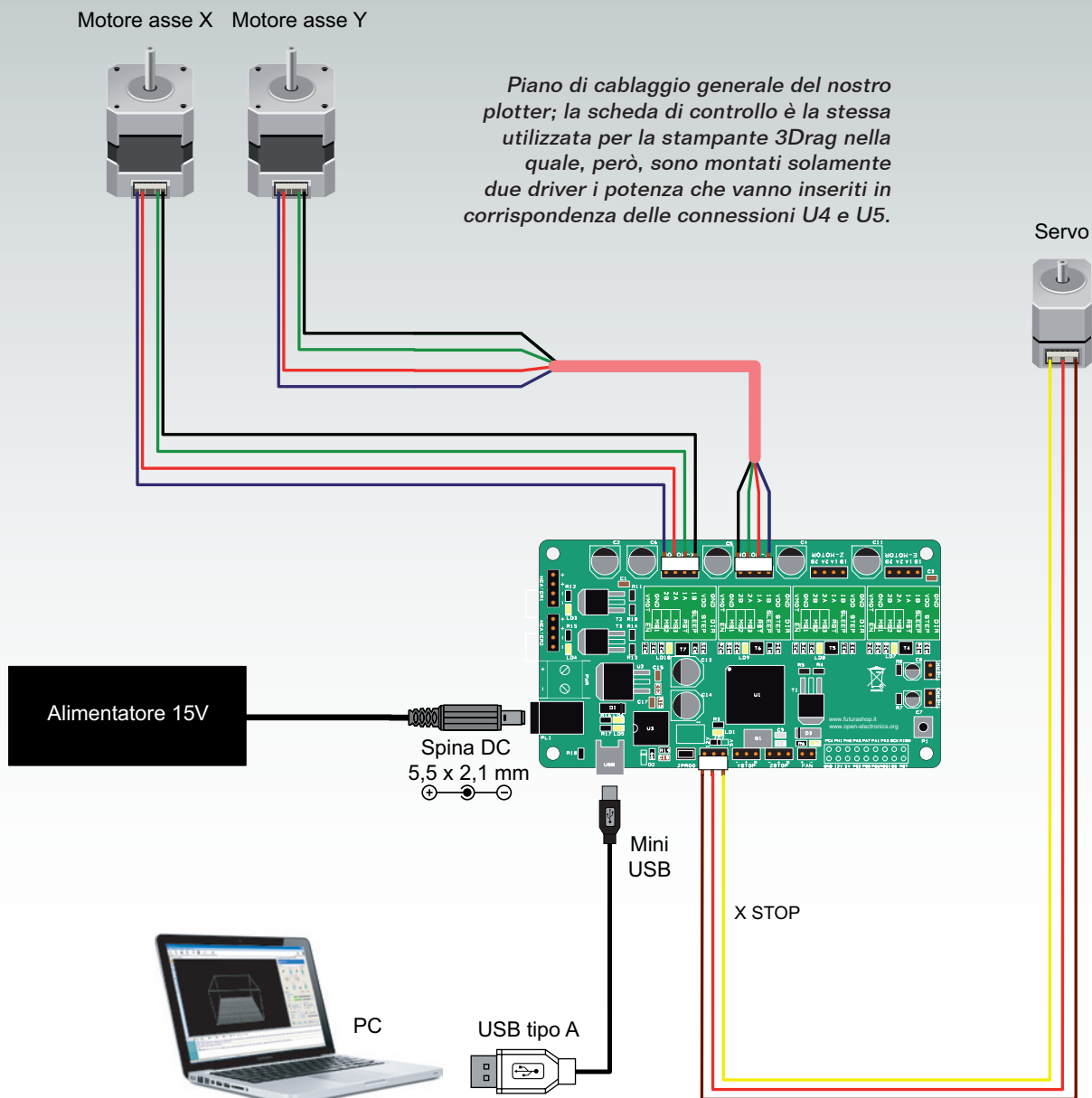


Fig. 1



cimentarsi con Processing e le sue librerie, può trovare comodo il software scaricabile da Github, alla pagina web <https://github.com/thunderbug1/Spherebot-Host-GUI>, che presenta una comoda interfaccia utente. Il software è molto intuitivo e permette anche di muovere manualmente tutti e tre i motori per posizionare correttamente il pennarello (Fig. 2).

OTTENERE IL FILE G-CODE

Per stampare le palline dell'albero di Natale con la nostra versione di Spherebot, occorre trasfor-

mare le vostre idee in stampe; quindi bisogna passare tramite un semplice software di conversione, in grado di trasformare in G-Code il vostro disegno o testo. Partiamo ancora una volta da un software completamente gratuito, che si chiama Inkscape e si scarica da <http://inkscape.org/it/>. Inkscape è un software per il disegno vettoriale basato sul formato SVG (Scalable Vector Graphics), capace di fornire un potente strumento grafico compatibile con gli standard XML, SVG e CSS. Inkscape inoltre è un'ap-

plicazione multiplatforma, che può essere eseguita sui sistemi operativi Microsoft Windows, Mac OS X e Unix-like (lo sviluppo è particolarmente focalizzato sulla piattaforma GNU/Linux e la particolarità del programma è di poter essere modificato e adattato alle varie esigenze grazie a degli script realizzati in Python). Inkscape può importare file di vario tipo e, grazie ad un'apposita estensione sviluppata dalla community, esportare i disegni vettoriali in G-Code. Ciò viene ottenuto, appunto, dal

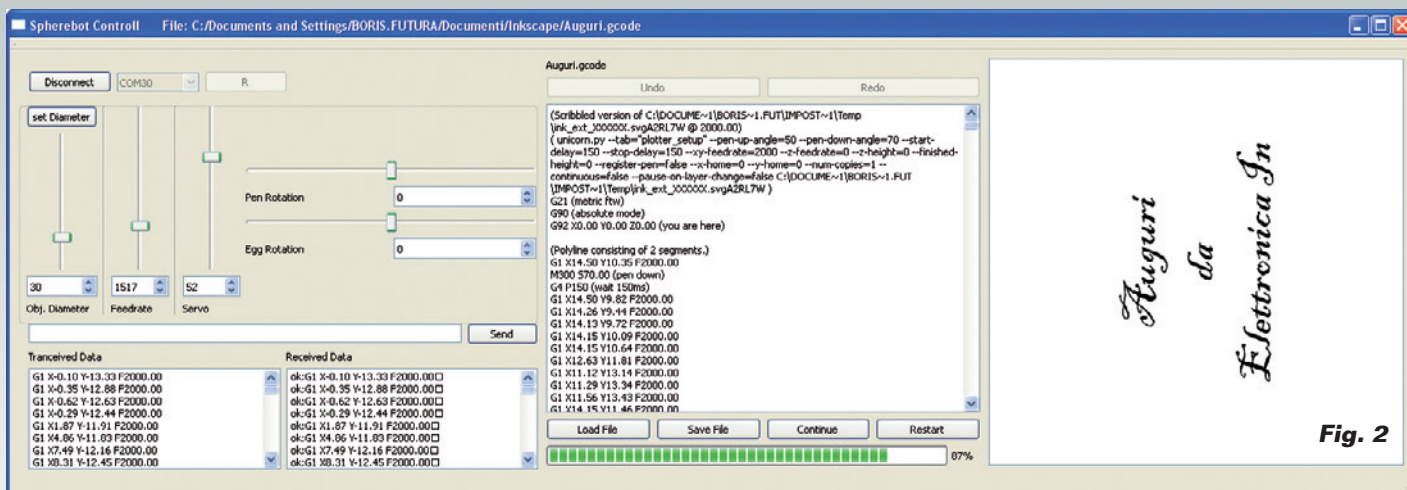


Fig. 2

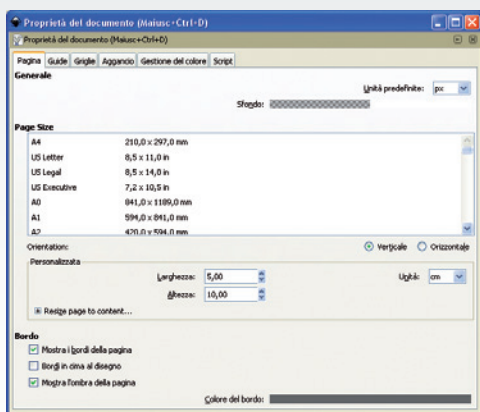


Fig. 3

plugin scaricabile e disponibile su Github alla pagina <https://github.com/martymcguire/inkscape-unicorn>.

L'installazione è semplice: è sufficiente copiare i file che si trovano sotto la cartella SRC dello ZIP scaricato sotto la cartella delle

estensioni di Inkscape, la quale, per impostazione predefinita, è *C:\Programmi\Inkscape\share\extensions*; la procedura si completa riavviando Inkscape. Sul web si trovano tantissimi tutorial per imparare ad utilizzare Inkscape; qui vi mostreremo solo come trasformare delle scritte in G-Code

lizzata: ad esempio 10 cm x 5 cm (Fig. 3).

Ora, con lo strumento testo (selezionabile con il tasto F8) digitate il vostro testo scegliendo il carattere che più vi piace. Per poter essere disegnato correttamente, il testo va ruotato di 180° (va riflesso in verticale), pertanto

MOTORE ASSE X

CONTROLLER

MOTORE ASSE Y

SERVO

per poterle aggiungere alle vostre decorazioni di Natale. Come prima cosa è necessario definire un'area di lavoro corretta in modo da rendersi conto se la scritta è della misura giusta. Per fare questo, dal menu *File* selezionate *Proprietà del documento*; nella finestra che si apre, trovate diversi formati standard disponibili. Ma la nostra pallina di Natale non è propriamente standard, pertanto definiamo una misura persona-

Guarda SpheroBot in azione su YouTube!



Vuoi vedere in azione il nostro SpheroBot? Scansiona il QR Code oppure collegati alla pagina <http://goo.gl/LYR8TO>.



selezionatelo con lo strumento *Seleziona e trasforma gli oggetti* (tasto F1) e, dal menu *Oggetto*, cliccate sulla voce *Ruota di 180° orari*. Centrate la scritta nella pagina ed eventualmente ridimensionatela utilizzando le apposite ancore (Fig. 6).

Adesso dovete creare il tracciato che il pennarello della macchina dovrà seguire per realizzare la scritta: allo scopo, dal menu *Tracciato*, selezioniamo la voce *Da oggetto a tracciato* (Fig. 7).

Ora non ci rimane altro che salvare il file corrispondente in G-Code: allo scopo, dal menu *File* selezioniamo la voce *Salva come* e tra i vari formati disponibili troveremo la possibilità di salvare il file con l'estensione *.G-Code. Confermando, si aprirà una finestra dove potremo definire alcuni parametri come la velocità di esecuzione, l'angolo che definisce il pennarello alzato, abbassato e altri parametri che vi consigliamo di copiare dalle figure Fig. 4 e Fig. 5.

L'ELETTRONICA

La nostra Spherebot è composta da due motori passo-passo ed un servomotore; come accennato, uno stepper serve per far ruotare sull'asse la pallina, mentre il secondo permette di muovere il pennarello intorno alla circonferenza della sfera. Il servomotore serve per alzare ed abbassare la

Fig. 4

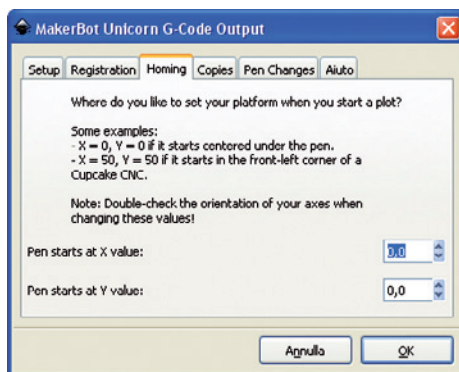
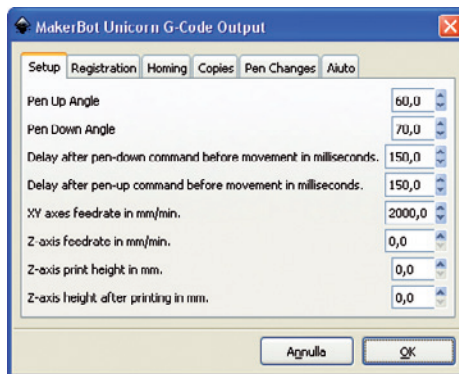


Fig. 5

penna. Inoltre, per consentire a chiunque di modificare facilmente il software senza bisogno di compilatori o programmatori, l'ideale sarebbe avere una board basata su Arduino. Tutte queste necessità vengono soddisfatte dalla scheda controller della stampante 3Drag da noi realizzata e pubblicata nel fascicolo n° 175; infatti questa scheda di controllo è basata su l'ATmega2560 (lo stesso microcontrollore di Arduino MEGA) ed è in grado di pilotare fino a quattro motori passo-passo. Nel nostro caso montiamo solamente due driver per stepper-motor e utilizziamo i motori previsti per gli assi X e Y: il primo per pilotare la rotazione della sfera sull'asse e il secondo per

muovere la penna. Per il servocomando utilizziamo, invece, i pin siglati X-STOP, originariamente previsti per la gestione degli end-stop (i finecorsa degli assi della 3Drag); infatti questi pin prevedono degli strip maschio dove è possibile inserire senza problemi il connettore del servo. L'unica modifica da apportare consiste nel chiudere il ponticello JPV per portare i 5V sullo strip (Fig. 8) X-STOP. C'è poi una modifica nel firmware, che serve in quanto X-STOP originariamente era un ingresso, mentre in questo caso usiamo la relativa linea di I/O dell'ATmega 2560 come uscita.

Fig. 6

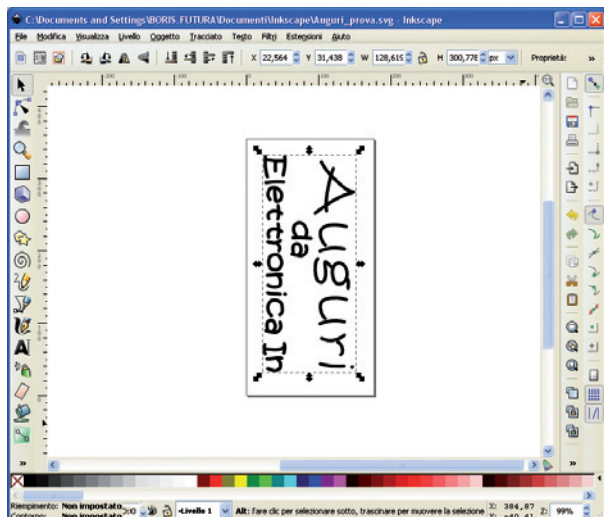


Fig. 7

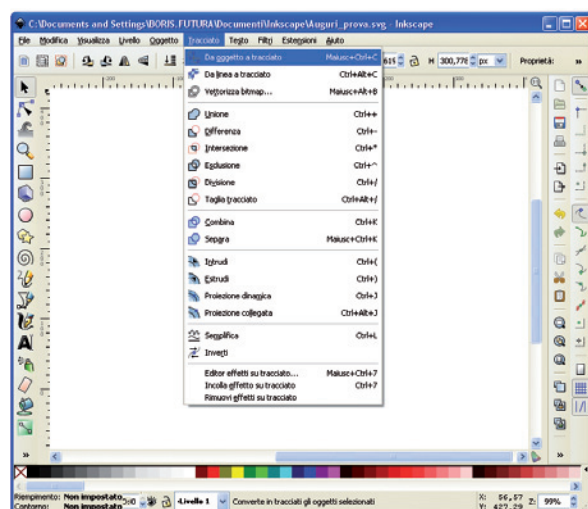
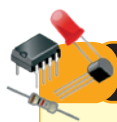




Fig. 8 - Per gestire il servo, dovete chiudere il jumper JPV su 5 V mediante una goccia di stagno.

IL FIRMWARE

Dato che l'hardware della 3Drag viene utilizzato per controllare una macchina con una funzione differente da quella prevista, è chiaro che bisogna caricare nella memoria dell'ATmega 2560 un firmware specifico, che sia in grado di governare i due stepper-motor e il servocomando. Trattandosi di un progetto open source, non abbiamo dovuto scrivere tutto il codice, ma è stato sufficiente trovare chi ci ha lavorato, quindi inserire le eventuali modifiche. Una grossa mano ce l'ha data Eberhard Rensch <http://pleasantsoftware.com/developer/3d>. Volendo, però, che lo sketch lavorasse anche con gli ultimi IDE di Arduino (superiori alla versione 1.0) abbiamo dovuto modificarlo un po' per renderlo compatibile. Lo sketch si occupa di gestire direttamente i due motori passo-passo e il servo in base ai comandi GCode ricevuti tramite la porta USB. La combinazione dei tre motori permette di scrivere sulla sfera senza grandi problemi. Scelta la board e modificato lo sketch, non resta altro che collegarla alla porta USB del PC, caricargli il firmware.



per il MATERIALE

Alcuni degli elementi necessari alla realizzazione di questo progetto vanno realizzati mediante una stampante 3D mentre altri vanno acquistati. In particolare, il controller per stampante senza driver (cod. 7350-3DCONTROLLER, Euro 48,00), i due driver (cod. 7350-3DDRIVER, Euro 11,00 cadauno), i due stepper motor (cod. 7300-STEPMOT3DRAG, Euro 24,00 cadauno) e il servo miniatura (cod. 7300-SERVO207, Euro 8,00). Tutti i prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
<http://www.futurashop.it>



Fiera

dell'informatica elettronica e radiantismo

Rovigo

Cen.Ser. - Viale Porta Adige 45

22 e 23 febbraio 2014

apertura dalle ore 9,00 alle ore 19,00

per tutta la durata della manifestazione saranno ospitate le iniziative

TRASHWARE

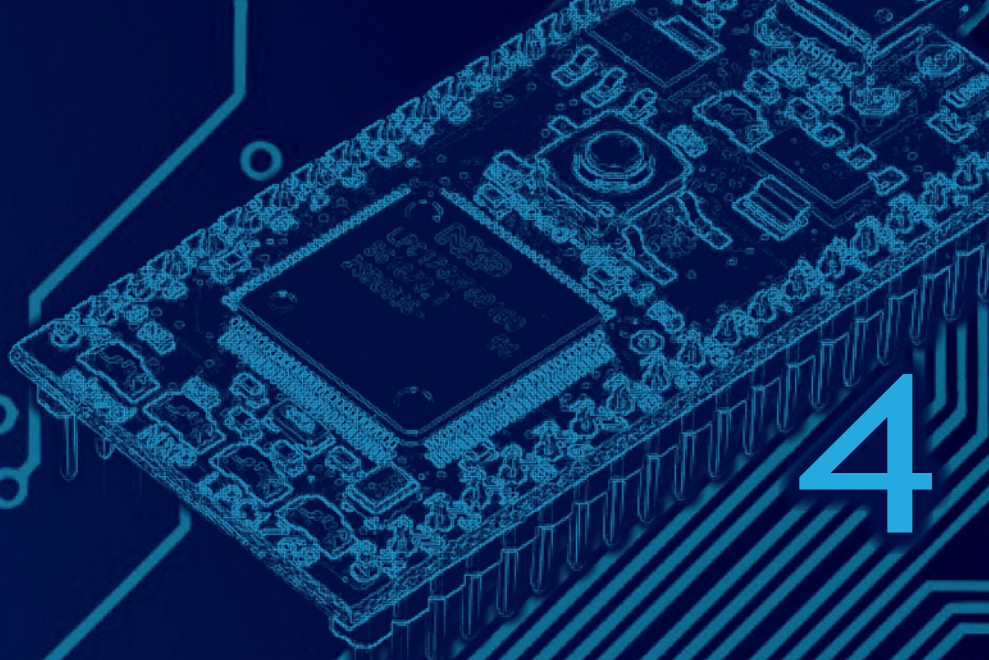
riutilizzo di materiale informatico
per scopi di utilità sociale, a cura di
Faber Libertatis

MERCANTE IN FIERA

mostra mercato dell'hobbista,
il luogo migliore per scambi tra
appassionati di radiantismo e
cine-foto-ottica

info: 0425 27401 - www.arearebus.com - fiera@arearebus.com

Realizziamo la nostra prima applicazione con una board mbed ready: la KL25Z prodotta da Freescale. Quarta ed ultima puntata.



4

Conoscere e usare

mbed

dell'ing. Vincenzo Mendola

Nell'ultimo anno abbiamo assistito alla nascita di tantissime nuove board embedded destinate al mondo hobbystico o comunque al mercato di fascia medio-bassa: sulla scia di Arduino e di Raspberry Pi hanno fatto la loro comparsa PcDuino, BeagleBoard, Cubieboard, Odroid, UDOO e tante altre ancora. Schede via via sempre più potenti, addirittura multiprocessore, ma che rimangono in quella fascia di prezzo (massimo 50-100 Euro) alla portata un po' di tutti, studenti, hobbysti e semplici appassionati. Un fiorire di nuovi prodotti che ha visto una convergenza tra il mondo Arduino e il sistema operati-

vo GNU/Linux all'insegna dell'open hardware e dell'open software. La maggior parte delle board presenta infatti la facilità della gestione dei GPIO e delle periferiche tipica di Arduino, mentre la connessione alla rete e quindi ad Internet e la gestione complessiva della board, è garantita da un sistema operativo basato su GNU/Linux. Questa è anche la logica progettuale delle nuove board immesse sul mercato o semplicemente annunciate dal team di Arduino. Parliamo, in primo luogo, di Arduino Yún, la prima scheda di casa Arduino che fa ricorso ad un sistema operativo GNU/Linux denominato Linino, ma anche di Arduino TRE che dovrebbe vedere la luce nella primavera dell'an-

Listato 1

```
#include "mbed.h"

int main() {

    DigitalOut ledG(LED_GREEN);

    while (true) {

        ledG = 1; // off
        wait(0.5);
        ledG = 0; // on
        wait(0.5);

    }

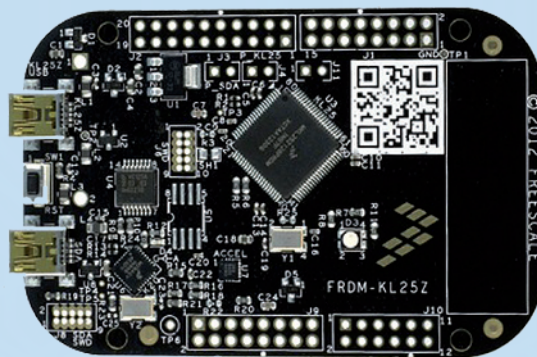
}
```

no prossimo e che utilizzerà un processore Sitara di Texas Instruments ed un sistema operativo GNU/Linux. Leggermente diversa è invece l'impostazione della board Intel Galileo che utilizza un processore x86 e nella quale lo sketch di Arduino "gira" all'interno del sistema GNU/Linux come un processo utente. Indubbiamente la comparsa e la disponibilità di una scheda a bassissimo costo basata su GNU/Linux come Raspberry Pi (ad oggi sono state vendute oltre due milioni di board) ha fatto capire che, il vantaggio di disporre on board di un sistema operativo flessibile (e gratuito!) come Linux sono di gran lunga superiori agli svantaggi che comporta una programmazione un po' più complessa. Tanto fervore con tante nuove schede ha anche avuto il benefico effetto di accentuare la concorrenza tra i produttori di semiconduttori, che si sono prodigati nell'immettere sul mercato prodotti sempre più performanti e sempre più economici. Tutto ciò si è tradotto in una riduzione dei prezzi delle board che ha generato una spirale positiva che ha fatto sì che anche le tecnologie più avanzate potessero risultare disponibili a prezzi incredibilmente bassi.

LA SCHEDA MBED READY KL25Z

Da questa corsa contro il tempo non resta fuori nemmeno il team mbed, che negli ultimi mesi ha introdotto, dopo anni di apparente staticità, numerose innovazioni, passando dalla riduzione generalizzata dei costi dell'hardware, all'aggiunta di nuovi modelli, non direttamente progettati dal team ufficiale, ma compatibili. Dunque, nell'universo mbed, prima conosciuto per i suoi modelli "storici" che ne hanno decretato il successo

Fig. 1



(parliamo della LPC1768 e della LPC1114, di cui al punto [1] dei riferimenti). Tra le nuove schede della community mbed, quella che ha attirato subito la nostra attenzione è stata la KL25Z sviluppata da Freescale (Fig. 1): venduta al pubblico ad un prezzo veramente concorrenziale (si parla di circa 12 euro) per un prodotto di questo tipo e

Listato 2

```
#include "mbed.h"

int main() {

    DigitalOut ledG(LED_GREEN);
    DigitalOut ledR(LED_RED);
    DigitalOut ledB(LED_BLUE);

    //ALL COLORS OFF
    ledR=1;
    ledG=1;
    ledB=1;

    while (true) {

        //GREEN

        ledG = 0; // on
        wait(0.25);

        ledG = 1; // off
        wait(0.25);

        //RED

        ledR = 0; // on
        wait(0.25);

        ledR = 1; // off
        wait(0.25);

        //BLUE

        ledB = 0; // on
        wait(0.25);

        ledB = 1; // off
        wait(0.25);

    }

}
```

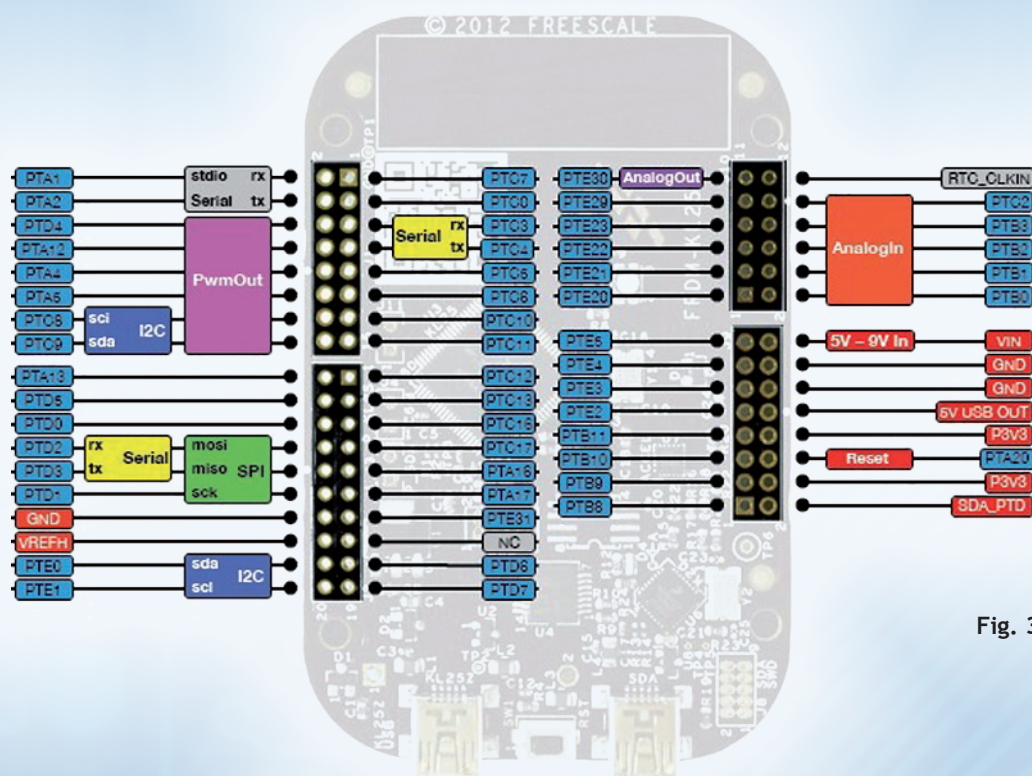



Fig. 3

con queste caratteristiche, la KL25Z è sostanzialmente una board “mbed ready” (così definita per via della possibilità di utilizzare le librerie sviluppate per le board mbed tramite un semplice upload attraverso la presa USB, come dettagliato sul sito ufficiale) dalle interessantissime e peculiari caratteristiche, le quali la differenziano da quelle delle due versioni presentate nelle precedenti puntate. La scheda di Freescale è facilmente upgradabile ad mbed, semplicemente aggiornandone il firmware, come dettagliato nella pagina descrittiva della board di cui al punto [2] dei Riferimenti. Una volta fatto l’aggiornamento, il sistema ci confermerà che la procedura è andata correttamente a termine (con un messaggio come quello riportato nella Fig. 2). La prima cosa che salta subito all’occhio è il form factor di questa nuova board, che è compatibile con quello delle board Arduino; inoltre la KL25Z di Freescale dispone di pad forati sui lati come Arduino, quindi in pratica, basta aggiungere gli header e si possono montare su di esso gli stessi shield facendo

però attenzione al fatto che gli mbed lavorano con logica a 3,3V. Questa caratteristica costituisce un vantaggio importante, in quanto non si può negare che la vastità di shield ed applicazioni disponibili per Arduino è molto allettante e che uno degli elementi che uno sperimentatore spesso valuta prima dell'acquisto di una scheda

Listato 3

```
#include "mbed.h"

BusOut RGB(PTB18, PTB19, PTD1);

int main() {

//ALL COLORS OFF

RGB=0x07;

while (true) {

    RGB=0x06;
    wait(0.25);
    RGB=0x05;
    wait(0.25);
    RGB=0x03;
    wait(0.25);
}

}
```

✔ Device 'kl25z' connected and added to your account.

Fig. 2

Listato 4

```
#include "mbed.h"

DigitalOut ledR(LED_RED);
Serial pc(USBTX, USBRX);

int main() {
    int i=1;
    pc.printf("Hello World!\r\n");

    while (true) {
        wait(1);
        pc.printf("%d\r\n", i);
        i++;
        ledR = !ledR;
    }
}
```

Listato 5

```
#include "mbed.h"
#include "MMA8451Q.h"

#define MMA8451_I2C_ADDRESS (0x1d<<1)

int main(void) {
    MMA8451Q acc(PTE25, PTE24, MMA8451_I2C_
ADDRESS);
    PwmOut rled(LED_RED);
    PwmOut gled(LED_GREEN);
    PwmOut bled(LED_BLUE);

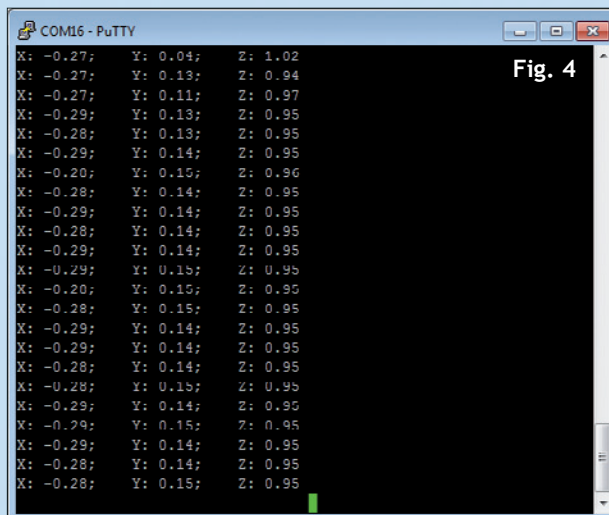
    while (true) {
        rled = 1.0 - abs(acc.getAccX());
        gled = 1.0 - abs(acc.getAccY());
        bled = 1.0 - abs(acc.getAccZ());

        printf("X: %1.2f;   ", acc.getAccX());
        printf("Y: %1.2f;   ", acc.getAccY());
        printf("Z: %1.2f \n\r", acc.getAccZ());
        wait(0.1);
    }
}
```

di prototipazione è la possibilità di utilizzare questo "bagaglio". Ecco perché la compatibilità con il mondo Arduino è attualmente (a ragione o meno) un fattore tenuto molto in considerazione da chi realizza nuove schede di sviluppo e prototipazione.

Vediamo dunque le caratteristiche della scheda Freescale (reperibili sul web alla pagina indicata al punto [3] dei Riferimenti): il cuore della KL25Z è un processore ARM® Cortex™-M0+ Core operante con un clock di 48MHz, con 16 kB RAM e 128 kB di Flash; oltre alle periferiche standard per questa tipologia di tool (2 bus SPI, 2 I²C-Bus, 3 UART, 6 moduli PWM, 6 ADC, vari GPIO), saltano all'occhio delle funzionalità avanzate e di sicuro interesse: a bordo si trovano un accelerometro a tre assi (si tratta di un MMA8451Q della Freescale), un LED RGB controllato in PWM e un touch sensor di tipo capacitivo.

Prima di proseguire è d'obbligo un accenno



alle specifiche elettriche: la tensione di alimentazione della board è compresa tra 5 e 9 volt (in continua), con un assorbimento massimo dichiarato di 100 mA; un regolatore sulla scheda riduce la tensione adattandola a quella richiesta dalla logica a 3,3 V, che caratterizza da tempo la maggior parte dei dispositivi simili al Freescale. Ciascun terminale di I/O della board è in grado di erogare 4 mA. In Fig. 3 è visibile il dettaglio del pinout del KL25Z.

Proviamo a prendere confidenza con la KL25Z partendo da alcuni semplici esempi applicativi. Il primo riguarda la gestione del LED RGB disponibile sul PCB, che otteniamo con un semplice esempio che ci permette di far lampeggiare il solo colore verde: lo trovate nel Listato 1. Se guardate lo schema elettrico del KL25Z (punto [4] dei Riferimenti) noterete che il LED RGB sulla scheda è di tipo ad anodo comune, quindi per pilotarlo è necessario fornire un livello basso al singolo LED da accendere; in questo caso è stato scelto il verde (LED_GREEN). Come vedete, il codice è veramente semplice ed auto esplicativo.

Vediamo ora, nel Listato 2, come far lampeggiare tutti e tre i colori che il LED montato sulla scheda mbed può produrre.

L'applicazione proposta non è affatto complessa; è solo un esempio di base.

La stessa cosa si sarebbe potuta ottenere servendosi delle istruzioni contenute nel Listato 3, dedotte osservando in dettaglio lo schema elettrico; guardare lo schema è una cosa che consigliamo sempre di fare, perché vi permette di capire maggiormente le caratteristiche dell'hardware che stiamo utilizzando, permettendoci di sfruttarne al meglio le caratteristiche.

A questo punto del corso riteniamo non sia più necessario spiegare le istruzioni riga per riga,

in quanto dovrete ormai avere il background che vi permette di comprenderle. Ad ogni modo, per chi volesse approfondire, può trovare ulteriori dettagli inerenti a "BusOut" visitando la pagina web <http://mbed.org/handbook/BusOut>.

Bene, detto ciò, alla nostra applicazione andiamo ad aggiungere la comunicazione con la seriale, la quale viene implementata grazie al codice che trovate nel **Listato 4**.

Passiamo quindi ad un esempio che combina l'uso dell'accelerometro con quello del LED RGB: il valore dell'accelerazione sui 3 assi è visibile sia in maniera numerica tramite la seriale del nostro mbed (**Fig. 4**, sia direttamente tramite il colore del LED sulla scheda, che varia in base al valore misurato. Nel **Listato 5** trovate il codice che permette di ottenere quanto appena detto. Il team di mbed ha creato una libreria per utilizzare in maniera immediata lo slide presente sulla board. Vediamo due semplici applicazioni che lo riguardano, il primo dei quali si basa sul codice che trovate nel **Listato 6**. Il secondo esempio si basa invece sul codice descritto nel **Listato 7**.

Nel primo caso, il LED mantiene visibile il livello di luminosità selezionato tramite lo slide, mentre il codice del **Listato 7** lo accende solo quando toccherete lo slide e la luminosità varierà fintanto che farete scorrere il dito. Per finire la lezione di quest'ultima puntata del corso, ecco un esempio prettamente "scenografico" tratto dalla pagina ufficiale del KL25Z (punto [5] dei Riferimenti) nella quale potrete trovare altri esempi applicativi. Il codice corrispondente da caricare in mbed è quello contenuto nel **Listato 8**.

Bene, ora che avete appreso i rudimenti per lavorare in maniera proficua con le schede mbed, potete proseguire da soli nell'uso e arricchire il vostro bagaglio di applicazioni in base alle vostre esigenze, ricordando che potrete sempre

Listato 6

```
#include "mbed.h"
#include "TSSensor.h"

int main(void) {
    PwmOut led(LED_BLUE);
    TSSensor tsi;

    while (true) {
        while(tsi.readPercentage()>0){
            led = 1.0 - tsi.
readPercentage();
            wait(0.1);}
    }
}
```

Listato 7

```
#include "mbed.h"
#include "TSSensor.h"

int main(void) {
    PwmOut led(LED_RED);
    TSSensor tsi;

    while (true) {

        led = 1.0 - tsi.readPercentage();
        wait(0.1);
    }
}
```

Listato 8

```
#include "mbed.h"

PwmOut r (LED_RED);
PwmOut g (LED_GREEN);
PwmOut b (LED_BLUE);

int main() {
    r.period(0.001);
    g.period(0.001);
    b.period(0.001);

    while (true) {
        for (float i = 0.0; i < 1.0 ; i += 0.001) {
            float p = 3 * i;
            r=1.0 - ((p<1.0) ? 1.0 - p : (p>2.0) ? p - 2.0 : 0.0);
            g = 1.0 - ((p<1.0) ? p : (p>2.0) ? 0.0 : 2.0 - p);
            b = 1.0 - ((p<1.0) ? 0.0 : (p>2.0) ? 3.0 - p : p - 1.0);
            wait (0.0025);
        }
    }
}
```

contare sull'aiuto dell'handbook, del cookbook e del forum dedicato al mondo mbed.

Con questo corso abbiamo concluso; torneremo sull'argomento quando la comunità di sviluppatori mbed sfornerà novità rilevanti. Comunque non mancheremo di pubblicare, in futuro, ulteriori approfondimenti che vi permettano di tenervi aggiornati sul mondo degli mbed che, per quanto viva un percorso diverso e lontano dalle "luci della ribalta" come Arduino o Raspberry Pi, è ugualmente interessante e meritevole di attenzione.

Riferimenti

- [1] <http://mbed.org/platforms/>
- [2] <http://mbed.org/platforms/KL25Z/>
- [3] <http://mbed.org/handbook/mbed-FRDM-KL25Z>
- [4] http://cache.freescale.com/files/soft_dev_tools/hardware_tools/schematics/FRDM-KL25Z_SCH.pdf?fp=1
- [5] <http://mbed.org/handbook/mbed-FRDM-KL25Z-Examples>



Visualizza l'orario nel formato mezz'ora e minuti, mediante due cerchi che riportano 24 LED per le ore e 60 per i minuti; il tutto funziona con logica cablata la cui frequenza di lavoro è scandita da un quarzo. Può essere alimentato a batterie o con un adattatore da rete.

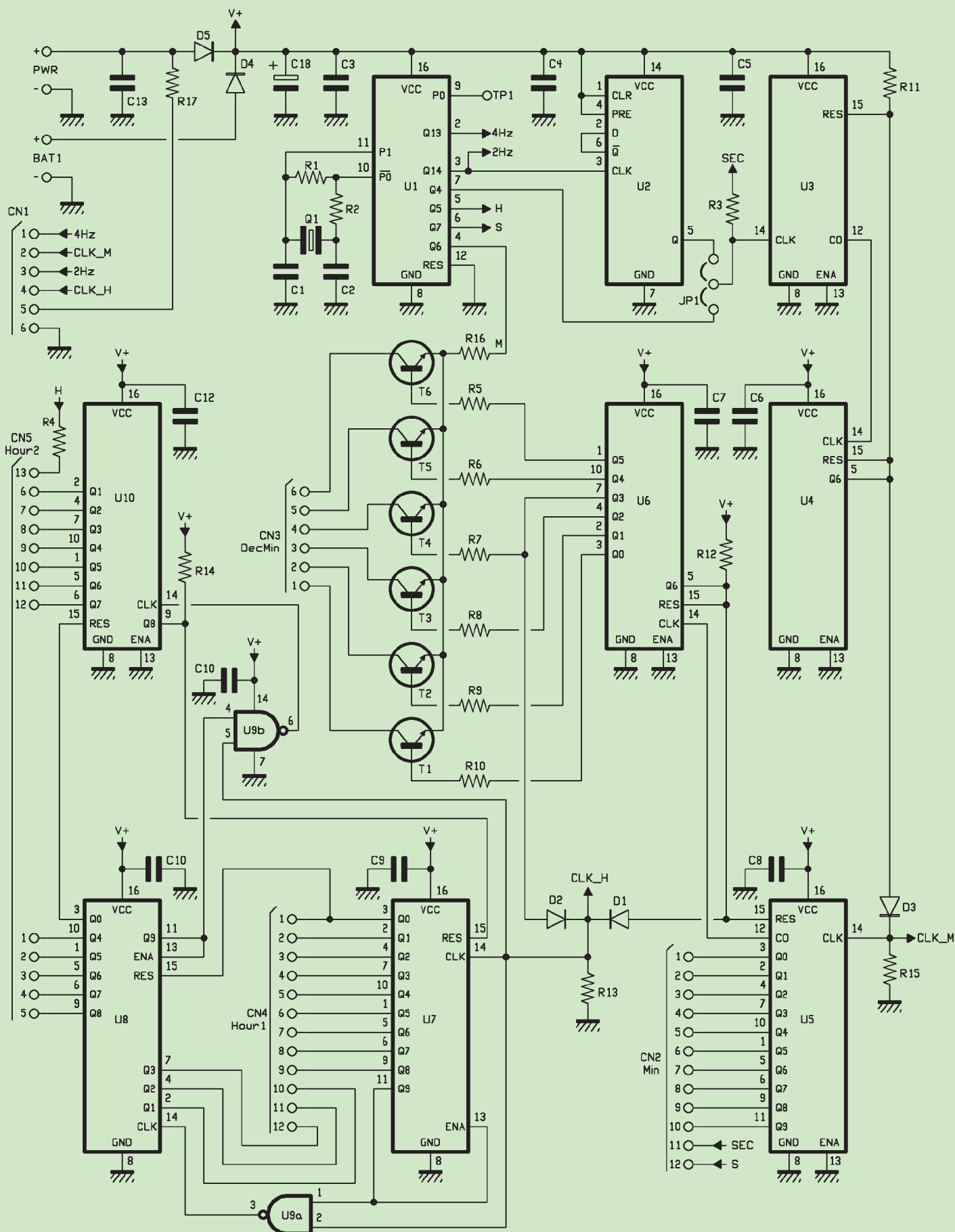


OROLed, UN OROLOGIO A LED A LETTURA ANALOGICA

..... del Prof. MICHELE MENNITI

I LED (Light Emitting Diode) ha avuto una grande importanza nell'elettronica, sin da quando venne utilizzato come semplice lampadina spia. La sua evoluzione oggi lo ha portato ad essere usato dappertutto: domotica, illuminazione industriale e auto-motociclistica, per non parlare dell'infinito numero di piccoli oggetti utili o di semplici gadget. Per rinnovare l'importanza di questo componente, abbiamo pensato di proporvi un complemento d'arredo di stile moderno, uno di quegli oggetti particolari che stanno bene dap-

pertutto e che risultano anche utili. Un orologio a LED da parete non è certo l'ultima novità nel campo dell'elettronica, ma il nostro progetto ha una caratteristica che lo rende sicuramente diverso da quanto solitamente si vede in giro o sulle pagine di questa stessa rivista: la lettura analogica. Gli orologi da parete di tipo analogico sono tutti costruiti col sistema delle tre lancette (ore, minuti, secondi che ruotano a 360°) dalla cui posizione si evince l'orario di massima, anche se non si dispone dei relativi numeri di riferimento da 1 a 12 (per le ore) e delle



tacche dei minuti. Per ottenere un vero effetto analogico abbiamo adottato una tecnica che consiste nel mostrare anche le mezze ore, perché senza di esse la lettura sarebbe stata piuttosto problematica. Infatti come è noto, man mano che la lancetta dei minuti procede, anche quella delle ore si sposta proporzionalmente, per cui se alle 12.00 la lancetta delle ore si trova esattamente sul 12, alle 12.35 essa si trova poco dopo la metà dello spazio tra 12 e 1. Se avessimo adottato solo dodici LED per le ore, nella seconda parte dell'ora la lettura analogica sarebbe risultata falsata. Provate ad immaginare che alle 12.35 la lancetta delle ore si trovi esattamente sul 12: dal punto di vista analogico in realtà si sarebbe più propensi a credere che siano le 11.35.

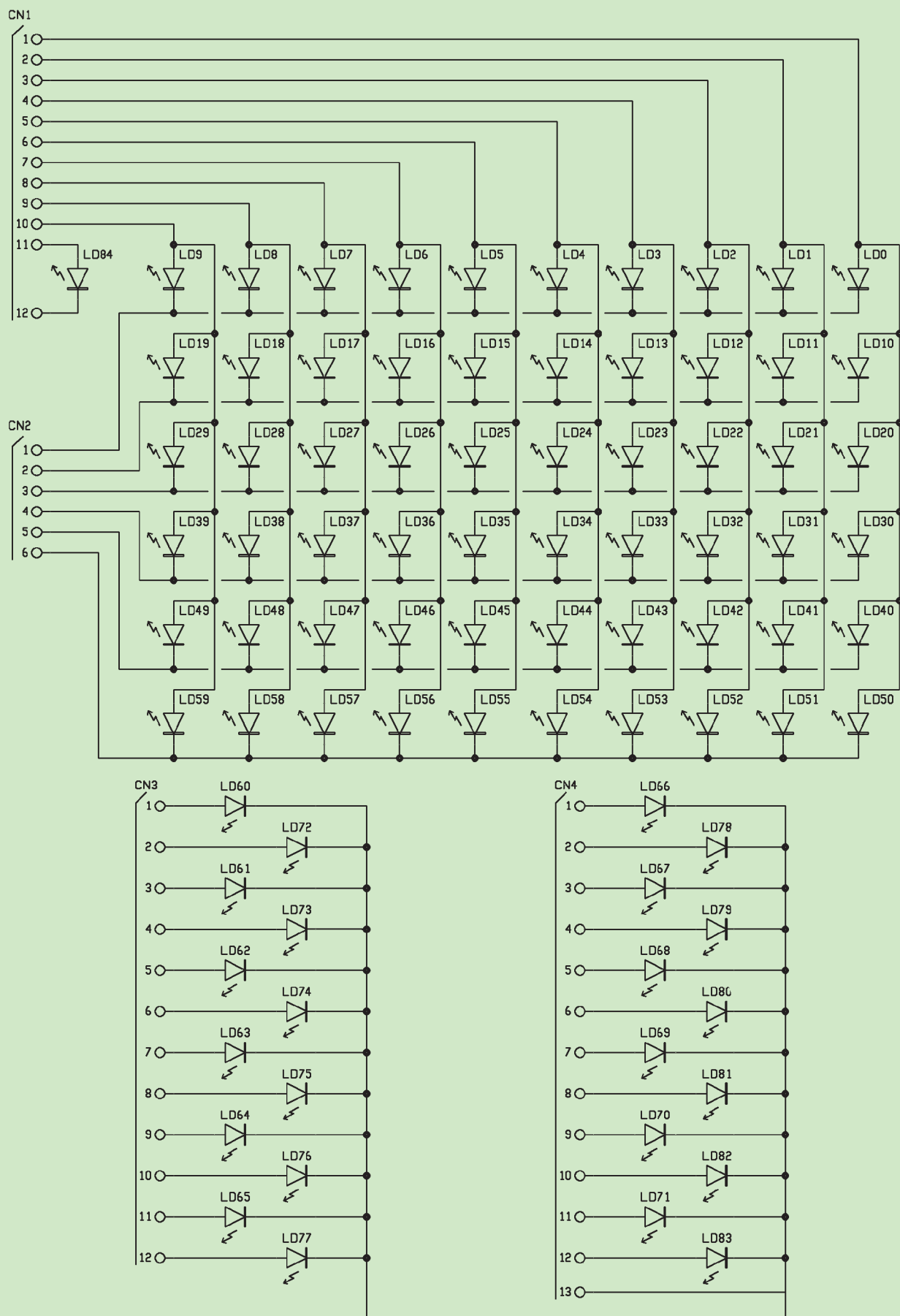
Il circuito è stato realizzato senza ricorrere all'uso di microcontrollori, ma usando solo logica discreta in tecnologia CMOS HC. Inoltre prevede la possibilità di essere alimentato sia con la rete, sia con una batteria messa in carica da un apposito alimentatore/caricabatteria integrato, da rete.

SCHEMA ELETTRICO

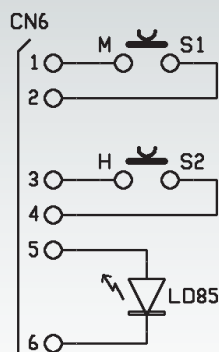
Prima di descrivere lo schema elettrico, diciamo che l'orologio è composto in realtà da tre circuiti, che sono quello base (chiamato **ORO logic**, contenente la logica di controllo) ed un quadrante (**LED Panel**) che ospita i LED cui spetta il compito di visualizzare l'ora; i due sono uniti elettricamente da quattro connettori SIL (montati ognuno su una delle schede, disposti a quadrilatero) e andranno uniti meccanicamente in fase di montaggio. Completa l'insieme, una scheda che ospita i pulsanti di regolazione, da collegare alla scheda della logica mediante un pin-strip.

Partiamo con il circuito che contiene la logica, il quale conta ben 11 integrati; per quanto possa apparire complesso, lo è meno di quanto sembri. Lo stadio iniziale del circuito si basa su un 74HC4060 (U1) che contiene un oscillatore (necessita di un quarzo, due resistenze e due condensatori esterni) e una serie di stadi divisori di frequenza. Il quarzo è il classico modello da orologi, ossia un piccolo cilindro che oscilla a 32,768 kHz; la frequenza dello stadio è regolabile mediante il compensatore C2, vedremo più avanti la semplice operazione di taratura. Questo integrato ha ben undici uscite: la prima (pin 9) riporta la frequenza di oscillazione a logica invertita, mentre le altre dieci riportano successive suddivisioni. Per i nostri scopi ne abbiamo usate sei, ma al momento ci interessa solo quella relativa al pin 3 (2 Hz). La frequenza corrispondente necessita di una ulteriore suddivisione x2, al fine di ottenere 1 Hz, che diventa il clock del nostro ORO-Led. Questo compito viene svolto da U2 (74HC74), un doppio flip-flop tipo D, di cui abbiamo usato un solo stadio nella classica configurazione di divisore x2. Sul suo pin 5 otteniamo appunto la frequenza di 1Hz, che viene applicata ad un pin laterale di un jumper a due vie (JP1); lo status normale di questo jumper porta tale frequenza al pin 14 di U3 (74HC4017). Questo integrato è un divisore a dieci uscite: in pratica, ad ogni impulso di clock sul suo pin 14, commuta su HIGH, sequenzialmente, una delle sue dieci uscite, lasciando tutte le altre su LOW; a fine conteggio il pin 12 (Cout) va su HIGH contestualmente alla prima uscita e si ricomincia. Ciò significa che il pin 12 emette un impulso HIGH ogni dieci secondi, che viene

inviato direttamente al successivo 74HC4017, il quale serve per il conteggio delle decine di secondi; infatti ogni uscita resta su H per dieci secondi. Appena arriva il nuovo impulso sul pin 14 di U4 (proveniente, come detto, dal pin 12 di U3), esso cambia uscita; dovendo fornire a sua volta un impulso in uscita ad intervalli di un minuto, U4 è settato come contatore x6, infatti la sua settima uscita (pin 5) è quella che fornisce l'impulso H al pin 14 del successivo 74HC4017 (U5); tale impulso, oltre a far partire il conteggio dei minuti, resetta U3 e U4, cioè il conteggio dei secondi e delle decine di secondi, essendo collegato ai rispettivi RESET (pin 15). Il diodo D3 serve per separare l'effetto pull-up di R11 da quello di pull-down di R15. Ogni 10 minuti U5 fornisce un impulso di conteggio al pin 14 del successivo 74HC4017 (U6), che è settato come contatore x3 e x6; infatti sia la quarta uscita (pin 7) che la settima (pin 5) forniscono un impulso H al pin 14 del successivo 74HC4017 (U7), o meglio, del gruppo di tre 74HC4017 costituito da U7, U8 e U10 che, nel loro insieme, e con le due porte AND aggiuntive, appartenenti a U9 (74HC08) costituiscono un contatore x24. Tornando a U6, c'è da notare che solo la settima uscita genera un effetto reset (sia per U5 che per U6); infatti la quarta uscita serve per il conteggio delle mezze ore, quindi il totale orario dei minuti deve continuare il proprio conteggio, e l'azzeramento deve avvenire solo ogni ora, quindi quando si attiva la settima uscita. Il gruppo U7, U8, U9 e U10 riceve un impulso ogni 30 minuti, attivando in sequenza le ventiquattro uscite dei tre integrati (nove di U7, otto di U8 e sette di U10, come chiaramente visibile



[schema elettrico **PULSANTIERA**]



nello schema elettrico); il 25° impulso provoca l'azzeramento dell'intero gruppo: ciò significa che sono trascorse esattamente 12 ore. A tutti gli integrati del circuito è stato applicato, vicinissimo al pin di alimentazione positiva, un condensatore da 100 nF (poliestere o ceramico) con la duplice funzione di evitare auto-oscillazioni e filtrare disturbi eventualmente presenti sull'alimentazione (C3÷C11).

Tutto il circuito descritto finora ha lo scopo di pilotare gli 85 LED presenti sul PCB del pannello di visualizzazione (LED Panel). Il LED 84 lampeggia al ritmo di 1 secondo (mezzo secondo spento, mezzo secondo acceso), infatti il suo anodo è collegato, tramite la resistenza di limitazione R3, al pin 5 di U2A (uscita 1Hz) che, avendo un duty-cycle del 50%, fornisce una semionda positiva per 0,5 secondi ed una semionda negativa per altri 0,5 secondi; il catodo invece è collegato al pin 6 di U1, che fornisce costantemente una frequenza di 256 Hz. Questo piccolo stratagemma permette un notevole risparmio di corrente; infatti, nella realtà, mentre durante la semionda negativa sul pin 5 di U2A il LED 84 è effettivamente spento, durante la semionda positiva esso lampeggia ad una frequenza di 256 Hz, quindi per metà del tempo esso

è spento, ma l'effetto naturale della persistenza ottica dell'occhio umano, fa sì che esso sembri sempre acceso. In pratica il LED alterna una fase di spento ad una fase velocissima di acceso/spento, invisibile all'occhio umano, ma che permette di risparmiare sul consumo di corrente.

I 60 LED dei minuti vengono accesi con la tecnica del multiplex: i contatti 1÷10 del connettore CN2 vengono portati uno alla volta a livello alto; ciascuno di essi, connesso ad un'uscita del contatore dei minuti U5, raggiunge (mediante il connettore CN1 del LED Panel) gli anodi di un gruppo di 6 LED. Quindi se U5 ha attiva l'uscita Q0 (pin 3), mediante il pin 1 di CN2, i LED dei minuti 0, 10, 20, 30, 40, 50 ricevono tutti contemporaneamente il livello H sul proprio anodo, ma uno solo di essi si accenderà effettivamente. Infatti i catodi di questi 6 LED non sono collegati ciascuno ad un contatto del connettore CN2 del pannello, che porta al CN3 del circuito logico. A tale connettore sono collegati i collettori di sei transistor NPN (TR1÷TR6) che vanno in conduzione sequenzialmente, uno per volta, quando la rispettiva base viene a trovarsi ad una tensione maggiore di 0,7V, mediante il livello H proveniente da U6 e limitato in corrente dalle resistenze R5÷R10. Quando un

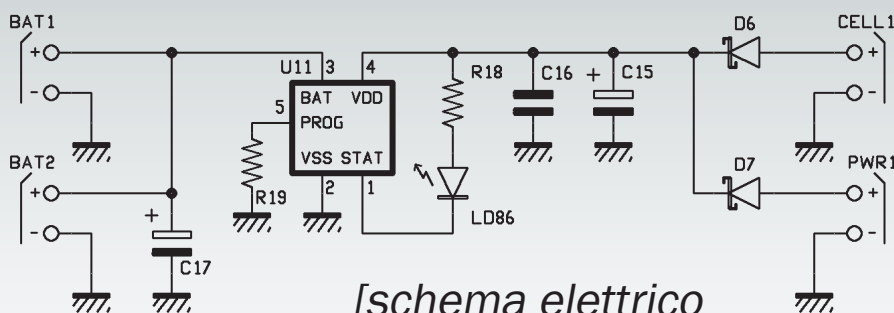
transistor va in conduzione, un intero gruppo di 10 catodi viene collegato, mediante la resistenza di limitazione R16, al pin 4 di U1, il quale pulsa ad una frequenza fissa di 512 Hz (sempre per risparmiare elettricità) ma si accende solo quello il cui anodo è allo stato H.

Con questo sistema, ossia pilotando con le uscite di U5 gli anodi di dieci gruppi di sei LED e con le uscite di U6 i catodi di sei gruppi di dieci LED, si accenderà un LED alla volta, per 60 secondi esatti.

I ventiquattro LED vengono pilotati sequenzialmente dal gruppo di uscite di U7, U8 e U9, ogni pin pilota l'anodo di un singolo LED, mentre tutti i catodi sono collegati in parallelo e, mediante la resistenza di limitazione R4, vanno al pin 5 di U1, sul quale è presente una frequenza fissa di 1024 Hz; anche qui, far pulsare i LED contribuisce al risparmio energetico. Questi 24 LED si accendono ad intervalli esatti di 30 minuti, compiendo perciò un ciclo di 12 ore.

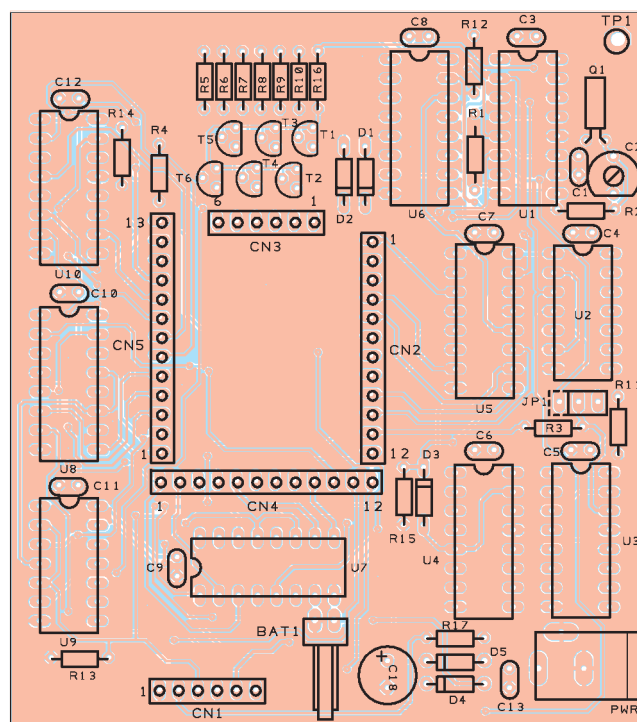
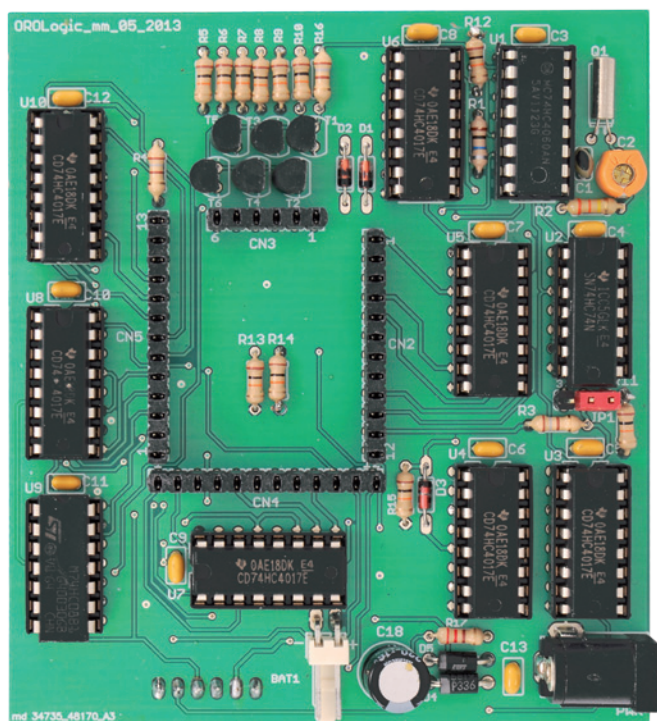
FUNZIONAMENTO DELL'OROLed

Vediamo ora il funzionamento di base dell'OROLed, sempre facendo riferimento allo schema elettrico. Alla prima accensione le resistenze di pull-up R11, R12, R14 forniscono una tensione



[schema elettrico **CARICABATTERIA**]

[piano di montaggio della **SCHEDA BASE**]



Elenco Componenti:

R1: 10 Mohm	T1÷T6: BC547
R2: 330kohm	U1: 74HC4060
R3, R4, R16: 180 ohm	U2: 74HC74
R5÷R15: 10 kohm	U3÷U8: 74HC4017
R17: 220 ohm	U9: 74HC08
R18: 470 ohm	U10: 74HC4017
R19: 2,2 kohm	U11: MCP73831T-2 (SOT23-5L)
C1: 10 pF ceramico	D1÷D3: 1N4148
C2: Compensatore 7-50pF	D4÷D7: 1N5819
C3÷C13: 100 nF ceramico	LD85: LED 5mm verde
C15: 4,7 µF 100 VL elettrolitico	LD86: LED 5mm rosso
C16: 100 nF ceramico	Q1: Quarzo 32768Hz
C17: 4,7 µF 100 VL elettrolitico	S1: Microswitch
C18: 220 µF 25 VL elettrolitico	S2: Microswitch

Varie:

- Jumper
- Strip maschio 3 vie
- Strip maschio 6 vie
- Strip femmina 6 vie (2 pz.)
- Strip femmina 12 vie (2 pz.)
- Strip femmina 13 vie
- Zoccolo 7+7 (2 pz.)
- Zoccolo 8+8 (8 pz.)
- Connettore 2 poli (JST2)
- Connettore molex 2 poli 2.54 mm
- Plug alimentazione
- Circuito stampato

positiva ai pin 15 dei contatori 74HC4017, resettando questi ultimi, quindi ognuno di questi integrati porterà il proprio pin 3 su H e tutti gli altri pin di uscita su L; non dimentichiamo che U7, U8 e U9 si comportano come un unico contatore con 24 uscite, perciò solo il pin 3 del primo integrato è coinvolto in questo meccanismo di reset. Questa condizione iniziale si traduce nell'accensione dei LED 60 e 0, che indicano le

ore 12.00 esatte; il lampeggio del LED 84 scandisce i secondi; allo scadere del sessantesimo secondo, il LED 0 si spegne e si accende il successivo LED 1. Questa sequenza continua senza altre variazioni fino al LED 29; nel momento in cui si accenderà il LED 30, significa che è passata una mezz'ora, allora si spegne il LED 60 e si accende il LED 72. L'accensione dei LED 72 e 30 indica le ore 12.30. A quel punto

continua la sequenza di accensione dei LED da 31 a 59; quando si accende il LED 0 si spegne il LED 72 e si accende il LED 61. La coppia di LED 61 e 0 indica l'ora 13.00, e così via. Quando inizia il secondo ciclo, la coppia di LED 60 e 0 indicherà la mezzanotte esatta. Quindi in ogni istante saranno accesi fissi sempre due LED, uno per ore/mezz'ora e l'altro per i minuti, oltre al LED dei secondi che lampeggerà in

continuazione. In **Fig. 1** vedete la combinazione di LED che indica l'orario 10:24.

Per regolare l'orario, abbiamo previsto due pulsanti, montati su una scheda separata (**Fig. 2**) che servono per agire separatamente sulle ore (S1) e sui minuti (S2); la regolazione avviene applicando una frequenza di 4 Hz all'ingresso di U5 (contatore dei minuti) o di 2 Hz all'ingresso di U7 (contatore di ore e mezz'ora).

E veniamo all'alimentazione: i consumi relativamente bassi dei tre LED coinvolti nell'indicazione dell'orario permettono di ricorrere, temporaneamente o stabilmente, all'uso delle batterie. L'alimentazione può essere fornita in due modi: da un alimentatore da rete, prevedendo una batteria tampone, oppure solo a batteria. Nel primo caso bisogna montare il jack PWR previsto sul PCB principale della logica (ORO logic) oltre a C13, C18, D4, D5, R17 e LD85 (quest'ultimo va sulla schedina dei due pulsanti di regolazione); alla presa BAT1 andranno collegate 3 batterie alcaline da 1,5V in serie, in modo da garantire una tensione di 4,5V. Con questa combinazione l'OROLed lavora sempre con alimentazione fissa di 5V stabilizzata (bastano 100mA) applicata al jack PWR; quando essa viene a mancare, le tre batterie tampone continueranno ad alimentare l'OROLed garantendo svariate ore di autonomia.

Si può anche ricorrere all'alimentazione a batteria con ricarica automatica o manuale: in questo caso si montano solo il C18, D4 e la presa BAT1, mentre bisogna montare la schedina esterna LiPo. Questa scheda si può alimentare mediante una fonte di alimentazione esterna, che può essere garantita da una idonea batteria di celle solari o da un semplice

alimentatore stabilizzato da 5V (bastano 800mA); la fonte di alimentazione viene applicata all'integrato U11 (MCP73831T-2) che è un eccellente regolatore per caricare batterie LiPo. La batteria LiPo (3,7V - 2Ah o maggiore) viene caricata e contemporaneamente viene alimentato il circuito; se tale fonte esterna viene meno, sarà la batteria LiPo che si occuperà di alimentare l'intero circuito.

Riguardo alla scheda LiPo, bisogna notare che dell'integrato U11 esistono quattro versioni diverse che differiscono per la tensione di carica della batteria; la versione che serve per le LiPo è appunto MCP73831T-2, la quale carica

la batteria a 4,23 V. Usando le altre versioni si rischia di non caricare la batteria a sufficienza o di danneggiarla.

L'autonomia complessiva garantita dalla batteria dipende dalla capacità di quest'ultima: nei nostri test abbiamo usato una 2.000 mAh e con essa l'OROLed ha lavorato senza problemi per circa quindici giorni; la durata



Fig. 2

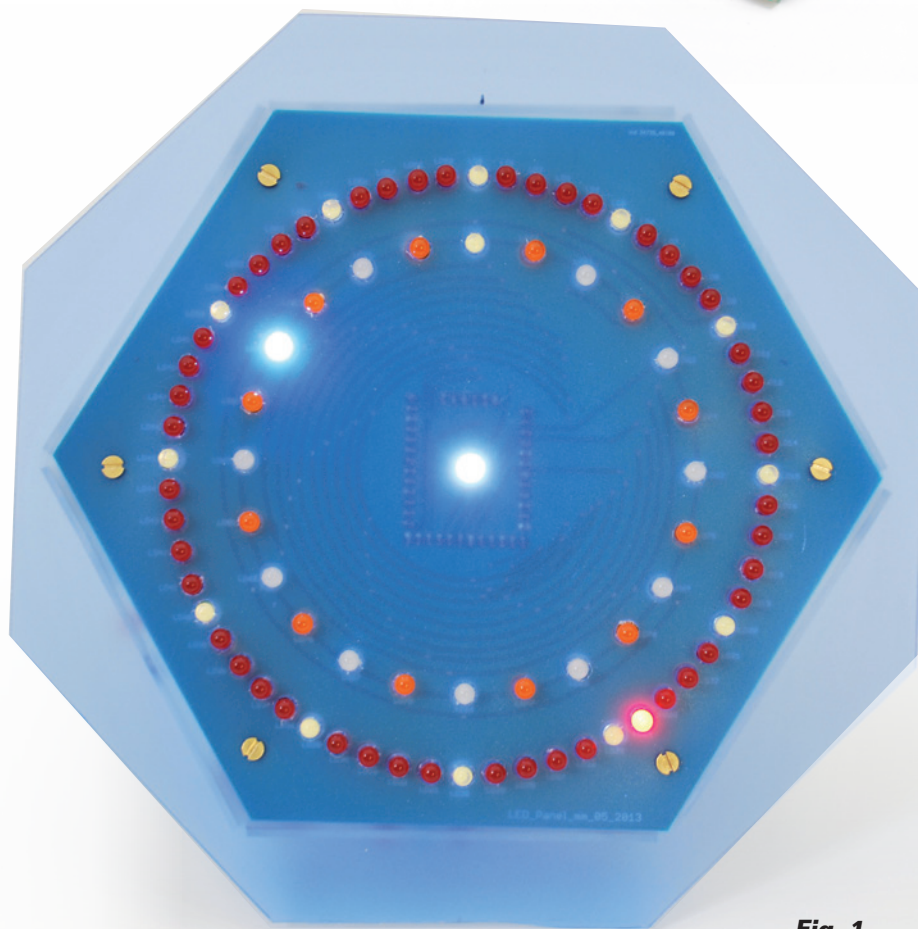
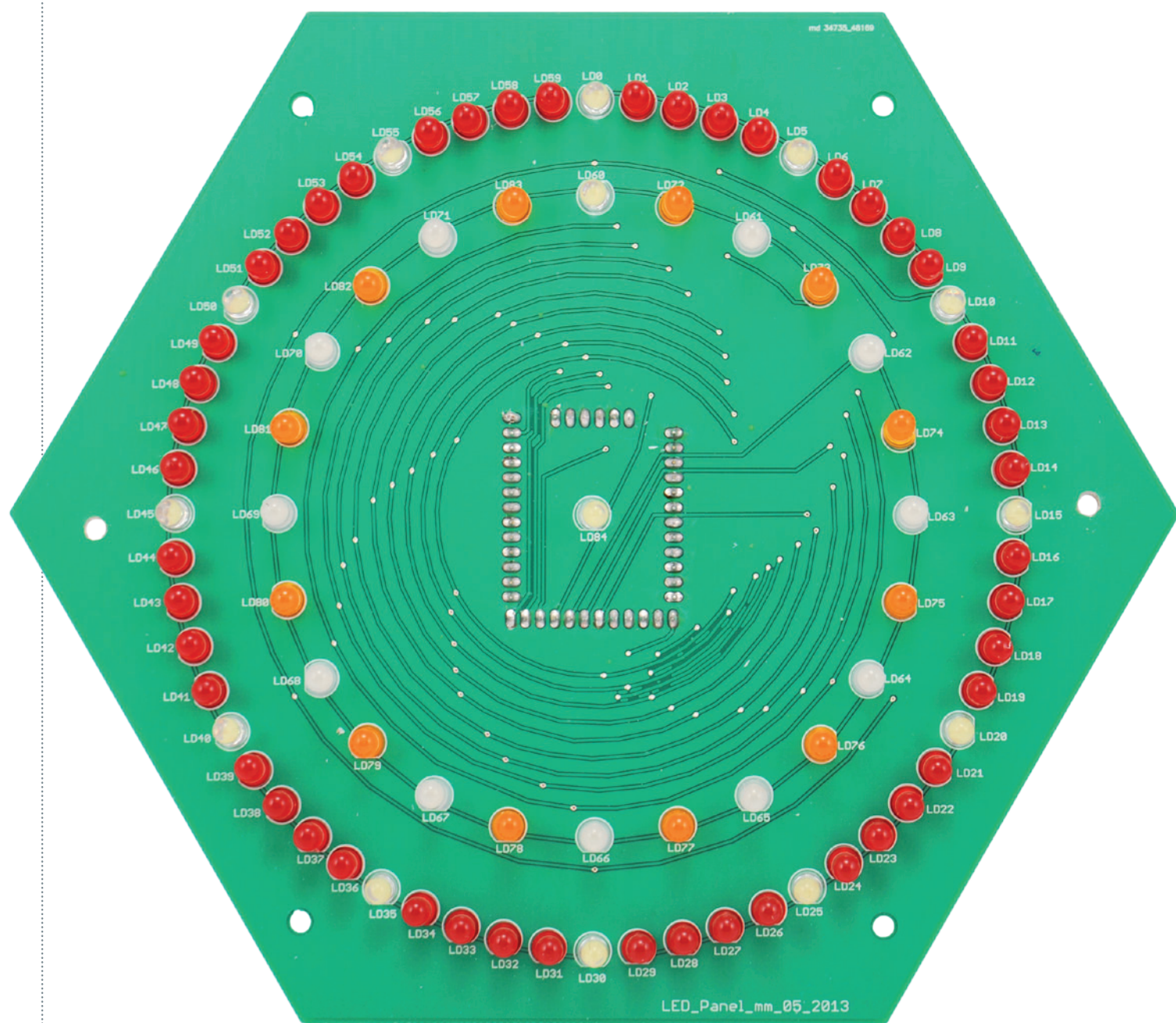


Fig. 1

[piano di montaggio del **LED PANEL**]



Elenco Componenti:

LD0: LED 5 mm bianco
LD1÷LD4: LED 5 mm rosso
LD5: LED 5 mm bianco
LD6÷LD9: LED 5 mm rosso
LD10: LED 5 mm bianco
LD11÷LD14: LED 5 mm rosso

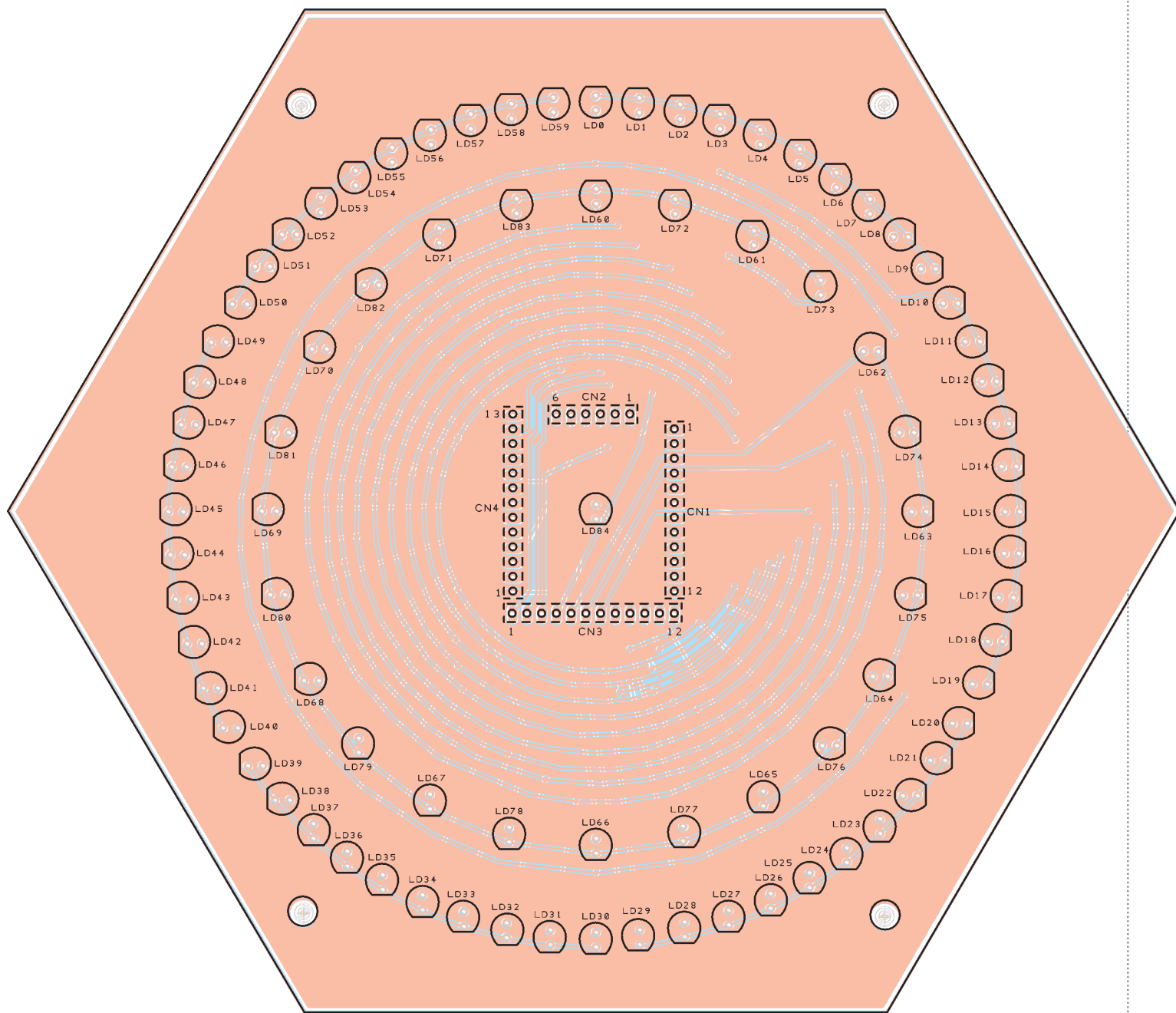
LD15: LED 5 mm bianco
LD16÷LD19: LED 5 mm rosso
LD20: LED 5 mm bianco
LD21÷LD24: LED 5 mm rosso
LD25: LED 5 mm bianco
LD26÷LD29: LED 5 mm rosso

LD30: LED 5 mm bianco
LD31÷LD34: LED 5 mm rosso
LD35: LED 5 mm bianco
LD36÷LD39: LED 5 mm rosso
LD40: LED 5 mm bianco
LD41÷LD44: LED 5 mm rosso

della ricarica dipende dalla resistenza R19, da noi calcolata per garantire una corrente massima

di 400 mA circa, corrente che deve essere però erogata anche dall'alimentatore stabilizzato

esterno. Il vantaggio dell'aver previsto un sistema specifico di caricabatterie LiPo consiste nel



LD45: LED 5 mm bianco
 LD46÷LD49: LED 5 mm rosso
 LD50: LED 5 mm bianco
 LD51÷LD54: LED 5 mm rosso
 LD55: LED 5 mm bianco
 LD56÷LD59: LED 5 mm rosso

LD60÷LD71: LED 5 mm bianco
 LD72÷LD83: LED 5 mm arancione
 LD84: LED 5 mm bianco

Varie:
 - Strip maschio 6 vie

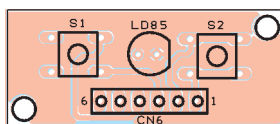
- Strip maschio 12 vie (2 pz.)
 - Strip maschio 13 vie
 - Circuito stampato

fatto che appena si vedono i LED affievolirsi è possibile inserire un comune alimentatore nella

presa jack PWR1; l'OROLed non si fermerà e la batteria inizierà a caricarsi, operazione indicata

dall'accensione del LED 86, che si spegnerà solo a carica ultimata (a quel punto si potrà staccare

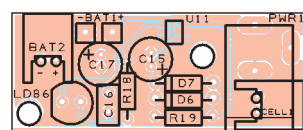
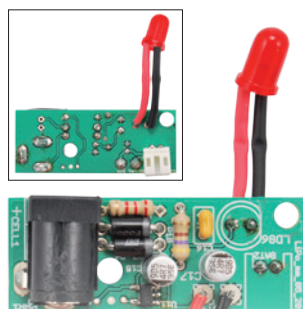
[piano di montaggio della **PULSANTIERA**]



Elenco Componenti:

S1: microswitch
S2: microswitch
LD85: LED 5 mm verde
Varie:
- Strip femmina 6 vie

[piano di montaggio del **CARICABATTERIA**]



Elenco Componenti:

R18: 470 ohm
R19: 2,2 kohm
C15: 4,7 μ F 100 VL elettrolitico
C16: 100 nF multistrato
C17: 4,7 μ F 100 VL elettrolitico
LD86: LED 5 mm rosso
D6, D7: 1N5819
U11: MCP73831T-2 smd SOT23-5L
Varie:
- Plug alimentazione
- Connettore JST 2 poli
- Circuito stampato
- Batteria LiPo 3,7V - 2Ah (o superiore)

l'alimentatore).

Volendo aumentare l'autonomia della batteria potete ricorrere ad una LiPo con maggiore capacità (ne esistono da 4.000 mAh e perfino da 6.000 mAh) oppure potete anche agire aumentando il valore delle resistenze R3, R4, R16, attualmente da 180 ohm. È una cosa che potete fare tranquillamente, a patto che usiate i LED da noi indicati o comunque dei LED ad alta luminosità e basso consumo; il valore lo troverete sperimentalmente, facendo prove nel luogo in cui installerete l'OROLed, considerando la luminosità ambientale. Infatti, aumentando il valore di tali resistenze avrete un risparmio energetico ma anche una diminuzione della luminosità dei LED, quindi occorre trovare la giusta combinazione.

Il collegamento della schedina al PCB della logica avverrà mediante un cavetto bicolore saldato direttamente ai capi "OUT + e -" della schedina stessa (attenzione alla polarità), mentre sul lato del PCB della logica bisognerà prevedere uno spinotto femmina tipo Molex, da collegare al maschio BAT1 previsto sul PCB; in tal modo non ci sarà poi più possibilità di errore. (Fig. 3).

Sconsigliamo di usare costantemente l'alimentatore col circuito LiPo e soprattutto con la batteria collegata; se optate per un'alimentazione costante da rete, vi conviene collegare l'alimentatore all'altra presa jack (PWR, quella prevista sul PCB della logica), quindi adottare la configurazione descritta precedentemente. Inoltre è assolutamente da evitare che la batteria LiPo si scarichi completamente, perché se dovesse scendere sotto il livello minimo non si potrebbe più recuperare.

Nei periodi in cui avete necessità di lasciare l'OROLed spento sinceratevi che la batteria sia carica, ma non al massimo: una condizione tra il 70% ed il 90% è ideale per la conservazione di queste batterie.

LA COSTRUZIONE

Iniziamo dal PCB della logica, che è la scheda più complessa, per numero di componenti. Come sempre, converrà montare prima i componenti a più basso profilo: resistenze e diodi, quindi gli zoccoli degli integrati ed i condensatori ceramici o in poliestere; a seguire i transistor e il quarzo, il condensatore elettrolitico. Se opterete per l'alimentazione tramite batteria LiPo, potete evitare di montare i componenti: jack PWR, C13, D5, anche se noi, visto il bassissimo costo di tali componenti, vi consigliamo di prevederli comunque; se un domani cambierete idea, non dovrete fare altro che spostare l'innesto dell'alimentatore esterno. Il quarzo andrà montato appoggiato sul PCB, come da serigrafia e, possibilmente,

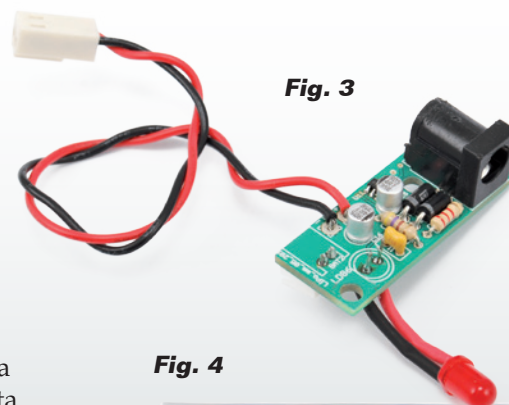
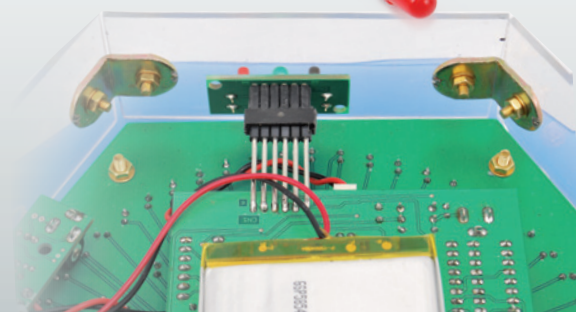


Fig. 3

Fig. 4



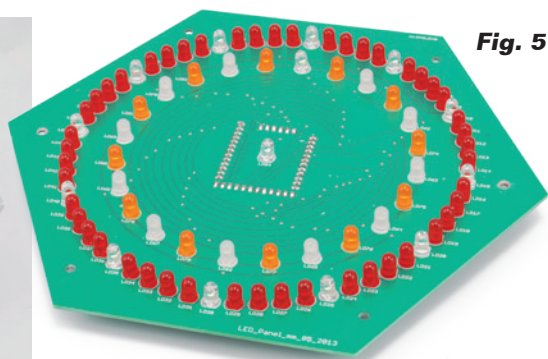
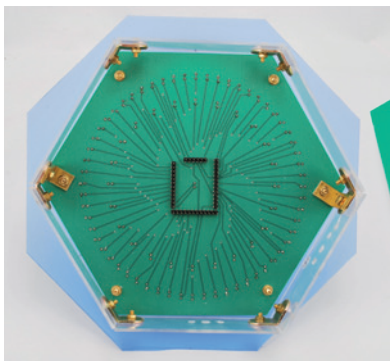


Fig. 5

saldato anche dal lato della testa (è prevista un'apposita piazzola). I connettori header (femmina) CN2, CN3, CN4 e CN5 dovranno essere montati facendo molta attenzione a che siano perfettamente perpendicolari al PCB; in caso contrario sarà estremamente problematico innestarli con i relativi pin del PCB LED Panel. Tali header dovranno avere un'altezza tale da superare tutti gli altri componenti presenti sul PCB; in caso contrario si potrebbero creare dei cortocircuiti tra questi componenti e le saldature del LED Panel. In alternativa si potranno usare sul LED Panel dei pin piuttosto lunghi ed il risultato sarà comunque raggiunto. Il connettore Molex BAT1 e l'altro connettore header (femmina a 90°) CN1, potete montarli indifferently sul lato componenti o sul lato saldature; la scelta dipenderà molto dal tipo di contenitore che userete o che costruirete: l'importante è che non dimentichiate mai di verificare le polarità delle connessioni, non ci sono particolari pericoli ma certamente il circuito non funzionerà come si deve (Fig. 4).

Il PCB dei pulsanti va montato obbligatoriamente, in quanto essi servono per la regolazione dell'orario, ma il LED 85 funzionerà solo con l'alimentazione via PWR, quindi se non montate tale presa ed i successivi componenti, inutile montare questo LED. Il PCB LiPo invece va montato solo se optate per l'alimentazione

a batteria senza alimentatore da rete; fate molta attenzione all'integrato, che è un SMD di piccolissime dimensioni. Anche in questo caso il connettore BAT2 (tipo jst 2 poli) può essere montato sul lato componenti o saldature, ma sempre senza perdere di vista la polarità. All'altro estremo è previsto il montaggio del jack PWR1 (posizione obbligata) oppure del connettore jst CELL1. Questo connettore lo dovete montare qualora decidiate di usare come fonte di energia alternativa quella proveniente da una o più celle solari: in questo caso dovete fare attenzione alla tensione massima, che non dovrà superare i 6V, ma anche

a quella minima, che non dovrà scendere sotto i 4V, mentre la corrente minima garantita non dovrà essere inferiore a 80-100 mA; con questi valori l'OROLed lavorerà regolarmente e durante le ore di luce si caricherà anche la batteria, quanto basta per coprire abbondantemente il periodo notturno. In tal modo potreste teoricamente avere un'autonomia illimitata. Volendo è possibile montare sia il connettore CELL1 che la presa PWR1, iniziando dal primo, che andrà montato dal lato saldature e poi spuntando il più possibile i pin, dopo averli saldati; il PWR1 si potrà montare lato componenti tenendolo leggermente sollevato rispetto al PCB. Il LED Panel tutto sommato non è problematico da montare: basta fare attenzione alla polarità ed al colore dei LED che, non essendo tutti uguali, prevedono una ben precisa collocazione, altrimenti poi aiuteranno poco nella leggibilità dell'orario. Prima di iniziare a montare e saldare i



Fig. 6

Fig. 7



LED è indispensabile aver deciso in quale contenitore montare l'OROLed; nel nostro caso entrambe le possibilità ci hanno permesso di infilare completamente i LED nel pannello, tale modalità è la più semplice e comoda ma ha un aspetto negativo, l'assoluta immobilità dei LED, nessuna possibilità di poterne muovere qualcuno se si facesse un pur minimo errore di foratura della mascherina. Potete invece decidere di lasciar fuoriuscire dal pannello una parte dei reofori, ma in questo caso dovete assolutamente trovare un modo per posizionarli tutti alla stessa altezza, in caso contrario l'effetto finale sarà pessimo. Dopo aver saldato tutti i LED (controllate bene ogni reoforo, facilmente una saldatura può sfuggire tra 170 fori così vicini l'un l'altro!) potrete saldare le file di pin maschi, che abbiamo denominato CN1, CN2, CN3 e CN4, che andranno inserite sul lato saldature del LED Panel; anche in questo caso, prestate la massima attenzione alla perpendicolarità dei pin rispetto al PCB, per garantirvi un innesto preciso del LED Panel nell'ORO logic (Fig. 5).

Il contenitore, una volta tanto, rappresenta un elemento fondamentale di questo progetto; un orologio da parete non è solo un utile strumento temporale, ma anche un complemento d'arredo, non a caso ne esistono infinite varietà. Noi abbiamo fatto due

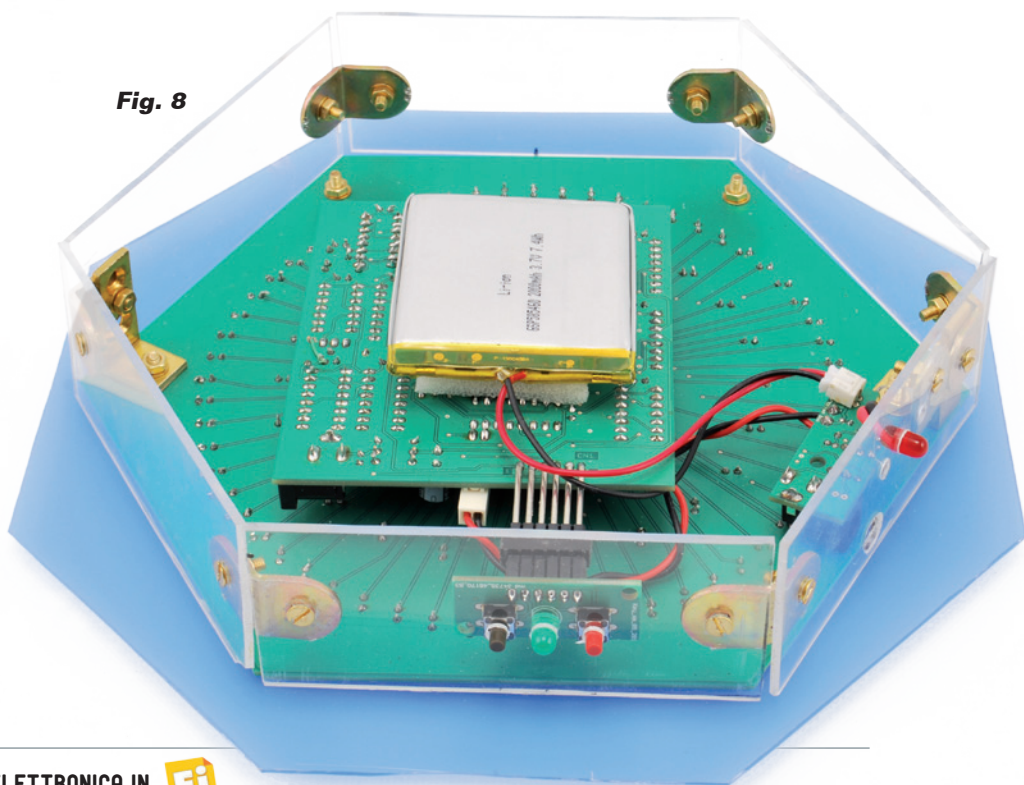
diversi esperimenti: il primo è consistito nel recuperare la carcassa di un orologio da parete low-cost, eliminando l'elettronica e forando il pannello (in genere è alluminio molto sottile); se si riuscisse a reperire un modello in cui le cifre delle ore e le tacche dei minuti combaciassero con la posizione dei LED, il risultato sarebbe eccellente. In Fig. 6 potete vedere una realizzazione del genere. L'autocostruzione di un contenitore dal design decisamente più accattivante non è proprio semplicissima, tuttavia, se si dispone del materiale e della necessaria attrezzatura, è fattibile. Noi abbiamo creato un contenitore esagonale a misura del LED Panel, in plexiglass trasparente, lavorando con Dremel, seghetto e lime, poi abbiamo ritagliato un ottagono in materiale più morbido, per ottenere la mascherina

che successivamente abbiamo forato.

Per effettuare la foratura della mascherina potete usare, come dima, la serigrafia (che scaricherete dalla sezione download del nostro sito) del LED Panel su un comune foglio di carta, che ritaglierete ed andrete poi a fissare con del nastro da carrozziere sulla mascherina, facendo attenzione al verso. A quel punto l'ideale sarebbe disporre di un trapano a colonna, iniziando a forare con la punta da ferro da 1 mm: dovrete praticare un foro esattamente al centro dei due cerchietti della serigrafia che corrispondono ai reofori dei LED, per tutti gli 85 LED. Una punta così sottile vi permette di forare con la massima precisione creando quindi una "guida" nella quale ripasserete prima con una punta da 3 mm ed infine con quella da 5 mm (sempre da ferro).

A questo punto non vi resta che dare una bella limata ai contorni dei fori, in modo da eliminare le sbavature. Questa tecnica è lunga

Fig. 8



e richiede molta pazienza, ma vi garantisce i risultati che vedete nelle foto (Fig. 7) del prototipo. In esse potete vedere come abbiamo collegato e fissato le due schede esterne per i pulsanti e la carica della batteria.

La batteria LiPo andrà fissata al PCB della logica, ricordando che tra i due componenti dovrete interporre un ritaglio di cartoncino o di spugna, per evitare il pericolo di forare accidentalmente la batteria a causa della pressione esercitata dalla sporgenza di qualche reoforo dal lato saldature; la cosa sarebbe estremamente pericolosa, perché la batteria al litio, se perforata, potrebbe anche esplodere.

Alle due facce della spugna o del cartoncino dovrete applicare un biadesivo del tipo gommato ed otterrete così un buon risultato (Fig. 8); naturalmente potrete decidere di fissare la batteria altrove, perché in genere c'è molto spazio, viste le ridotte dimensioni del PCB della logica. Però non fissate la batteria prima di aver completato le operazioni di collaudo e taratura, in quanto potreste aver necessità di smontare tutto più volte.

COLLAUDO, TARATURA E REGOLAZIONE

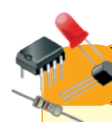
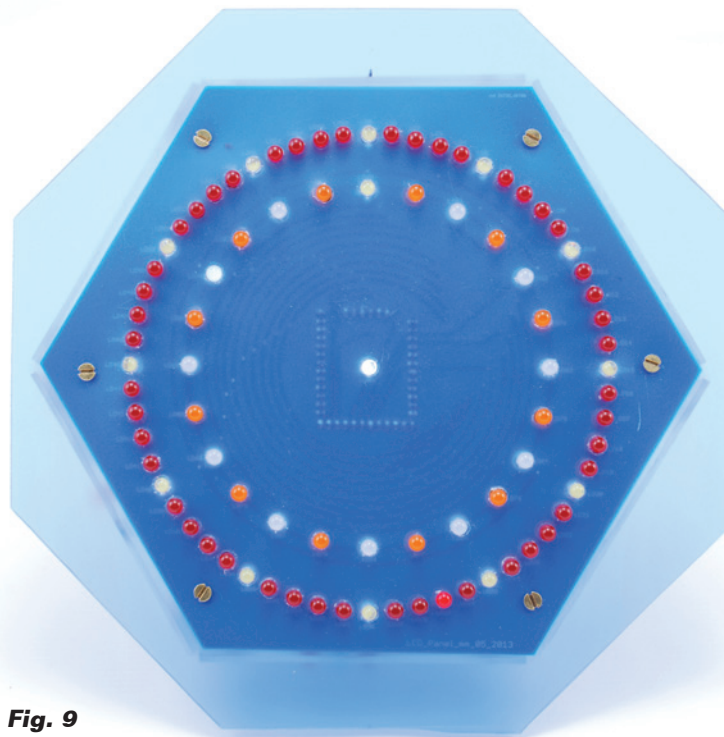
Prima di assemblare i due PCB principali verificate che la vite del compensatore C2 sia all'incirca al centro della sua corsa e settate il jumper JP1 tra il 2 ed il 3; questo ponte porterà all'ingresso del primo contatore U3 non più la frequenza base di 1 Hz, ma quella ben più veloce di 2.048 Hz (2 kHz). A questo punto potete collegare l'alimentazione (alimentatore o batteria). Lo scopo di questo settaggio è semplice, questa frequenza farà accendere il LED dei secondi in modo fisso e quelli dei minuti e delle ore/mezze ore a velocità elevatissima, questo vi

permetterà di verificare che tutti i LED si accendano correttamente in sequenza, in caso contrario vedrete immediatamente che problemi ci sono e andrete a verificare a colpo sicuro.

Risolti gli eventuali problemi di saldature o cortocircuiti, il jumper JP1 andrà posizionato tra 1 e 2, in modo da riportare il clock alla normalità, quindi bisognerà procedere con la taratura dell'oscillatore. Infatti C2 serve a regolare la frequenza di oscillazione esattamente a 32,768 kHz (32.768 Hz), perché in caso contrario la frequenza di clock dell'orologio non sarà mai di 1 Hz, ma qualcosa in meno o in più, e tale differenza influirà ovviamente sulla precisione del nostro OROLED. La taratura va fatta senza il LED Panel, anche perché non serve; è necessario disporre di un frequenzimetro o di un oscilloscopio digitale da collegare tra il test point TP1 e GND. Letta la frequenza, se diversa da 32.768 Hz dovrete ritoccare leggermente il compensatore C2 con un piccolo cacciaviti, fino a leggere il valore esatto. Consigliamo di effettuare

la taratura quando il circuito è alimentato da almeno 15 minuti, in modo che tutti i componenti si siano stabilizzati a quella che sarà la normale temperatura di lavoro. A questo punto non vi resta che montare tutto, fissare e collegare la batteria, regolare l'ora esatta agendo sui due pulsanti S1 e S2 ed appendere l'OROLEd alla parete (Fig. 9).

Fig. 9



per il MATERIALE

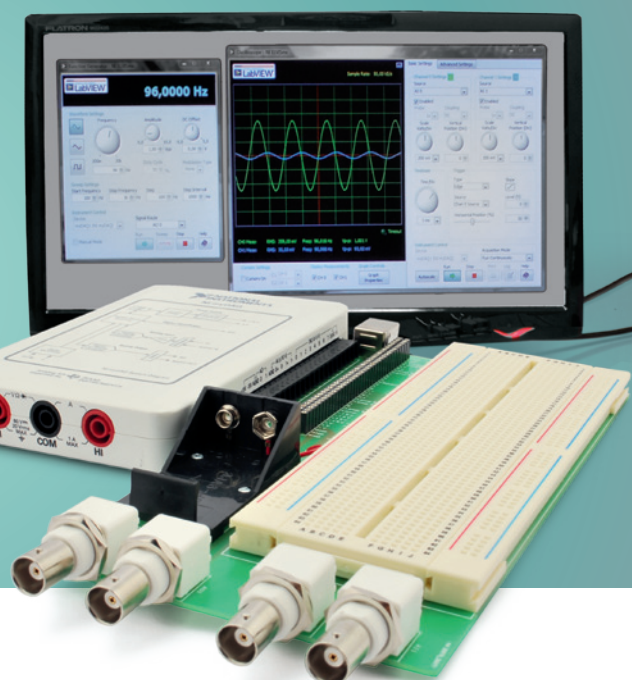
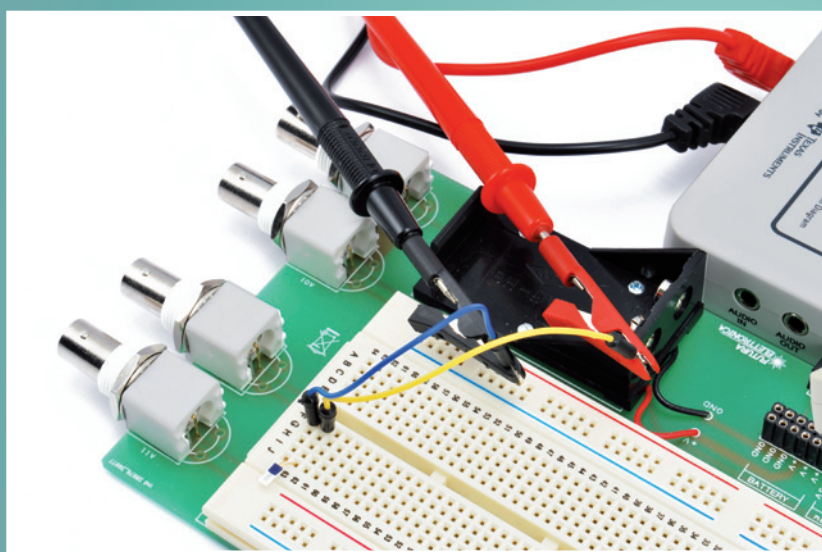
Tutti i componenti utilizzati in questo progetto sono di facile reperibilità; i master dei PCB possono essere scaricati dal sito della rivista. La batteria ai polimeri di Litio utilizzata nel progetto (cod. PLIB2A) costa 19,00 Euro IVA compresa ed è disponibile presso Futura Elettronica, come quasi tutti gli altri componenti.

Il materiale va richiesto a:
Futura Elettronica, Via Adige 11,
21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
<http://www.futurashop.it>

myKIT

MISURA DI TEMPERATURA CON SONDA RTD

di DAVIDE SCULLINO



National Instruments ha da tempo messo a disposizione del mondo accademico myDAQ, un dispositivo di acquisizione dati portatile e a basso costo che utilizza strumenti basati sul software NI LabVIEW, consentendo di misurare e analizzare segnali in tempo reale. NI myDAQ è l'ideale per effettuare misure con sensori analogico e, gestito da LabVIEW, permette di analizzare ed elaborare i segnali

acquisiti e controllare processi. In collaborazione con IRS, abbiamo sviluppato un kit didattico che prevede la realizzazione 22 esperimenti proprio usando NI myDAQ, comprendente tutti i componenti e una scheda (chiamata Protoboard) con montata una breadboard e le connessioni di alimentazione e di segnale, oltre ad un connettore a 20 contatti per il collegamento ad NI myDAQ. Abbiamo iniziato la

Collegiamo a myDAQ un sensore RTD al platino e leggiamo i valori istantanei e storici su un Pannello Frontale costruito integrando, grazie a LabVIEW l'equazione di Callendar-Van Dusen.

descrizione delle esercitazioni nel n° 171; ora proseguiamo con un nuovo esperimento, che consiste nell'effettuare misure di temperatura utilizzando un sensore Platinum RTD collegato a myDAQ di National Instruments. I dati corrispondenti, forniti da myDAQ, saranno acquisiti tramite DAQ Assistant, installato in LabVIEW mediante il driver NI DAQmx e poi verranno convertiti in valori di temperatura utilizzando la programmazione in LabVIEW. L'esercitazione avrà luogo collegando la sonda di temperatura ai terminali DMM (Digital MultiMeter) della scheda myDAQ e il software LabVIEW di National Instruments.

Una sonda RTD (Resistance Temperature Detector) è un sensore di temperatura costituito da un metallo, in questo caso platino, la cui resistenza varia linearmente con la temperatura in un certo intervallo. Esiste una gran varietà di sonde RTD per misurare un vasto range di temperature, in applicazioni e ambienti sensibili alla temperatura, ma a causa del costo elevato del platino questo genere di sensore è meno usato dei termistori. In questa esercitazione vi servirete di una sonda RTD Pt-100; il valore 100 sta ad indicare la resistenza nominale del componente a una temperatura di 0° C. Per compiere le misure si fa attraversare il sensore da una corrente costante e si legge la caduta di tensione che ne deriva ai suoi capi. Utilizzando la Legge di Ohm, la tensione e la corrente permettono di determinare la resistenza della sonda, la quale varia con la temperatura. Il valore di resistenza viene quindi utilizzato per calcolare la temperatura utilizzando l'equazione Callendar-Van Dusen.

Fig. 1
Sonda di temperatura RTD.



$$T_R = \frac{-R_0 A + \sqrt{R_0^2 A^2 - 4R_0 B(R_0 - R_T)}}{2R_0 B}$$

Equazione di Callendar-Van Dusen

TR = temperatura rilevata R0 = resistenza nominale
RT = resistenza sonda

ISTRUZIONI PER IL CABLAGGIO

La sonda Pt-RTD viene collegata nel circuito come una resistenza, quindi un capo si connette al polo positivo di MyDAQ e l'altro al negativo; l'orientamento non ha importanza. Secondo il tipo di sonda (2-wire, 3-wire, o 4-wire) può essere necessario unire tra loro i capi dei due lati, in modo da ridurre i collegamenti a due. In pratica, usando una sonda RTD 3-wire o 4-wire dovete unire i capi di un lato e quelli dell'altro, perché i terminali DMM sul myDAQ sono configurati per una misurazione della resistenza a due fili, dove la corrente di eccitazione proviene dai terminali di tensione d'ingresso. Nella Fig. 2 si vede la connessione della sonda RTD utilizzata in questa esercitazione, che è di tipo 2-wire.

PANNELLO FRONTALE LabVIEW

L'interfaccia utente che creerete mostra i valori della temperatura rilevata riportati su un indicatore numerico e su un grafico: quest'ultimo memorizza i valori storici in modo che si possano vedere i cambiamenti della temperatura nel tempo; inoltre, su un apposito indicatore viene riportato il valore di resistenza misurato della sonda e un controllo numerico permette di impostare il valore di resistenza nominale R0 che caratterizza il tipo di sonda RTD utilizzata. Sono presenti anche altri due controlli numerici che permettono di inserire le costanti A e B che vengono utilizzate nell'equazione di Callendar-Van Dusen.

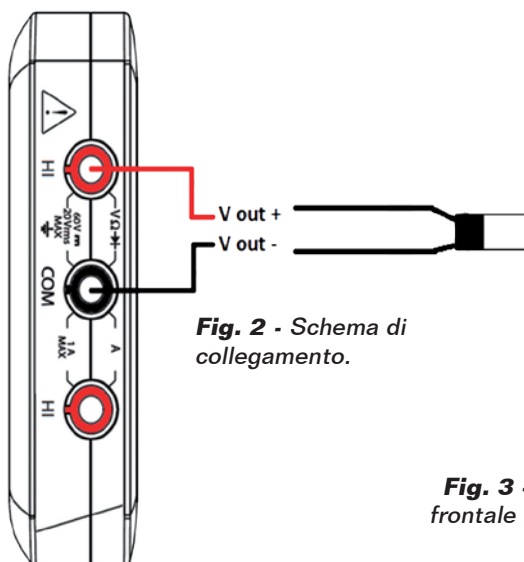


Fig. 2 - Schema di collegamento.

Fig. 3 - Pannello frontale LabVIEW.

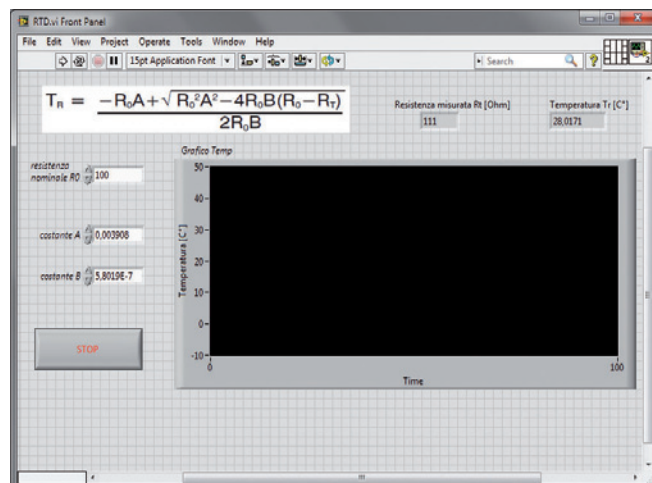




Fig. 4 - Schema a blocchi di codifica.

CODIFICA DELLA STRATEGIA

In LabVIEW avete bisogno di misurare il segnale proveniente dalla resistenza Pt-RTD impostando il valore della resistenza nominale R_0 che la caratterizza. Tuttavia per avere una migliore risoluzione di misura dovete limitarvi ad acquisire la resistenza da 0 a 200 Ω . Secondo le specifiche del dispositivo myDAQ, questa scelta vi garantisce una risoluzione di 0,1 Ω . Il valore di resistenza acquisito viene poi convertito in una temperatura tramite l'equazione polinomiale di Callendar-Van Dusen, che richiede la conoscenza di alcune delle costanti che la compongono. Infine, si fornirà in uscita il risultato a un indicatore numerico e a un grafico della temperatura sul Pannello Frontale. Il diagramma a blocchi di LabVIEW è molto simile allo schema a blocchi di codifica.

COME FUNZIONA

All'interno del ciclo *While*, sulla sinistra è situato DAQ Assistant, configurato per acquisire un singolo valore dai terminali DMM del myDAQ ogni volta che viene eseguito. Utilizzando la sonda RTD, una volta che il valore di resistenza è stato acquisito viene passato a un indicatore numerico e all'interno del *Formula Node*. Tale valore viene poi convertito in temperatura in gradi Celsius usando l'equazione polinomiale di Callendar-Van Dusen e le costanti fornite dal costruttore del sensore utilizzato, inserite tramite controlli numerici. Infine, la temperatura è visualizzata sull'indicatore Temperatura Tr [C°] e sul grafico del Pannello Frontale. Tutto il codice presente nel ciclo *While* continua a funzionare fino a

che non viene premuto il pulsante Stop sul Pannello Frontale. Note che le misure effettuate tramite gli ingressi DMM sono temporizzate dal software, quindi la velocità di aggiornamento è influenzata dal carico sul computer e dall'attività sull'USB.

COSTRUZIONE DEL PANNELLO FRONTALE

Aprirete un nuovo file LabVIEW e andate sul Pannello Frontale; qui cliccate con il pulsante destro del mouse e comparirà la palette *Controls* dove sono elencati tutti i tipi di controlli e di indicatori, ma anche decorazioni, tabelle, ecc. Se cliccate sulla puntina in alto a sinistra della palette potete bloccarla sullo schermo; ciò vi risparmierà di dover fare clic con il pulsante destro del mouse sul Pannello Frontale ogni volta che dovete farla riaprire. Nella palette *Controls* selezionate *Modern* (la prima voce), poi *Numeric* e inserite tre controlli numerici (*Numeric Control*) e due indicatori numerici (*Numeric Indicator*), uno per volta, sul Pannello Frontale. Ora inserite anche un pulsante Stop, che trovate sempre sulla palette *Controls* ma sotto la voce *Modern»Boolean*, ed un *Waveform Chart*, situato alla voce *Modern»Graph*. Disponeteli con lo strumento *Positioning* e rinominateli con lo strumento *Labeling*, come in Fig. 6. Rinominate anche l'etichetta dell'asse y del grafico in "Temperatura [C°]" utilizzando lo strumento *Labeling* e deselezionate la voce *Auto-scale y* dal menu a tendina che comparirà facendo clic col tasto destro del mouse sull'etichetta dell'asse y. Ora, con lo strumento *Labeling* inserite, in corrispondenza dei valori estremi dell'asse y, i valori -10, +50, perché difficilmente supererete queste soglie di temperatura (se dovete effettuare una misura di temperatura non compresa in questo range inserite soglie adeguate). La formula di Callendar-Van Dusen che vedete in alto a destra nel Pannello Frontale

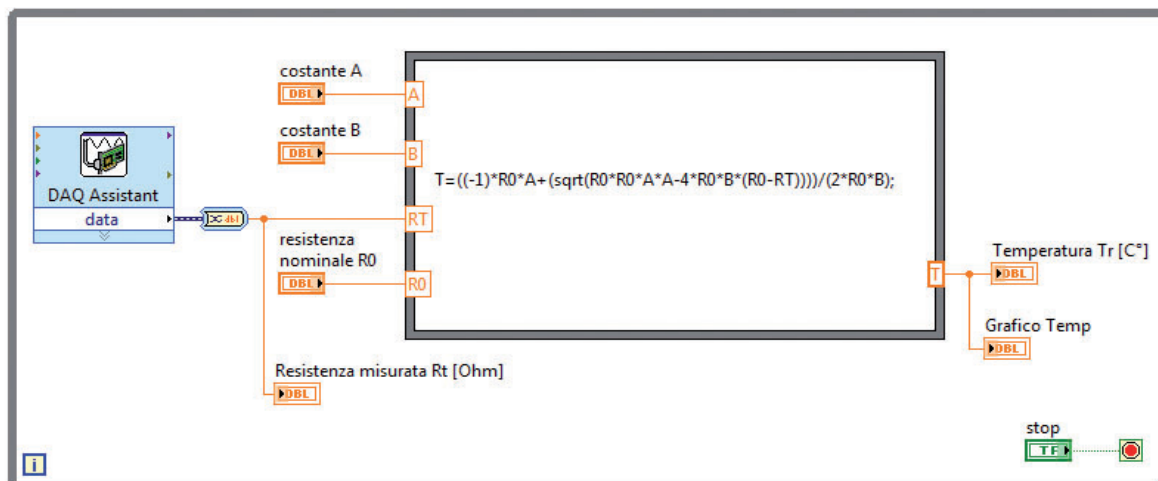


Fig. 5
Diagramma
a blocchi
LabVIEW.

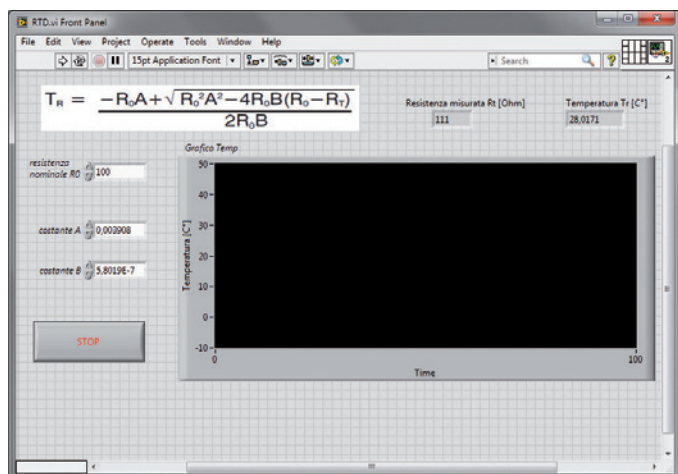


Fig. 6 - Pannello frontale LabVIEW completato.

è una semplice immagine incollata in quel punto. Ora inserite i seguenti valori sui controlli numerici A e B e sul controllo numerico *resistenza nominale R0*: R0=100, A=0,003908, B=-5,8019E-7. Questi parametri vanno impostati come controlli e non come costanti, perché può accadere che sonde RTD di costruttori differenti forniscano dati diversi da quelli attesi e quindi ci sia la necessità di variarli dal Pannello Frontale. Comunque, andate su *edit* nella barra menu in alto e selezionate *Make Current Values Default*; ciò vi risparmierà di dover inserire nuovamente questi valori ogni qualvolta aprirete e utilizzerete questo VI (ovviamente se utilizzerete sonde diverse con parametri A, B e R0 diversi inserite i valori corretti e ripetete tale operazione).

COSTRUZIONE DEL DIAGRAMMA A BLOCCHI

Ora vi occuperete della realizzazione del diagramma a blocchi. Se non lo visualizzate, potete farlo selezionando *Window»Show Block Diagram* o digitando <Ctrl+E>. Gli oggetti del Pannello Frontale appaiono come terminali sul diagramma a blocchi. I terminali sono porte d'ingresso e di uscita che scambiano informazioni tra il Pannello Frontale e il diagramma a blocchi. Qui dovreste già visualizzare i controlli numerici e il pulsante Stop con le stesse etichette che avevate messo sul Pannello Frontale. Inoltre dovreste visualizzare i terminali, ad esempio quello del pulsante Stop, con la seguente icona: . È però preferibile visualizzarlo così:  (ciò permette di ridurre lo spazio che occupa sul diagramma a blocchi); per farlo, impartite il comando *Tools»Options..* e sulla sinistra della finestra che si aprirà selezionate *Block Diagram*, quindi alla voce *General* disabilitate *Place front panel terminals as icons*. Ora, prima di iniziare ad implementare il diagramma a blocchi, osservate come i rettangolini che rappresentano i controlli possono essere impostati

sia come indicatori (cioè ricevono dati) sia come controlli (cioè producono dati). Nel diagramma a blocchi, il controllo si distingue dall'indicatore per due particolari: il primo è una freccia sul terminale che indica la direzione del flusso dei dati (i controlli hanno frecce che mostrano il dato partire dal terminale, mentre negli indicatori la freccia che mostra il dato entra nel terminale); il secondo è il bordo intorno al terminale (i controlli hanno un bordo più spesso di quello degli indicatori). Per passare da indicatore a controllo dovete cliccare col tasto destro del mouse sopra ogni rettangolino che rappresenta l'oggetto e impartire il comando *Change to Control*; questa operazione può essere effettuata anche dal Pannello Frontale. Notate che il pulsante Stop è già impostato come controllo. Prestate attenzione ai collegamenti che effettuerete sul diagramma a blocchi: essi permettono di trasferire dati tra oggetti del diagramma a blocchi; ogni collegamento parte da una sorgente di dati e la collega ad uno o più VI e funzioni che leggono i dati stessi. I collegamenti hanno differenti colori, stili e spessori, in funzione del tipo di dati. Un collegamento spezzato  appare come una linea nera tratteggiata con una X rossa in mezzo. I collegamenti possono rompersi per svariati motivi: ad esempio se si tenta di collegare due oggetti con tipi di dati tra loro incompatibili. La **Tabella 1** illustra i tipi di collegamento più comuni.

I collegamenti possono essere realizzati solamente tra ingressi e uscite che veicolino dati compatibili tra loro. Nel diagramma a blocchi, facendo clic con il pulsante destro del mouse, compare la palette *Functions*, che contiene i VI, le funzioni e le costanti da usare per creare il diagramma a blocchi. Se volete bloccare la palette *Functions* sullo schermo, fate clic sulla puntina che si trova in alto a sinistra. Ora potete iniziare la costruzione del diagramma a blocchi; vedrete che i terminali degli indicatori e dei controlli inseriti nel pannello frontale sono già presenti all'interno del *While Loop*. In questo VI il blocco DAQ Assistant è configurato per l'acquisizione di valori di resistenza dagli ingressi DMM del dispositivo myDAQ. Le seguenti istruzioni descrivono passo-passo la configurazione

Tabella 1

Tipo di collegamento	Scalare	Array 1D	Array 2D	Colore
Numerico	 	 	 	Arancione (virgola mobile), Blu (intero)
Booleano				Verde
Stringa				Rosa

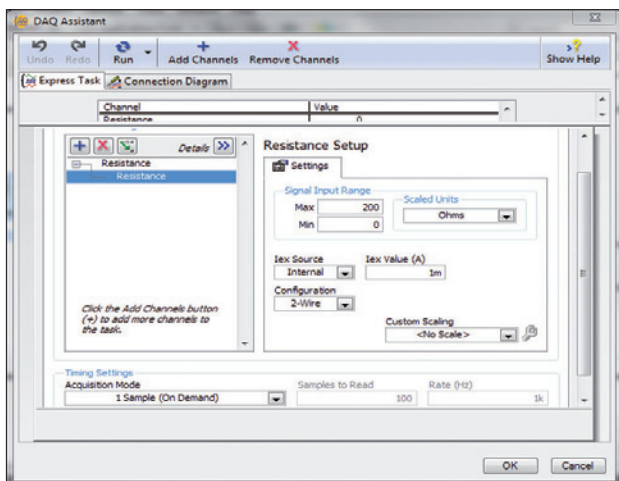


Fig. 7 - Configurazione DAQ Assistant.

del DAQ Assistant:

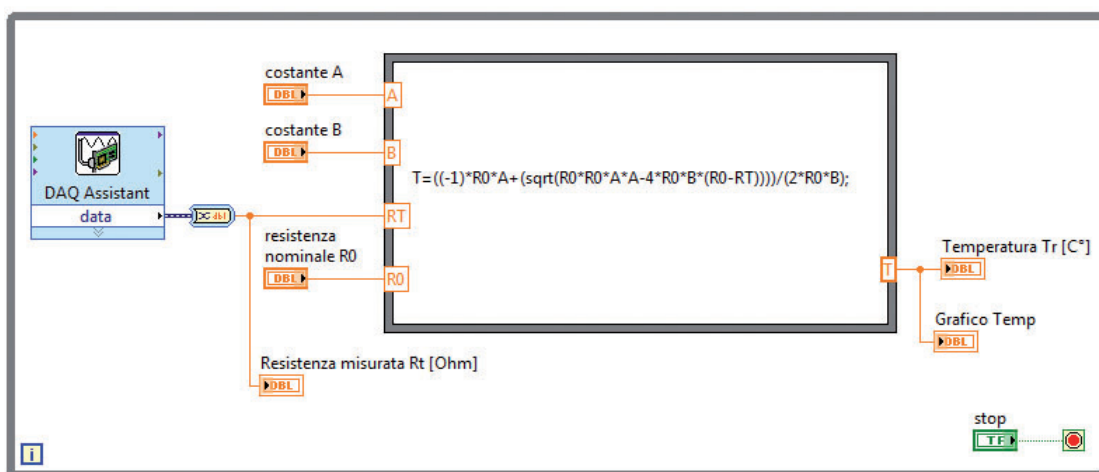
1. Assicurarsi che myDAQ sia collegato al PC;
2. Premere Ctrl-Spazio per aprire la finestra Quick Drop (al primo utilizzo, questa operazione richiederà circa un minuto);
3. Cercare la voce DAQ Assistant e fare doppio clic su di essa quando appare nella lista;
4. Trascinarla sul diagramma a blocchi;
5. Quando appare un nuovo riquadro per le attività di configurazione rapida, selezionare:
 - a. Acquire Signals;
 - b. Analog Input;
 - c. Resistance;
 - d. Dev 1 (NI myDAQ). Nota: Se si dispone di altri hardware NI installati, myDAQ non sarà Dev1.
 - e. dmm;
 - f. Finish;
6. Inserire in Resistance Setup il corretto Signal Input Range;
 - a. Max: 200;

- b. Min: 0;
7. Configurazione dell'eccitazione:
 - a. Internal;
 - b. 1mA;
8. Cambiare il Timing Settings:
 - a. 1 Sample (On Demand);
9. Fare clic su OK.

Una volta terminata la configurazione, andate sulla palette *Functions* e selezionate *Programming»Structures»Formula Node*, quindi posizionare il *Formula Node* nel diagramma a blocchi e, utilizzando lo strumento *Positioning*, ingranditelo un po'. Cliccate con il pulsante destro del mouse in alto sul bordo sinistro del *Formula Node* e selezionate *Add Input*: comparirà un riquadrino sul bordo del *Formula Node*; con lo strumento *Labeling* assegnategli il nome A.

Ora, con lo strumento *Wiring* collegate il controllo numerico costante A all'ingresso del *Formula Node* appena creato. Ripetete questa operazione altre tre volte in modo da collegare al *Formula Node* i restanti ingressi B, Rt e R0, facendo attenzione che per collegare l'uscita data del DAQ Assistant all'ingresso Rt bisogna inserire il blocco *Convert from Dynamic Data* situato nella palette *Functions* sotto la voce *Express»Signal Manipulation* (Fig. 8). Questo permetterà all'espressione che inserirete nel *Formula Node* di avere un riferimento a ogni ingresso che utilizzerà. Appena inserirete il blocco *Convert from Dynamic Data* si aprirà una finestra per la configurazione; dalla voce *Conversion* selezionate *Single Scalar* e fate clic su OK. Ora andate sul bordo destro del *Formula Node*, cliccate col tasto destro del mouse, selezionate *Add Output* e nominate questa uscita T. Con lo strumento *Wiring* collegate questa uscita all'indica-

Fig. 8
Diagramma
a blocchi
LabVIEW.



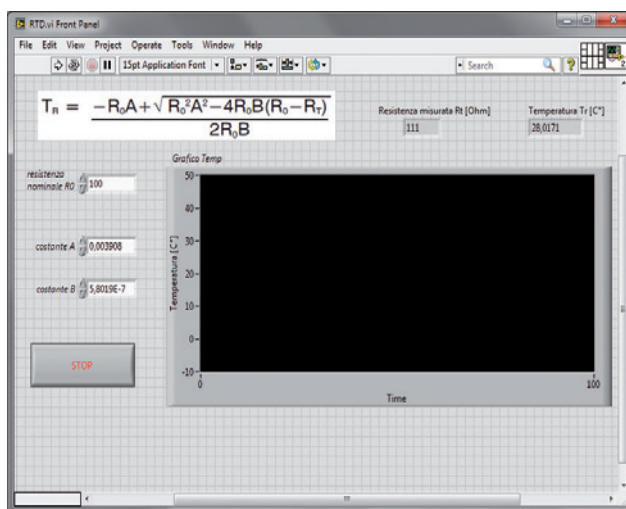


Fig. 9 - Il VI in esecuzione.

tore numerico *Temperatura Tr [C°]* e al *Grafico Temp.* Per inserire l'espressione di Callendar-Van Dusen nel formula node, utilizzate lo strumento *Labeling* e scrivete quanto segue:

$$T = ((-1) * R_0 * A + (\sqrt{R_0 * R_0 * A * A - 4 * R_0 * B * (R_0 - R_t)})) / (2 * R_0 * B);$$

Se volete visualizzare l'espressione nel centro del *Formula Node*, premete più volte il tasto Invio per andare a capo di una nuova riga. Come potete osservare, nell'espressione sono presenti diversi termini che avete già inserito come ingressi sul bordo sinistro del *Formula Node*.

Inoltre il risultato, che è la temperatura rilevata, viene passato come valore in uscita all'output *T* presente sul bordo destro del *Formula Node*. Per avere anche un'indicazione sulla misura della resistenza della sonda, collegate l'indicatore *Resistenza misurata Rt* all'uscita *Data* di DAQ Assistant dopo che è stata convertita dal blocco *Convert from Dynamic Data* dal tipo di dato "dinamico" al valore di tipo "double" (Fig. 8).

Per finire, inglobate tutto il codice grafico fatto finora in un *While Loop* che troverete sulla palette *Functions* alla voce *Programming»Structures* e collegate con lo strumento *Wiring* il terminale del pulsante Stop al *Conditional Terminal* nell'angolo in basso a destra del ciclo *While*.

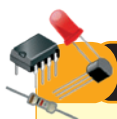
I commenti del caso, potete inserirli a vostro piacimento utilizzando lo strumento *Labeling*; i commenti non interferiscono con il codice grafico, ma, semmai, ne aumentano la comprensione. Se preferite cambiare le impostazioni di scrittura (font, dimensione, colore, ecc.), sia nel Pannello Frontale che nel diagramma a blocchi, nella barra del menu in alto cliccate su *Text Settings* (è il riquadro che riporta scritto la dimensione e il font che state utiliz-

zando, ossia questo: 15pt Application Font).

Quando avrete finito di costruire il vostro VI andate su *File»Save as* e scegliete un nome e una destinazione per salvare il vostro VI. Ora cliccate sul pulsante *Run* nella barra degli strumenti per mettere il VI in esecuzione (Fig. 9).

Mentre il VI è in esecuzione, l'icona del pulsante *Run* si trasforma nella seguente: . Una volta terminata l'esecuzione, il pulsante *Run* torna al suo stato originale e gli indicatori del Pannello Frontale mantengono i dati visualizzati. Mentre il VI è in esecuzione, appare il pulsante *Abort Execution* (); facendovi clic potete fermare il VI immediatamente. Il pulsante *Abort Execution* va usato soltanto se non riuscite ad arrestare il VI in altro modo.

Notate che, di norma, se un VI non va in esecuzione vuol dire che in esso vi sono degli errori che non lo rendono eseguibile; potete rendervi conto di ciò perché quando il VI che state creando o modificando contiene errori, il pulsante *Run* appare spezzato (). In questo caso, anche dopo aver finito i collegamenti nel diagramma a blocchi, il VI non è eseguibile. Generalmente l'errore può consistere in un ingresso richiesto non collegato o in un collegamento spezzato. Per risolvere il problema, fate clic sul pulsante *Run* spezzato in modo da accedere alla finestra di dialogo *Error List*, nella quale viene elencato e descritto ogni errore riscontrato; qui, facendo doppio clic su un errore andate direttamente alla posizione in cui esso si trova nel diagramma a blocchi e rimediate all'errore.



per il MATERIALE

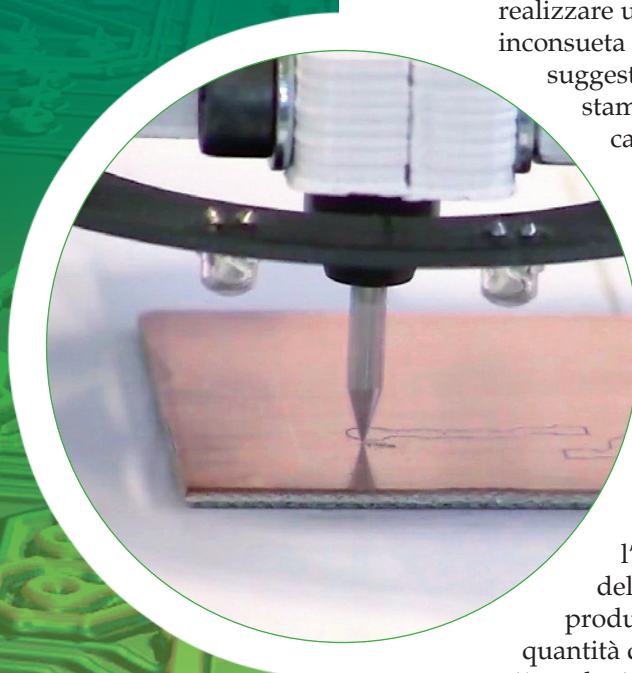
Questo progetto fa parte del set di esperimenti didattici realizzato da IRS-Elettronica per la board NI myDAQ con il supporto del software LabVIEW. Il set (cod. SETmyKIT, Euro 232,00) comprende tutti i componenti per realizzare i 22 esperimenti, i VI relativi ed una protoboard. È possibile richiedere protoboard aggiuntive (cod. PROTOBOARDmyDAQ) al prezzo di 52,00 Euro. Tutti i prezzi sono comprensivi di IVA e non comprendono la board NI myDAQ né il software LabVIEW. Futura Elettronica distribuisce questi prodotti presso gli utenti privati mentre la distribuzione presso gli Istituti Scolastici è a cura di IRS srl (Via Vigonovese 81 - 35127 PADOVA - tel: +39 049 8705156 - info@irsweb.it).

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
www.futurashop.it

LA 3DRAG DIVENTA CNC E INCIDE I PCB

a cura della REDAZIONE



Come trasformare la nostra stampante 3D in una fresa a controllo numerico utilizzabile per creare circuiti stampati senza photoresist, sviluppo e acido, semplicemente inviando al controller il file G-Code ottenuto partendo dai Gerber o da un plugin di Eagle.

Vedendo e provando la stabilità e la precisione della nostra stampante 3Drag, molti di voi ci hanno chiesto se fosse stato possibile montare sulla testa, invece del sistema di deposizione riscaldatore/estrusore, un elettromandrino, in modo da realizzare una macchina CNC. Ebbene, questo è possibile, non a caso abbiamo lavorato alla modifica ed ora vi presentiamo il risultato con una prima applicazione che certo stimolerà l'interesse di molti di voi. Infatti, dopo aver montato l'elettromandrino e testato le qualità di lavorazione, abbiamo provato a realizzare un'applicazione un po' inconsueta ma indubbiamente

suggestiva: incidere i circuiti stampati per via meccanica, invece che chimicamente, come ci è stato insegnato a scuola e come si fa di norma nell'industria. Già, perché questa tecnica, alternativa a quella canonica, può sorprendere per qualità del risultato e può dimostrarsi, se non più pulita (perché l'incisione da parte della fresa comporta la

produzione di una cospicua quantità di sfridi) quantomeno più ecologica, dato che non comporta la necessità di usare e perciò riciclare soluzioni acide (come il percloruro ferrico) basiche (il liquido di sviluppo a base di idrossido di sodio), ecc. L'unico difetto, se vogliamo, è che l'incisione del rame comporta, a causa della forma della sommità della fresa, una minima erosione della vetronite o bachelite componente il circuito stampato, il che causa una certa diminuzione della resistenza meccanica e, laddove lo

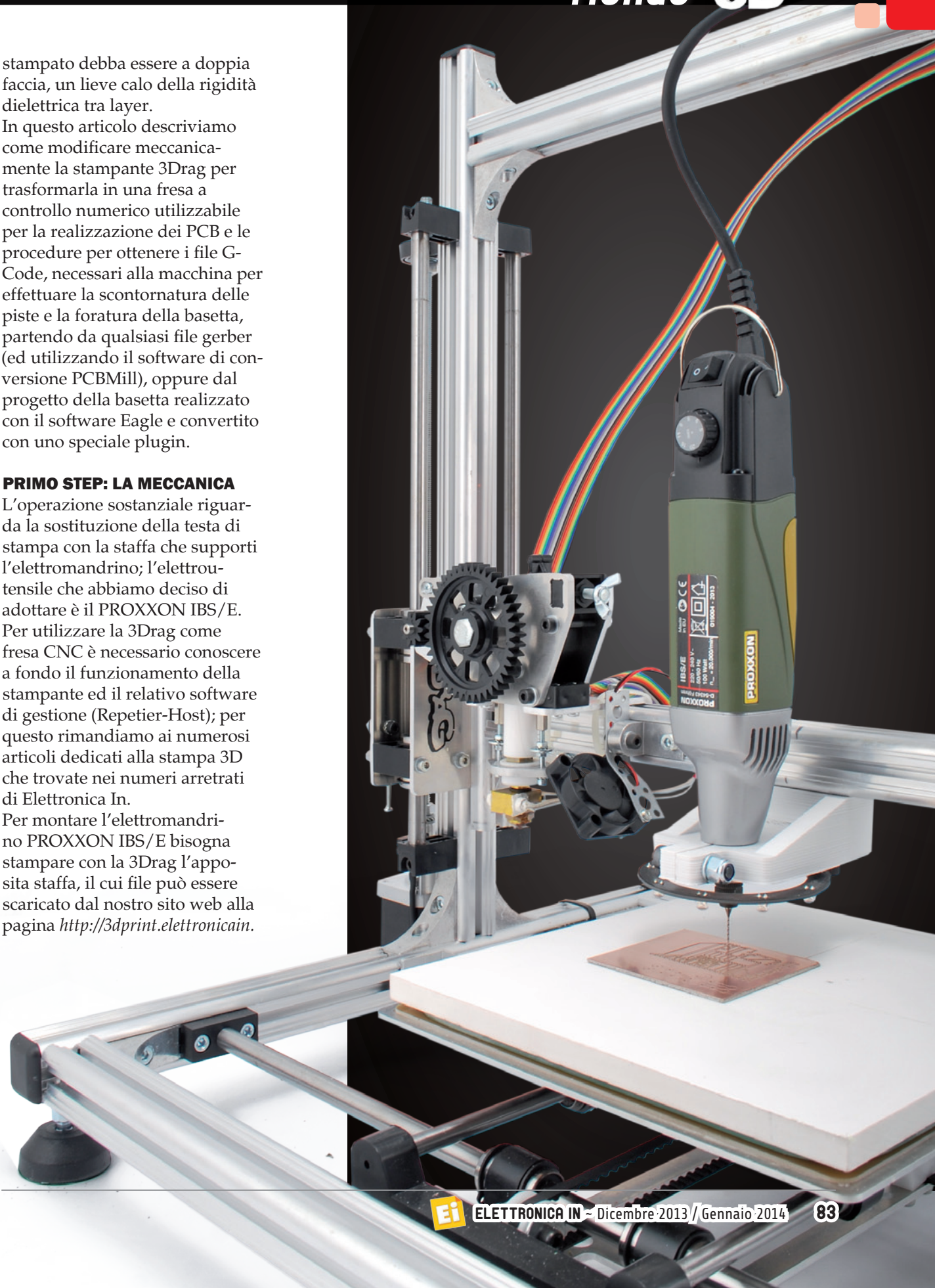
stampato debba essere a doppia faccia, un lieve calo della rigidità dielettrica tra layer.

In questo articolo descriviamo come modificare meccanicamente la stampante 3Drag per trasformarla in una fresa a controllo numerico utilizzabile per la realizzazione dei PCB e le procedure per ottenere i file G-Code, necessari alla macchina per effettuare la scontornatura delle piste e la foratura della basetta, partendo da qualsiasi file gerber (ed utilizzando il software di conversione PCBMill), oppure dal progetto della basetta realizzato con il software Eagle e convertito con uno speciale plugin.

PRIMO STEP: LA MECCANICA

L'operazione sostanziale riguarda la sostituzione della testa di stampa con la staffa che supporti l'elettromandrino; l'elettrotensile che abbiamo deciso di adottare è il PROXXON IBS/E. Per utilizzare la 3Drag come fresa CNC è necessario conoscere a fondo il funzionamento della stampante ed il relativo software di gestione (Repetier-Host); per questo rimandiamo ai numerosi articoli dedicati alla stampa 3D che trovate nei numeri arretrati di Elettronica In.

Per montare l'elettromandrino PROXXON IBS/E bisogna stampare con la 3Drag l'apposita staffa, il cui file può essere scaricato dal nostro sito web alla pagina <http://3dprint.elettronica.in>.



Il necessario per la modifica

Per trasformare la stampante in fresa CNC è necessario prima di tutto avere a disposizione l'attrezzatura, le minuterie metalliche e i vari elementi (visibili in foto) di cui riportiamo l'elenco completo.

Attrezzi:

- Calibro centesimale a corsoio.
- Chiave fissa misura 10 - 11 mm.
- Chiave a brugola misura 4 e 5 mm.
- Squadra con lato da almeno 150 mm.



Minuterie metalliche:

- Pz. 2 viti a testa cilindrica M5x40.
- Pz. 1 vite a testa cilindrica M6x35.
- Pz. 2 rondella dentata M5.
- Pz. 2 rondella piane M5x20.
- Pz. 1 rondella piana M5.
- Pz. 2 rondella piana M6.
- Pz. 1 dado autobloccante M6.

Varie:

- Elettromandrino PROXXON IBS/E (o altro modello simile).
- Staffa per elettromandrino stampata con la 3Drag (con collare idoneo all'elettrotensile).
- Adattatore per fissaggio comparatore, stampato con la 3Drag (opzionale).
- Illuminatore (opzionale; per illuminare il piano di lavoro è possibile utilizzare una comune lampada).
- Lastra di vetro di forma quadrata da 200 mm di lato e circa 3 mm di spessore (è possibile utilizzare anche un normalissimo specchio di forma quadrata acquistabile presso IKEA - modello SÖRLI).
- Piano di forma quadrata, da 200 mm di lato e circa 10 mm di spessore, in legno (MDF, multistrato o simile) o materiale plastico (forex, poliuretano espanso ecc.) privo di gobbe, cunette o altre deformazioni.
- Nastro biadesivo.

it/wp-content/uploads/2013/12/Supporto-mandrino-G-. La staffa, dotata di collare con foro da 20 mm, permette di fissare alla struttura qualsiasi elettromandrino con attacco cilindrico avente lo stesso diametro.

Per utilizzare la 3Drag modificata come fresa CNC, è necessario che il piano di lavoro sia perfettamente complanare con l'utensile al fine di ottenere un'incisione di uguale profondità di tutto il circuito stampato; allo scopo è necessario effettuare una regolazione particolarmente precisa, come descriveremo in seguito. Per tale regolazione potrà anche essere utilizzato un comparatore che andrà montato sulla stessa staffa dell'elettrotensile mediante un adattatore stampabile con la stessa 3Drag e che, ovviamente, andrà realizzato prima di apportare le modifiche meccaniche

descritte in seguito. Il file di tale adattatore è disponibile alla pagina web <http://3dprint.elettronica.in.it/wp-content/uploads/2013/12/Adattatore-Comparatore-G-code.zip>. L'elettromandrino viene alimentato direttamente dalla rete a 220 Vca, quindi non viene gestito dalla 3Drag; dovete accenderlo e spegnerlo manualmente tramite l'apposito interruttore. A corredo, monteremo un anello di LED bianchi che illuminerà la zona in lavorazione.

L'illuminatore può essere alimentato mediante un proprio alimentatore da rete in grado di fornire in uscita una tensione continua di 15 Vcc, oppure può sfruttare lo stesso alimentatore della scheda controller della 3Drag.

Iniziamo con la procedura per modificare la meccanica.

1. Inserite un dado M5 zincato nella cava superiore del

braccio sfruttando l'apertura presente tra il profilo in alluminio e il supporto plastico frontale.

2. Spostate il corpo estrusore svitando le due viti M5 che ne fissano la staffa, ma prima fate un segno con la matita sul braccio della stampante in corrispondenza della mezzzeria della staffa che regge l'estrusore, così che in futuro possiate rimontare l'estrusore.
3. Svitare con una chiave a brugola le due viti di fissaggio che vincolano la staffa al braccio, quindi allentate anche le viti di fissaggio della ventola.

L'Illuminatore a LED.



4. Spostate il corpo estrusore sopra il braccio completamente a sinistra e portate dallo stesso lato anche la ventola: in questo modo l'ugello dell'estrusore verrà allontanato sufficientemente dal piano di lavoro rimanendo bloccato alla struttura.
5. Inserite in un foro della staffa una vite M5, che dovrà essere avvitata nel dado M5 precedentemente introdotto nel braccio.



Inserimento vite M5 nella cava superiore del braccio.



Le viti da rimuovere sono indicate dalle frecce rosse.



La staffa per l'elettromandrino.

6. Stringete le viti per assicurare gli elementi al braccio e fissate nuovamente i cavi con fascette. Conviene ricoprire con una pellicola trasparente l'intero corpo estrusore per ripararlo dalle polveri che si generano durante le operazioni di fresatura del PCB.

7. Prendete la staffa per l'elettromandrino e inseritele una vite a testa cilindrica M6x35 (completa di due rondelle piane e di dado autobloccante M6) che servirà a stringere il collare per bloccare l'elettromandrino.

8. Inserite nel foro dell'illuminatore una vite M5x40 completa di rondella dentata e, dall'altra parte, di rondella piana. Inserire la vite di fissaggio dell'illuminatore in uno dei fori della staffa come visibile nella foto; nell'altro foro, introducete l'altra vite M5x40 completa di rondella dentata e due rondelle piane.

9. Montate sotto il braccio della stampante la staffa che sosterrà l'elettromandrino mediante le due viti a testa cilindrica M5x40 inserite in precedenza e avvitate ai due dadi quadri già presenti nel braccio previsti per il fissaggio dell'estrusore; prima di stringere le viti, posizionate la staffa dell'elettromandrino con la mezzeria spostata di circa 1 cm verso sinistra rispetto al segno di riferimento, riportato a matita sul braccio, in modo che durante l'HOME,

la posizione dell'elettromandrino risulti esterna al piatto (ciò eviterà collisioni con l'utensile).

10. Inserite l'elettrotrusore nel collare e, dopo averlo fissato al supporto plastico, stringete nel mandrino una qualsiasi punta e verificate che, nella posizione di HOME, non vi sia contatto tra l'utensile e il piano di lavoro, quindi stringere le viti della staffa dell'elettromandrino.

11. Applicate sul piatto di stampa, con un po' di biadesivo, una lastra di vetro di forma quadrata da 200 mm di lato e circa 3 mm di spessore. In alternativa è possibile utilizzare anche un comune specchio di forma quadrata; ad esempio il SÖRLI acquistabile presso IKEA.

12. Fissate sul vetro/specchio un piano "sacrificale" di adeguato spessore (circa 1 cm), che servirà, durante la foratura dei PCB, a consentire la discesa dell'utensile senza toccare

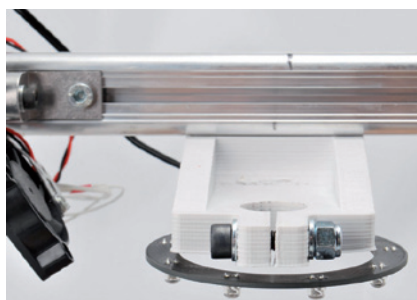
il vetro; il piano può essere in MDF, forex, multistrato, ecc... Lo spessore deve essere tale da tenere sollevato il PCB quanto basta per eseguire le lavorazioni senza che venga azionato il finecorsa Z.

Per poter utilizzare la 3Drag come fresa CNC è anche necessario effettuare l'aggiornamento firmware della scheda elettronica, in modo da consentire all'asse Z di assumere valori negativi rispetto alla posizione di zero; allo scopo, dalla sezione *Software* del nostro sito scaricate il firmware "Marlin versione V1 (Piatto riscaldato ON, LCD ON, Mill ON)" e seguite la relativa procedura di aggiornamento.

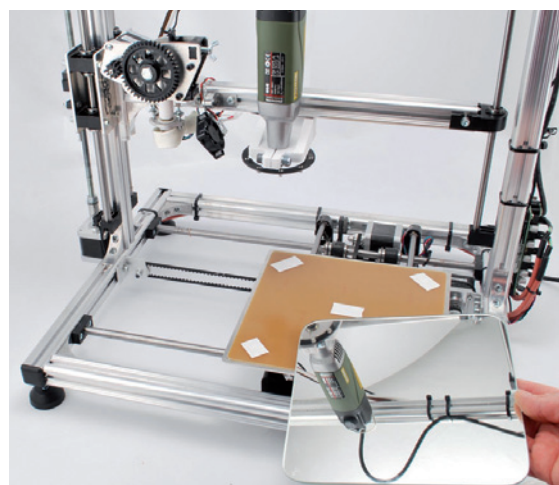
CREARE IL G-CODE PARTENDO DAL FILE GERBER

Per creare un PCB con la 3Drag è necessario avere a disposizione i file G-Code per la scontornatura delle piste e la realizzazione dei fori relativi al circuito stampato stesso. In questa sezione descriviamo come ottenere questi file G-Code partendo da un file gerber (questo è il formato usato dai software di progettazione per PCB); per la conversione utilizziamo il software gratuito PCBMill, fornito da CNCdrive (si trova sotto la sezione Software

Apposizione della lastra di vetro.



L'illuminatore montato sul collare.



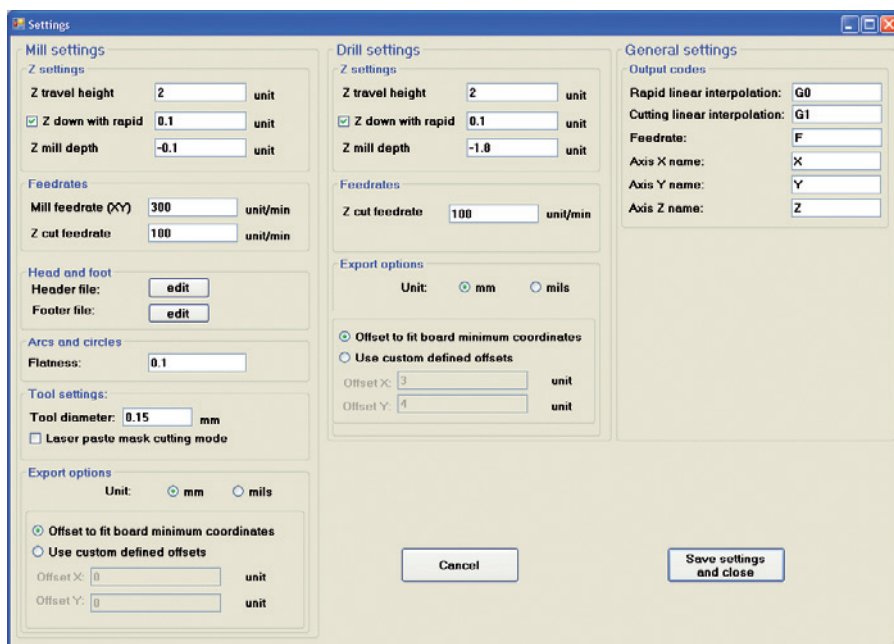


Fig. 1

tools freeware, alla pagina <http://cncdrive.com/downloads.html>), che permette di ottenere file in formato G-Code importabili in Repetier Host. La realizzazione del circuito stampato prevede l'impiego di un bulino per effettuare l'incisione delle piste e di una punta di diametro adeguato per la foratura, da montare prima uno e poi l'altro nell'elettroman-

drino. Scaricato il file, installatelo tramite doppio clic sul file *Setup* e poi lanciatelo mediante un doppio clic sulla relativa icona; prima di tutto è necessario effettuare le impostazioni per ottenere file G-Code corretti. Cliccate su PCBMill -> Settings e si aprirà la finestra di Fig. 1. Il menu *Settings* è diviso in tre parti: la prima (*Mill settings*)

permette di impostare alcuni parametri necessari alla creazione del G-Code relativa all'incisione per le piste, come ad esempio la velocità di avanzamento, la profondità dell'incisione, ecc. Consigliamo di attenersi ai parametri che riportiamo in Fig. 1. *Tool diameter* dipende dalla fresa (bulino) che verrà utilizzata (nel nostro caso 0,15 mm). La seconda parte, ossia *Drill settings*, riguarda le impostazioni per la foratura dei CS; la terza è relativa alla generazione del G-Code (qui dovete lasciare i parametri predefiniti).

Creazione dei file G-Code

Affinché la lavorazione sia corretta, dovete generare un file gerber relativo al "mirror" del circuito stampato, ossia un negativo, in modo che, il lato rame (definito normalmente come Bottom Copper), rimanga rivolto verso l'alto. Dal menu di PCBMill impartite il comando *Open Gerber* e aprite il file relativo alle piste (ad esempio *Bottom Copper.gbr*); automaticamente, il software realizza il G-Code relativo alla scontornatura delle piste (Fig. 2).

I codice può essere visualizzato selezionando la voce *Export -> Milling G-Code* (Fig. 3).

A questo punto bisogna impostare Repetier-Host e preparare il G-Code: aprire Repetier Host, selezionare "CNC Router" (da "Impostazioni Stampante", indicato dalla freccia rossa in Fig. 4) e impostare i parametri come si vede in Fig. 5.

Per visualizzare correttamente sia l'incisione che il percorso dell'utensile, selezionare *Show Filament* e *Show Travel*. Ora importate nel G-Code Editor di Repetier-Host (semplicemente con un copia incolla) il G-Code relativo alla fresatura delle piste, generato precedentemente con PCBMill,

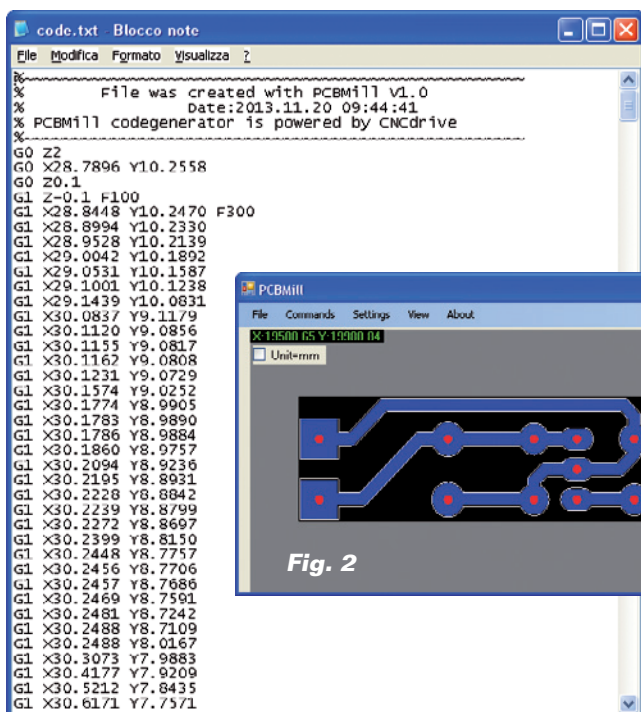


Fig. 3

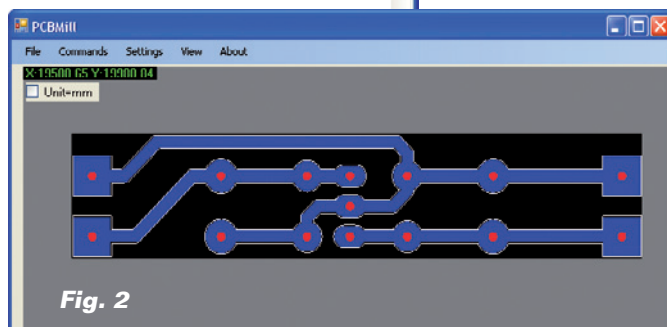


Fig. 2

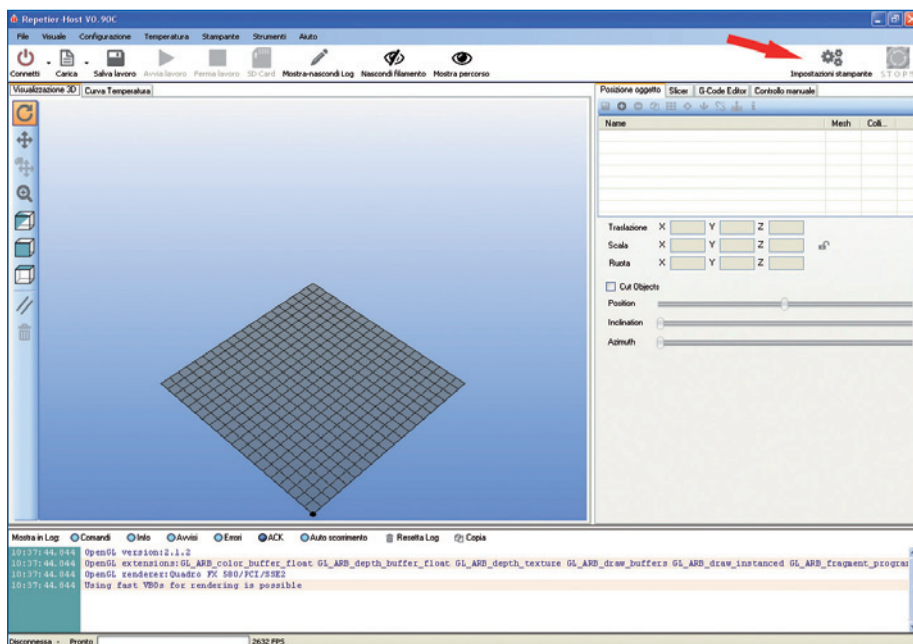


Fig. 4

già mostrato nella Fig. 3. L'unica accortezza è eliminare le prime 5 righe del G-Code, ovvero quelle che iniziano con %:

```
% File created with PCBMill V1.0
% Date:2013.11.19 04:36:26
% PCBMill codegenerator is powered by CNCdrive
%
```

Repetier-Host è in grado di interpretare senza alcun problema il G-Code creato da PCBMill (in cui sono state eliminate manualmente le prime 5 righe), tuttavia, prima di partire con l'incisione, è bene definire il G-Code iniziale e quello finale per imporre alla macchina la posizione di partenza e quella di termine lavoro. Questa operazione è molto semplice da eseguire in Repetier, in quanto è previsto uno Start Code (Codice iniziale) che Repetier lancerà prima di eseguire il G-Code principale (quello di lavorazione della basetta) e un End Code (Codice finale) che verrà eseguito una volta terminato il lavoro. Sulla destra selezionare il tab *G-Code Editor*, aprire il menu a tendina e selezionare *Start Code* (Fig. 6). Inserire le seguenti righe di comando:

G92 X0 Y0 Z0 ; offset del sistema di coordinate (imposta il punto di partenza come home)
G01 F500 ; movimento lineare (imposta una velocità di movimento di 500 mm/min).

La stessa cosa deve essere fatta per l'End Code, inserendo il comando:

G01 X0 Y0 F1200 ; movimento lineare (torna al punto X0 Y0 con una velocità di 1.200 mm/min).

PREPARAZIONE DELLA CNC E INCISIONE DEL PCB

Montate nell'elettromandrino il bulino per l'incisione, con dimensione della punta identica a quella definita nelle impostazioni di PCBMill (consigliamo un bulino in metallo duro, perché la vetronite è altamente abrasiva) lasciandolo sporgere il minimo indispensabile, onde evitare vibrazioni e run-out eccessivi. Avvicinate il più possibile l'utensile al piano sacrificale in modo da sfiorarlo; muovete il piano in tutte le direzioni per verificare che la distanza dall'utensile sia costante. Effettuate una pre-

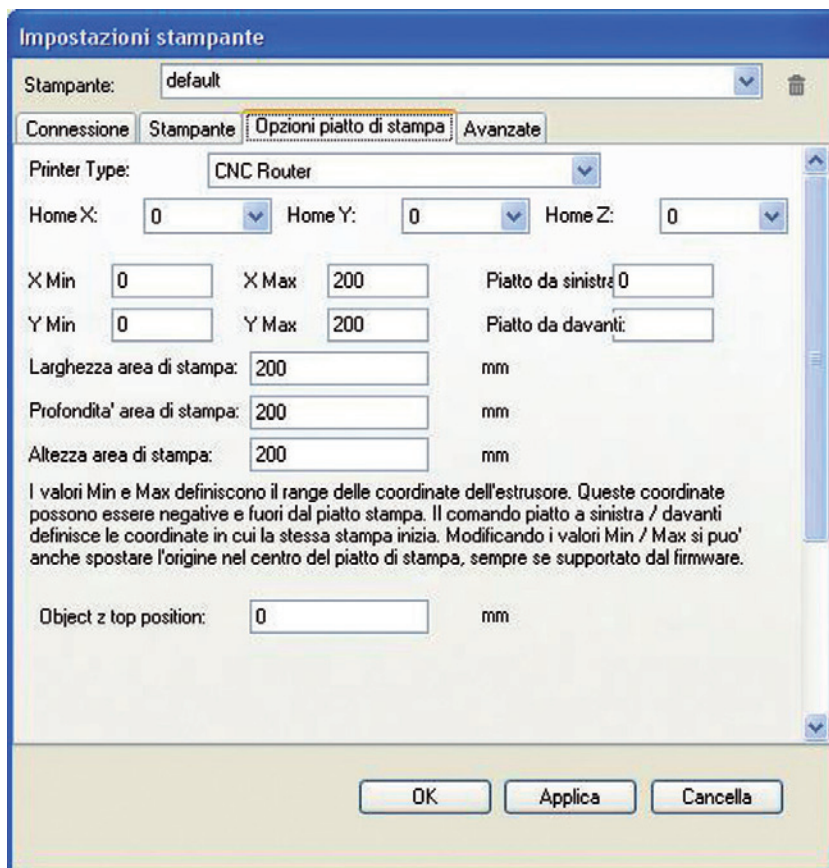


Fig. 5

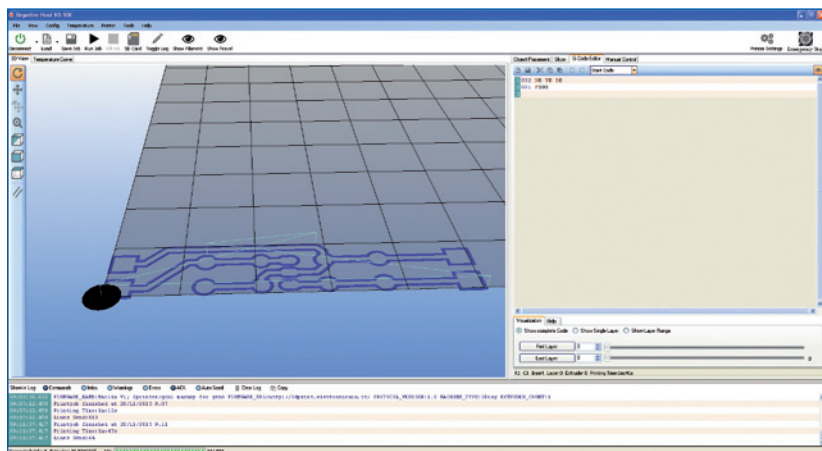


Fig. 6

regolazione del piatto della stampante, agendo sulle apposite manopole, in modo che sia il più possibile complanare con l'utensile (Fig. 7).

Ora potete fissare con del biadesivo, sul piano sacrificale, la bassetta ramata da lavorare. Dopo aver fissato la bassetta col biadesivo, è necessario effettuare una regolazione fine agendo sulle apposite manopole, in modo che la bassetta sia complanare con l'utensile (per eseguire l'operazione è possibile procedere come in precedenza avvicinando la punta dell'utensile alla bassetta quindi muovendo lentamente il piatto in direzione X e Y in modo che la distanza rimanga il più possibile costante su tutta la superficie della bassetta. In alternativa è possibile utilizzare un comparatore con apposito adattatore -quello stampato in precedenza- inserito nel collare

previsto per l'elettromandrino. Iniziamo dunque l'incisione delle piste: abbassate l'asse Z (utilizzando il controllo manuale di Repetier-Host), quanto basta per avvicinare il bulino alla bassetta (circa 2÷3 mm) quindi regolate gli assi X e Y per posizionare l'utensile in corrispondenza dell'origine del PCB che si vuole incidere. Ora, accendete l'elettromandrino, impostate la massima velocità, quindi ruotate manualmente (molto lentamente) l'albero del motore Z per abbassare il braccio della macchina fino a sfiorare con la punta del bulino il PCB. L'azzeramento corretto si ottiene quando l'utensile inizia a "lucidare" il rame sottostante, senza inciderlo.

Nella schermata di Repetier-Host, fate clic sul pulsante *Avvia lavoro*: la macchina eseguirà la scontornatura del PCB come programmato. Terminata l'incisione, l'elettro-utensile verrà riportato esattamente al punto di partenza.

Creazione file G-Code relativo ai fori

Dal menu file di PCBMill selezionare *Open drill*: si aprirà un menu nel quale si dovrà selezionare la voce *Read drill file* (Fig. 8). Caricate il file relativo ai fori del PCB di cui si è effettuata la scontornatura delle piste mediante fresatura (ad esempio *Drill Data.drl*); il software creerà automaticamente il relativo G-Code

mostrando con dei punti rossi i fori che verranno realizzati. Ora, mediante il comando *Export -> Drill G-Code* (Fig. 9) esportate il file. Il file generato da PCBMill (Fig. 10) è aperto in Blocco Note o altro editor di testo predefinito nel sistema operativo, deve essere importato nel G-Code Editor di Repetier-Host, semplicemente con un copia incolla. In questo caso il file importato non deve essere modificato. Nella finestra di visualizzazione 3D appariranno i percorsi dell'utensile di foratura (*Mostra Filamento* e *Mostra percorso* devono essere attivi). Sollevate di una decina di centimetri l'asse Z agendo sui tasti del controllo manuale (attenzione a non muovere X e Y) sostituite il bulino con una punta da trapano per ferro, di adeguato diametro (ad esempio 0,8 mm). Fate scendere nuovamente l'asse Z fino a posizionare la punta a circa 1÷2 mm dalla piastra ramata. Ora accendete l'elettromandrino e ruotate manualmente l'albero del motore Z per abbassare il braccio della macchina fino a sfiorare con la punta il PCB: il punto di contatto dovrebbe coincidere con il punto di contatto precedentemente definito con l'utensile per incisione.

Ora, in Repetier-Host cliccate sul pulsante *Avvia lavoro*; la macchi-

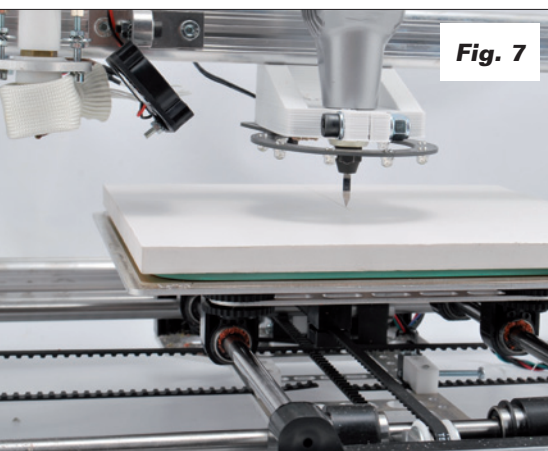


Fig. 7

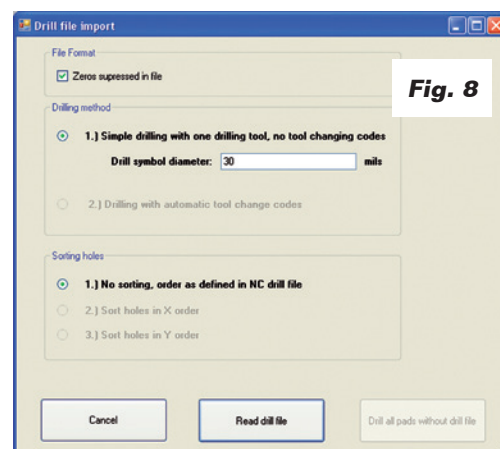


Fig. 8

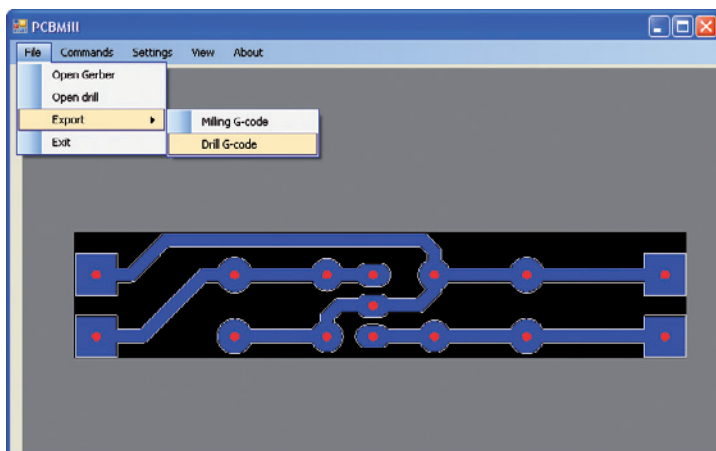


Fig. 9

na eseguirà la foratura prevista per il PCB (Fig. 11) terminata la quale si riposizionerà nelle coordinate X0 Y0.

STAMPARE DA EAGLE

I G-Code possono essere ottenuti anche utilizzando l'apposito plugin di Eagle, il noto programma CAD disponibile anche in versione freeware. Il plugin si chiama PCB-gcode e permette di definire l'estensione del file da esportare (.nc) in modo che sia compatibile con Repetier-Host.

Se non avete già installato EAGLE, andate alla pagina web www.cadsoftusa.com/download-eagle/?language=en e scaricate la versione per il vostro sistema operativo, quindi lanciate il file .exe ed effettuate l'installazione freeware. Poi scaricate PCB-gcode (noi abbiamo usato la versione 3.6.0.4) dal sito <http://groups.yahoo.com/neo/groups/pcb-gcode/files/!%20Software> (è necessario avere un account Yahoo) oppure scaricare dai forum dedicati (ad esempio <http://pcb-gcode.org/read.php?12,803,803>). Il manuale completo di PCB-gcode (pcb-gcode.pdf) è disponibile sotto la cartella "docs" del programma pcb-gcode-3.6.0.4 scaricato.

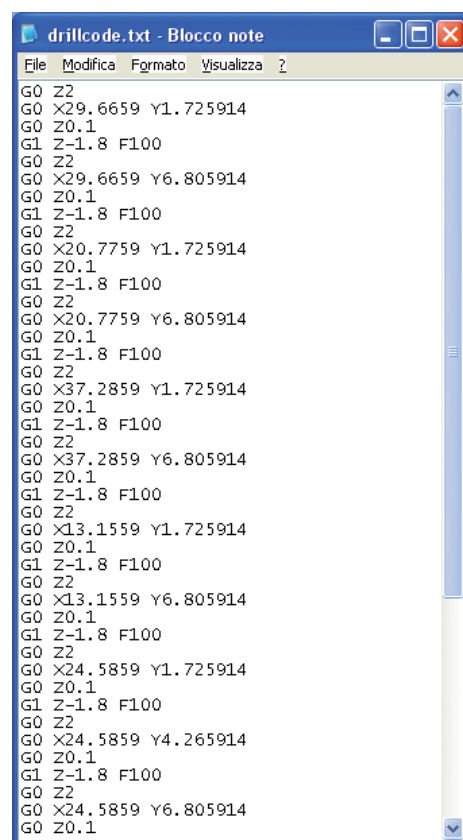
Decomprimete la cartella, quindi copiate i file nella cartella ULP di Eagle; ora avviate quest'ultimo, cliccate su *File/Open/Board* e selezionate il file del PCB che si desidera realizzare mediante

fresatura (Fig. 12).

Effettuate il "mirror" del circuito stampato ed allineate l'angolo inferiore sinistro della basetta con il punto di origine del foglio di lavoro; questa operazione serve per mantenere il PCB all'interno del piano di lavoro e per orientare il PCB in modo che il lato rame da lavorare (definito "Bottom Copper") sia rivolto verso l'alto. Adesso configuriamo i parametri PCB-gcode: avviate PCB-gcode digitando, nella command line, *run pcb-gcode-setup* (Fig. 13).

Dopo l'invio apparirà la finestra in cui sono disponibili diverse cartelle di configurazione (*Generation Options, Machine, GCode Style, GCode options...*); nella cartella *Generation Options* spuntate le due voci sotto *Top Side* e la voce *Single pass* di *Isolation*. Quest'ultima permette di effettuare il contorno delle piste con

Fig. 10



un'unica passata dell'utensile. Impostate un valore per l'isolamento minimo (*Minimum*, cioè la minima distanza tra piste e pad) e la dimensione della punta dell'utensile di fresatura utilizzato (*Etching Tool Size*): ad esempio 0,2mm. Nella cartella *Machine* è necessario impostare, oltre all'unità di misura (mm, Mils,

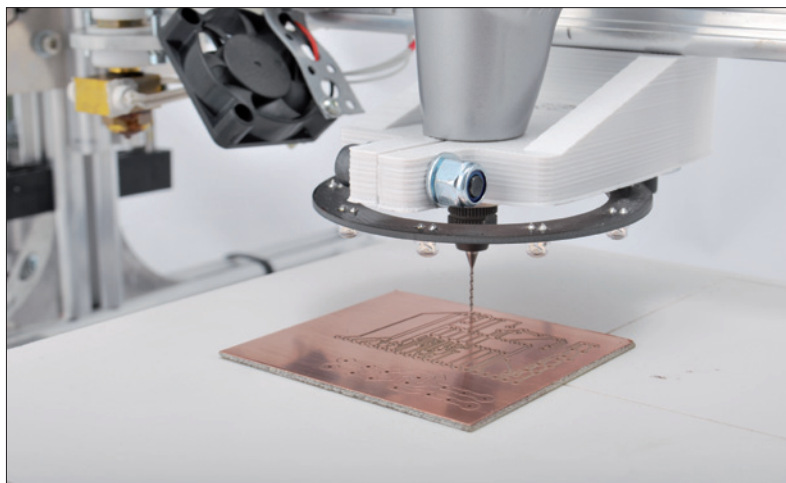


Fig. 11

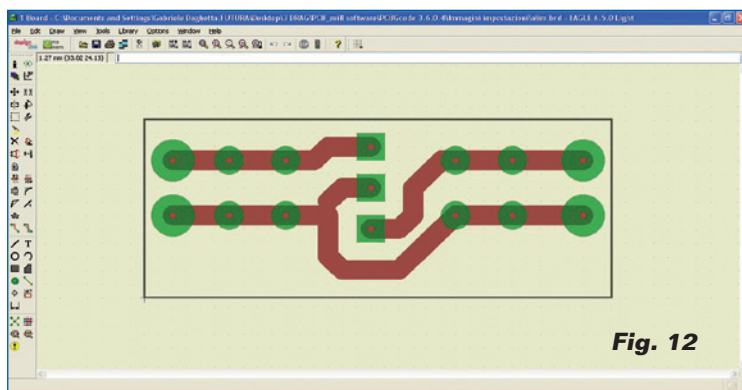


Fig. 12

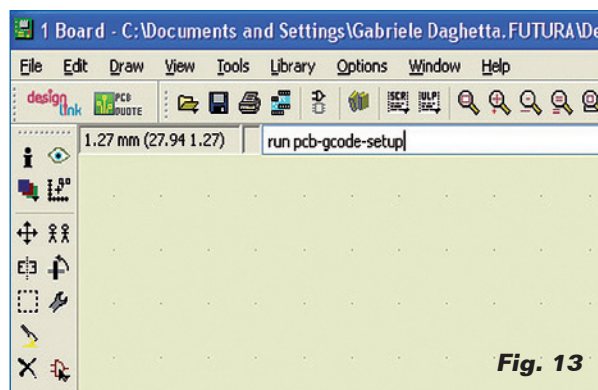


Fig. 13

ecc...), anche i parametri relativi agli avanzamenti della macchina, ovvero i *Feed Rates* (300 mm/min per X e Y e 100 mm/min per Z), i posizionamenti dello Z (Fig. 14):

- *Z High*: sollevamento massimo dello Z (ad esempio 5 mm);
- *Z Up*: sollevamento dello Z durante lo spostamento da un punto all'altro (ad esempio 0,8 mm);
- *Z Down*: profondità di incisione (ad esempio 0,1 mm);
- *Drill Depth*: profondità di foratura (ad esempio 1,8 mm; dipende dallo spessore della bassetta);
- *Drill Dwell*: tempo di attesa

prima della risalita della punta di foratura (ad esempio 0,5 s).

Nella cartella *GCode Styles* impostata lo stile del G-code generato più adatto alla nostra macchina (abbiamo scelto *C:/Program Files (x86)/EAGLE-6.5.0/ulp/profiles/generic.pp*).

La cartella *GCode Options* (Fig. 15) permette di scegliere se inserire commenti nel G-code generato (nel nostro caso non deve essere spuntata alcuna voce), se utilizzare un G-code personalizzato ed altro ancora. Ai fini della realizzazione del nostro PCB, abbiamo spuntato solamente la voce *Use simple drill code* (che permette di ottenere un G-code di foratura compatibile con Repetier-Host) e definito "nc" come *Extension* (sempre per rendere compatibile con Repetier-Host i file generati). Le ultime due cartelle (*Plugins* e *Other*) non sono state prese in considerazione.

Dopo aver impostato i parametri di PCB-gcode, fare clic su *Accept* per confermare le proprie scelte (la finestra di setup si chiuderà e a monitor riapparirà la schermata del Board Editor con il vostro PCB. Per creare i file G-code utili per l'incisione delle piste e la foratura della propria bassetta, digitare nell'apposito campo *run pcb-gcode* e premere Invio.

Fig. 14

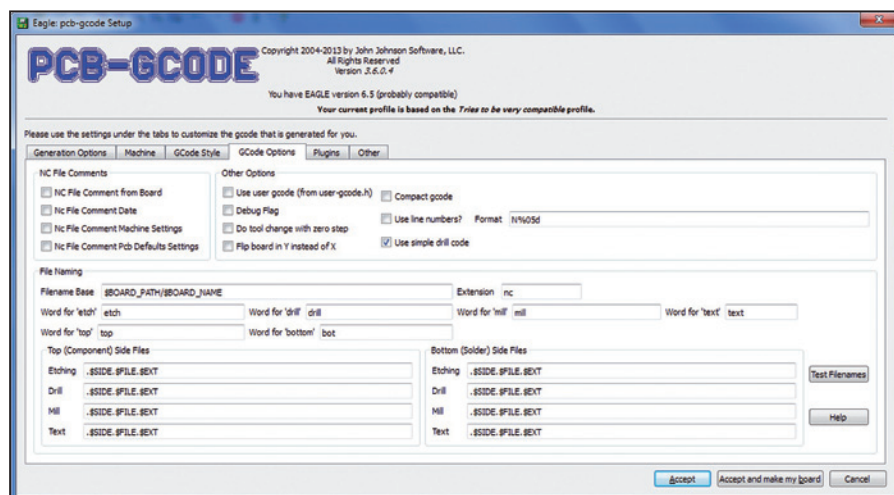
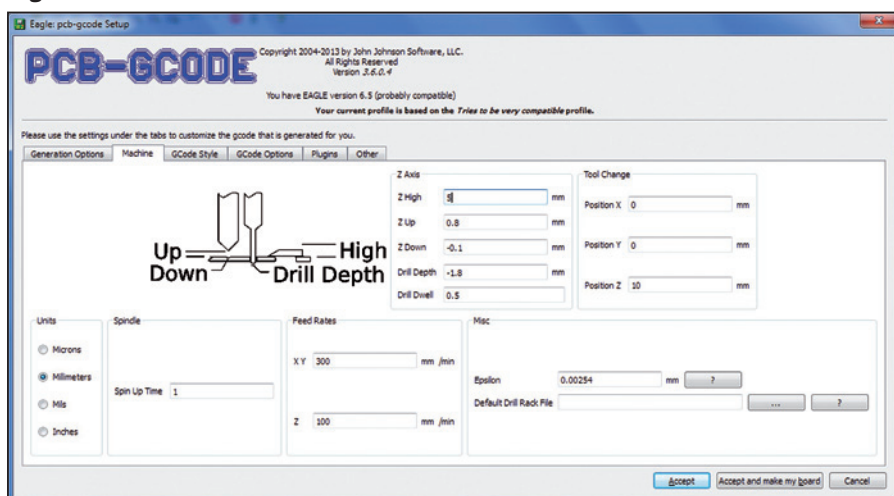


Fig. 15

IMPOSTARE REPETIER-HOST E IMPORTARE/MODIFICARE IL G-CODE

A questo punto aprite Repetier

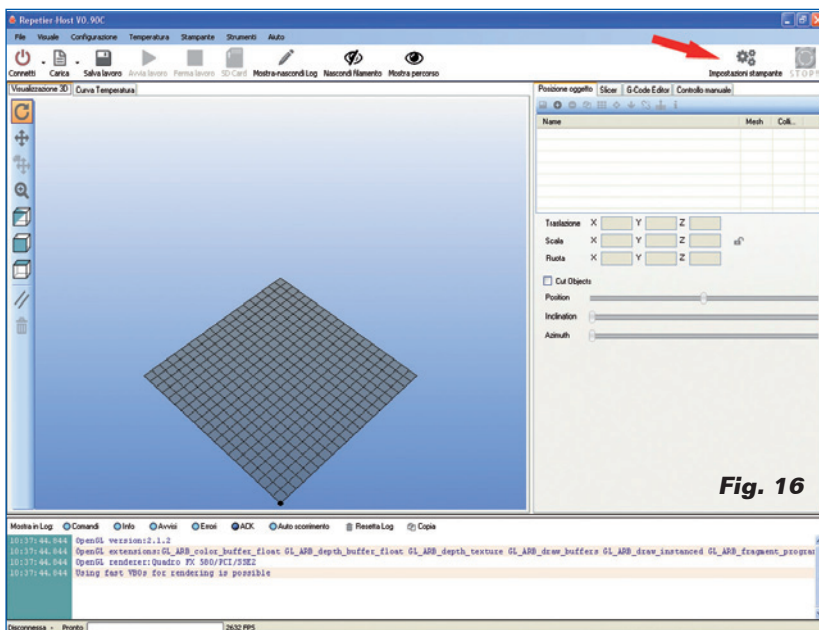


Fig. 16

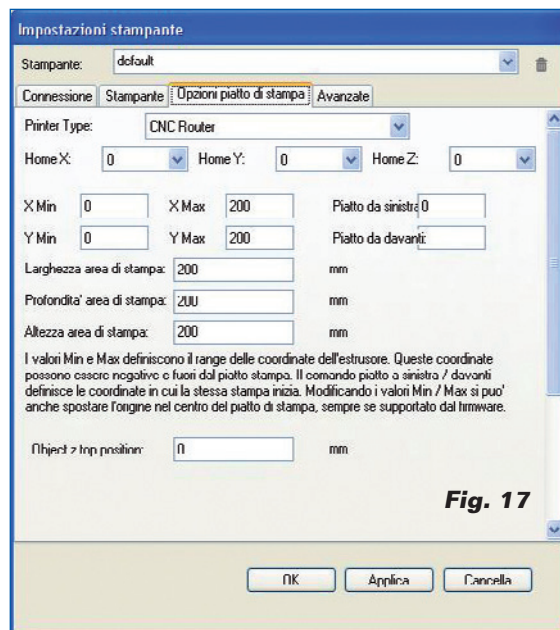


Fig. 17

Host, selezionate **CNC Router** (vi si accede da **Impostazioni Stampante**, in alto a destra, indicato dalla freccia rossa in Fig. 16) e impostate i parametri come in Fig. 17. Caricate in Repetier-Host il **Gcode xxx.top.etch.nc**, ovvero il file corrispondente al percorso utensile del PCB che si desidera realizzare (xxx rappresenta il nome assegnato al PCB, che in questo caso è "alim"). Nella finestra di visualizzazione 3D appariranno i contorni delle tracce del PCB che verrà realizzato mediante fresatura (**Mostra Filamento** e **Mostra percorso** devono essere attivi). Dal G-Code Editor dovete

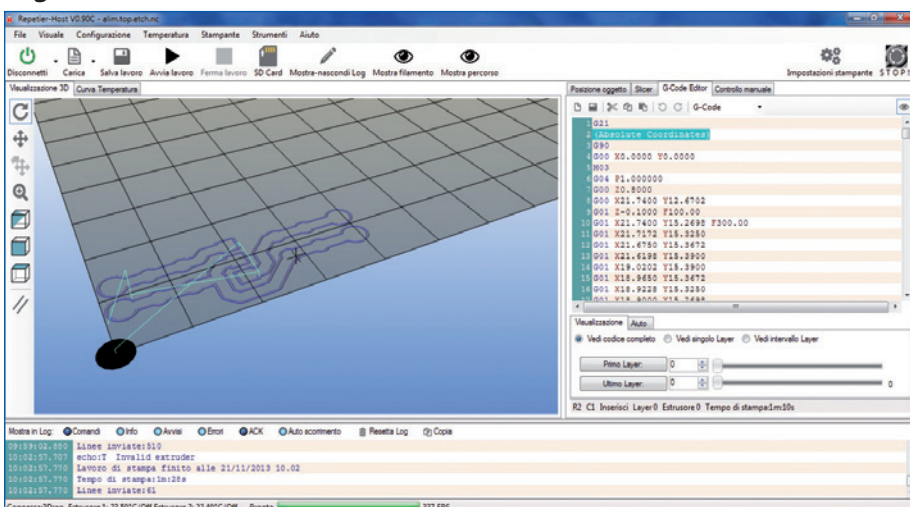
eliminare le righe iniziali in cui compaiono le parentesi tonde (Fig. 18): in questo caso, la seconda riga di comando. Ora dovete cliccare sul menu a tendina del G-Code Editor e selezionare **Codice iniziale**, quindi inserire il GCode iniziale:

- 1^ riga = **G92 X0 Y0 Z0**; offset del sistema di coordinate, imposta X0 Y0 Z0 come punto di partenza;
- 2^ riga = **G01 F500**; movimento lineare con velocità di 500 mm/min.

Ora cliccate sull'icona disco per salvarlo, poi selezionate **G-Code finale**, inserite **G01 X0 Y0 F500**

(torna alla posizione X0 Y0 con movimento lineare di 500 mm/min) e salvate. Queste modifiche servono per definire l'offset del sistema di coordinate in modo che la macchina operi nell'area desiderata e torni al punto di partenza terminata la lavorazione. Impostato Repetier Host, eseguite la lavorazione come già descritto nelle pagine precedenti. Bene, con questo crediamo di avervi detto tutto; se volete maggiori dettagli, li trovate nel tutorial che abbiamo pubblicato alla pagina web <http://3dprint.elettronica.in/la-3drag-per-tuoi-circuiti-stampati/>.

Fig. 18



per il MATERIALE

La stampante 3Drag è disponibile sia in scatola di montaggio che già montata e collaudata. L'attuale versione (1.2) completa di piatto riscaldato è disponibile in kit al prezzo di 680,00 Euro (cod. 3DRAG/K) mentre la versione montata costa 860,00 Euro (cod. 3DRAG/M). I prezzi sono comprensivi di IVA.

Il materiale va richiesto a:
Futura Elettronica, Via Adige 11,
21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
<http://www.futurashop.it>

Abbiamo appena presentato le schede Arduino Yún e Galileo, che già si affaccia una nuova board, molto più potente, sempre Arduino compatibile, e sempre con GNU/Linux a bordo. Questa volta tutta italiana.



UD00

LA SCHEDA ITALIANA "DA CORSA"

di MARCO MAGAGNIN

Non c'è che dire. Ormai è una valanga. Praticamente ogni mese assistiamo all'uscita di una nuova piattaforma che integra l'architettura GNU/Linux con le funzionalità di Arduino, in forma emulata, o con un microcontrollore dedicato. Nel tempo abbiamo presentato pcDuino, Arduino Yún, la scheda Galileo. Ora è la volta di UD00. E già si profila all'orizzonte la disponibilità di Arduino Tre, e chissà quali altre nel frattempo. Il progetto UD00 nasce dalla volontà dei fondatori di realizzare un supporto di apprendimento digitale, che coniugasse potenza di calcolo e si integrasse con la massima facilità con la logica dei microcontrollori. Lo scopo?

Quello di essere d'aiuto nella formazione di una nuova generazione di progettisti, tecnici e sviluppatori software con le conoscenze necessarie a realizzare progetti nel campo delle tecnologie digitali, nel physical computing, nel multimediale, nelle arti interattive, nell'Internet delle cose, e così via. Il tutto a prezzi accessibili. Il progetto Udoo nasce in Italia, grazie alle società Seco di Arezzo e Aidlab di Siena, ma vanta numerose collaborazioni internazionali. A coordinare il progetto ci sono Bruno Sinopoli, Marco Sogli e Andrea Camposarcone. Anche la scelta fatta dal gruppo di lavoro, su come finanziare il progetto, è nel segno dell'innovazione. Infatti è stata presenta-

ta una richiesta di finanziamento per 27.000 dollari su kickstarter.com: in 60 giorni ne sono stati raccolti ben 641 mila da 4.172 sottoscrittori.

Eccola dunque qua, anzi, eccole qua, in quanto il progetto UDOO offre tre possibili variazioni sul tema. UDOO è un mini computer che integra un potente chip Freescale a due o quattro core realizzato in tecnologia ARM Cortex-A9 con clock a 1GHz (sul quale può "girare" sia GNU/Linux che Android), col microcontrollore Atmel SAM3X8E ARM Cortex-M3, quello di Arduino Due, realizzando una potente piattaforma per impieghi universali. UDOO dispone anche di uscita video, acceleratore grafico, audio in ingresso ed in uscita, Ethernet a 1

Gbit, RAM DDR da 1 GB, 76 I/O digitali, un modulo WiFi e altro ancora.

CARATTERISTICHE TECNICHE

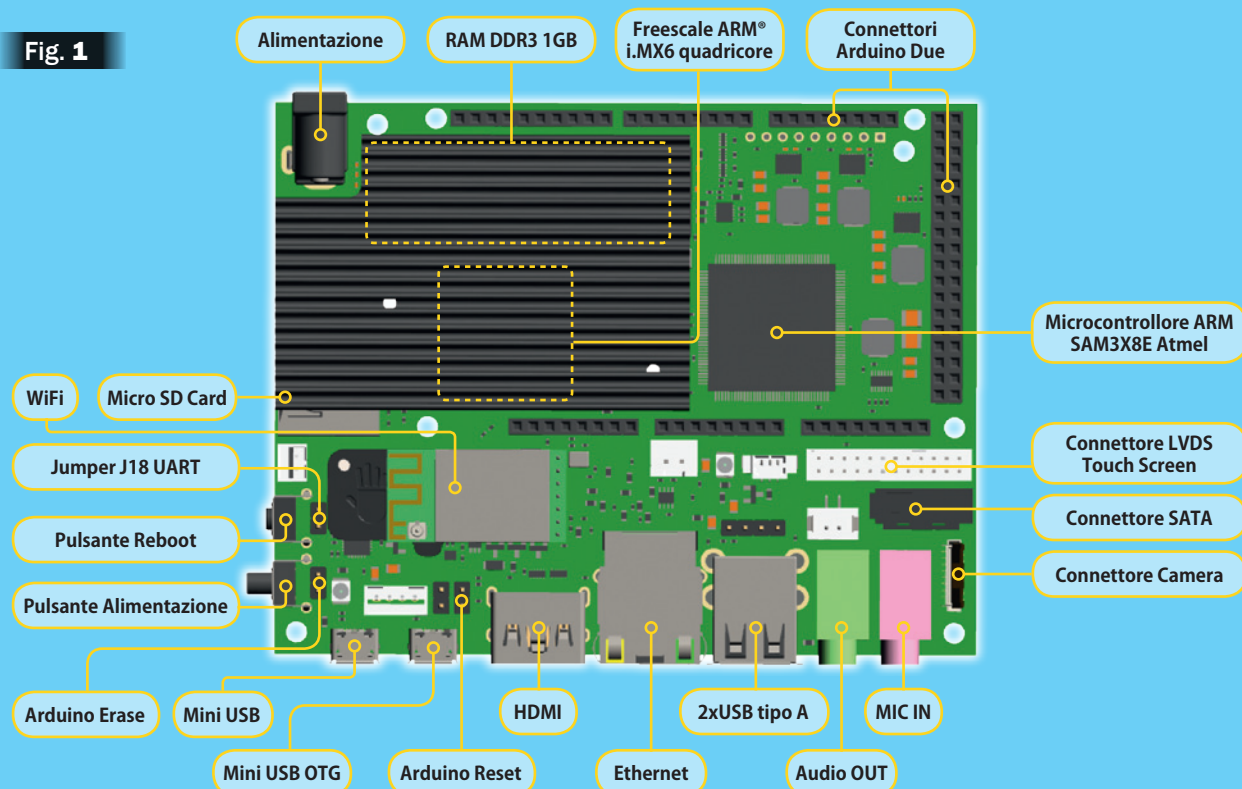
La scheda UDOO è disponibile in tre versioni, la UDOO Dual Basic, la UDOO Dual e la UDOO Quad. Noi abbiamo provato direttamente la Quad, che è la più completa e performante ma anche la più costosa. La disponibilità di tre versioni consente di poter scegliere la scheda più adatta al progetto in sviluppo, in base alla miglior combinazione di potenza, funzionalità e prezzo. Le foto a corredo della **Tabella 1** mostrano le tre schede disponibili mentre all'interno della tabella stessa sono descritte le caratteristiche tecniche delle tre board. Tutte e

tre possono ospitare come sistema operativo sia le distribuzioni di GNU/Linux, Linaro Ubuntu, Yocto e Debian, che Android, nella versione 4.2.2 Jelly Bean. Vediamo dunque in cosa si differenziano le tre schede. Nella Quad è disponibile un connettore SATA per il collegamento di un disco fisso esterno. In questo modo possiamo trasformare la UDOO in un vero e proprio PC. Le altre due schede della serie non hanno connettore SATA. L'altra significativa differenza tra la scheda Quad e le altre è il processore, un Freescale i.MX6Quad con quattro CPU Cortex A9 per la Quad mentre le altre montano il Freescale i.MX6DualLite con due CPU Cortex A9. Infine la differenza tra la scheda Dual e la Dual Basic



Tabella 1

Caratteristiche	UD00 Quad	UD00 Dual	UD00 Dual Basic
CPU	Freescale i.MX6Quad, 4 x ARM® Cortex™-A9 core @ 1GHz	Freescale i.MX6DualLite, 2x ARM® Cortex™-A9 core @ 1GHz	
GPU	GPU Vivante GC 2000 per grafica 3D + Vivante GC 355 per grafica vettoriale 2D +Vivante GC 320 per composizione 2D	GPU Vivante GC 880 per grafica 3D e 2D (grafica vettoriale) + Vivante GC 320 per composizione 2D	
Chip Arduino	Atmel SAM3X8E ARM Cortex-M3 CPU (lo stesso di Arduino Due)		
RAM	RAM DDR3 1GB		
I/O	76 pin GPIO compatibili anche con la configurazione di Arduino Uno R3		
Video	HDMI e LVDS + Touch		
Micro USB	2 Micro USB (1 OTG)		
USB	2 USB 2.0 tipo A e 1 USB 2.0 collegabile al connettore interno		
Audio	jack di ingresso ed uscita audio analogici		
Videocamera	Connettore per videocamera CSI		
Micro SD Card	Lettore di micro SD card integrato, utilizzabile come dispositivo di boot		
Alimentazione	Connettori per l'alimentazione esterna (6-15V) e per una batteria tampone		
Network	Ethernet RJ45 (10/100/1000 MBit)		NO
Wireless	Modulo WiFi		NO
SATA	Connettore SATA con presa di alimentazione		NO

Fig. 1

risiede nella mancata presenza, in quest'ultima, del modulo WiFi. Tutte le tre schede si prestano ad essere utilizzate come PC, con interfaccia grafica desktop, video, mouse e tastiera. In **Fig. 1** sono indicate le principali periferiche presenti sulla scheda. Sottolineiamo solamente la presenza di jack audio analogici sia di ingresso che di uscita, il connettore SATA ed i pulsanti di reset del microcontrollore Arduino e di interruzione dell'alimentazione della scheda. Descriviamo ora la modalità di condivisione delle risorse tra il processore Freescale ed il microcontrollore Atmel. La comunicazione "diretta" tra i due avviene attraverso il collegamento delle rispettive porte seriali. I pin "0" e "1" dell'UART seriale di Arduino sono fisicamente collegati ai pin KEYROW0 and KEYCOL0 del processore Freescale, alla velocità di 115200 bps. Dal lato GNU/Linux la connessione seriale è riconosciuta con il nome di device

/dev/ttymx3. Questa non è assegnata alla console di sistema, come avviene nel caso di Arduino Yún, ma è una connessione seriale diretta che può essere utilizzata per comunicare tra lo sketch di Arduino ed i programmi in esecuzione nel sistema operativo GNU/Linux. L'altra importante caratteristica è la condivisione dei pin di I/O. Entrambe le sezioni di UDOO lavorano con i livelli a 3,3V ed i pin di Arduino sono fisicamente collegati ad altrettanti pin di I/O del processore Freescale. Questo significa che, rispettando determinate condizioni di non conflittualità, tutti i pin possono essere letti e scritti, sia dallo sketch di Arduino, che dai programmi in esecuzione nel sistema operativo GNU/Linux. Per quanto concerne il nostro modo di progettare applicazioni "embedded", questa funzionalità permette una notevole capacità di elaborazione parallela e di conseguenza una grande fles-

sibilità nella progettazione. Dal lato Arduino la mappatura dei pin di I/O corrisponde a quella di una scheda Arduino Due. Dal lato GNU/Linux, per quanto abbiamo potuto sperimentare fino ad ora, l'utilizzo dei pin di I/O rispecchia le specifiche previste per la scheda Raspberry Pi, con la differenza che i pin sono 76 e la potenza di calcolo risulta essere da due a quattro volte superiore. Nella mappa di **Fig. 2** possiamo vedere le corrispondenze tra i pin di Arduino ed i pin collegati al processore Freescale. Nella **Fig. 3** possiamo vedere le "regole" da rispettare nella condivisione dei pin tra le due sezioni. Per fare un esempio, prendiamo il pin D13 che è collegato al pin GPIO40 del processore Freescale. Le diverse possibilità di configurazione possono essere così riassunte.

- Arduino vede D13 come output mentre per il processore Freescale il pin GPIO40 risulta essere un input; in questo

modo i programmi in elaborazione in GNU/Linux possono essere avvisati sui cambiamenti operati da Arduino; in questa configurazione può essere collegato un carico esterno, ad esempio un relè, sempre comandato da Arduino;

- Arduino vede D13 come input mentre per il processore Freescale il pin GPIO40 risulta essere un output. In questo caso sono i programmi in elaborazione in GNU/Linux che comandano. Lo sketch di Arduino può solo leggere lo stato del pin. Anche in questa configurazione può essere collegato un carico esterno, che viene pilotato dai programmi GNU/Linux;
- Entrambi gli ambienti vedono i rispettivi pin come ingressi, all'esterno colleghiamo un sensore che, in questa configurazione può essere "letto" contemporaneamente da entrambi gli ambienti. Un buon metodo per realizzare un "trigger" per innescare eventi che devono

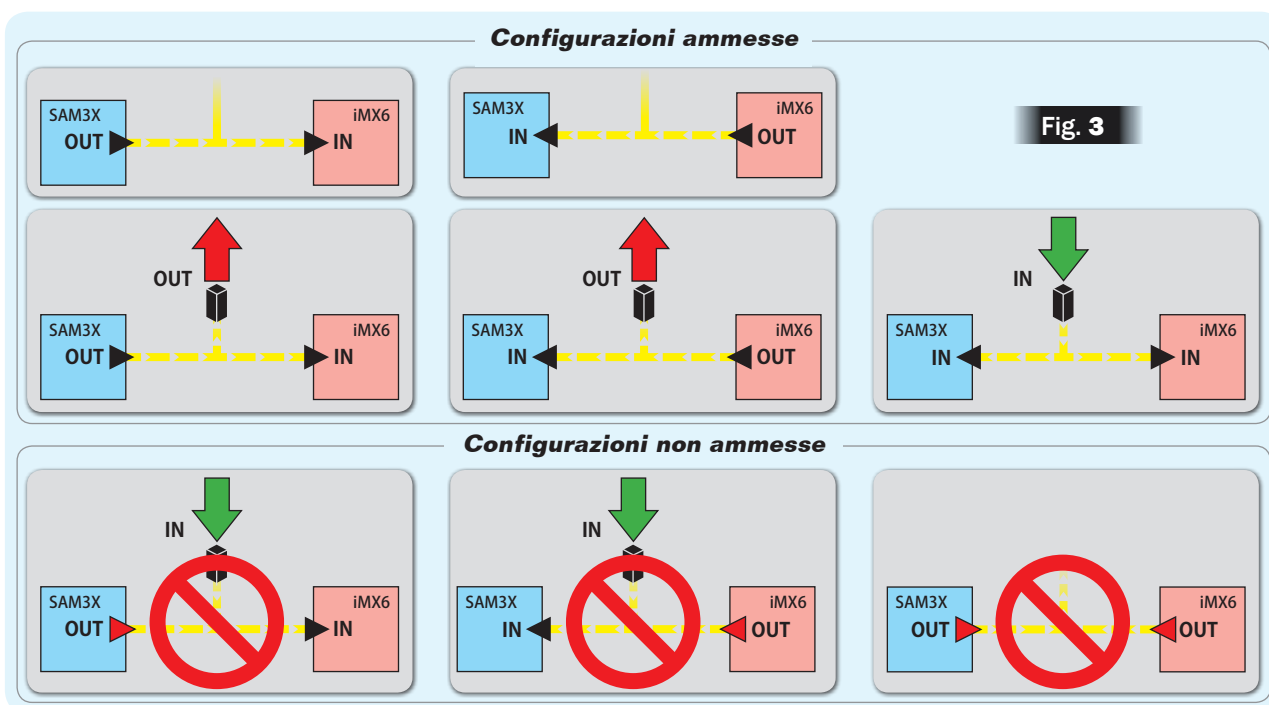
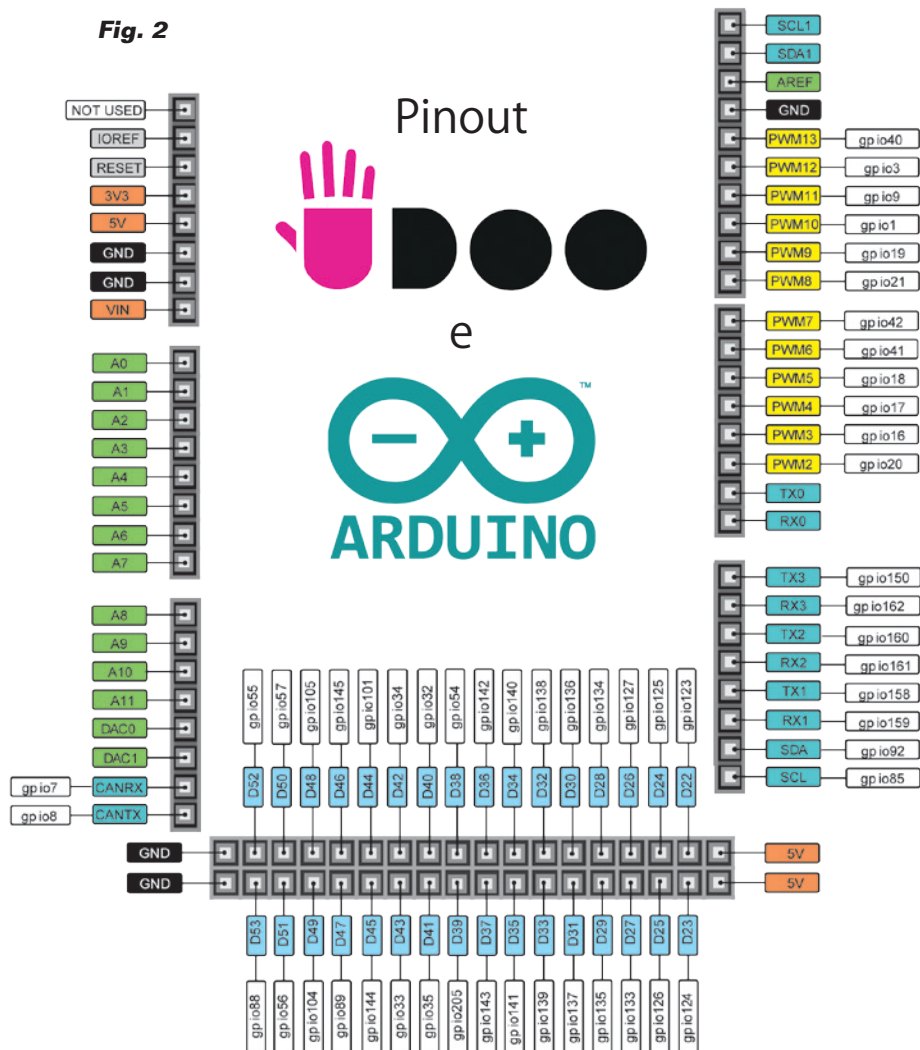


Fig. 3

essere intercettati in contemporanea sia dallo sketch di Arduino che dai programmi GNU/Linux;

- Tutte le altre configurazioni generano conflitti e non possono essere utilizzate.

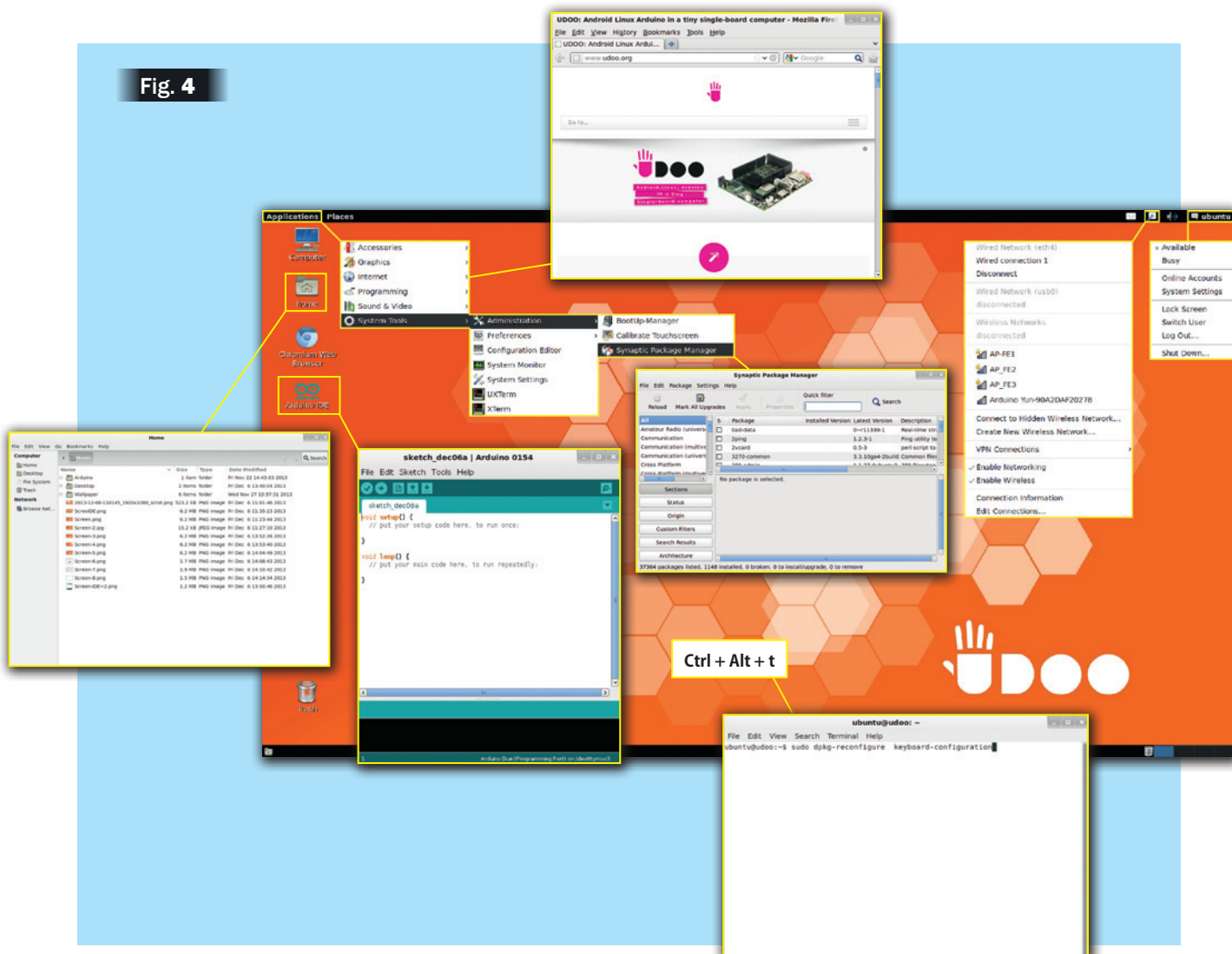
METTIAMO IN FUNZIONE LA SCHEDA UDOO

La scheda UDOO, per essere utilizzabile, richiede la presenza di un sistema operativo compatibile. Sono disponibili alcune distribuzioni di GNU/Linux della famiglia Debian, in particolare Debian stesso, alcune versioni di Ubuntu Linaro, la distribuzione

Yocto e Android Jelly Bean 4.2.2. Il boot del sistema operativo avviene dalla micro SD Card, ma sono possibili altre opzioni. Per questa anteprima abbiamo scelto di utilizzare la più potente e completa delle schede UDOO, la versione Quad. Nel caso della UDOO Quad, per esempio, è possibile eseguire il boot anche dal disco fisso collegato alla porta SATA. Nel futuro non mancheremo di descrivere questa ed altre innumerevoli funzionalità offerte da UDOO. Come prima operazione dobbiamo preparare una micro SD Card con il sistema operativo prescelto. Anche in questo caso

vi sono diversi modi per eseguire l'operazione. Dallo scaricare l'immagine del sistema operativo prescelto, alla "composizione" del filesystem partendo dai file binari delle singole partizioni per finire con la compilazione del kernel a partire dai sorgenti. Possibilità di studio e sperimentazione a tutti i livelli. Noi abbiamo preferito la via "breve", scaricando dalla sezione "Downloads" >> "SD Images" del sito di UDOO la versione di Ubuntu Linaro per la versione Quad della scheda. Al momento della nostra prova la versione più recente disponibile era la "udoo_quad_ubuntu1204_v2.0.zip". Una

Fig. 4



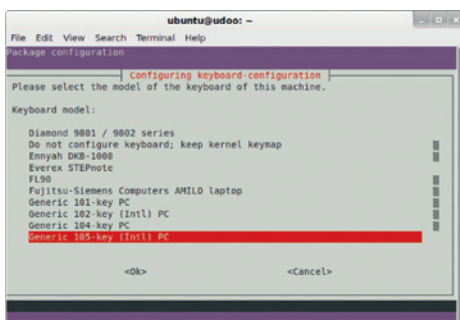


Fig. 5

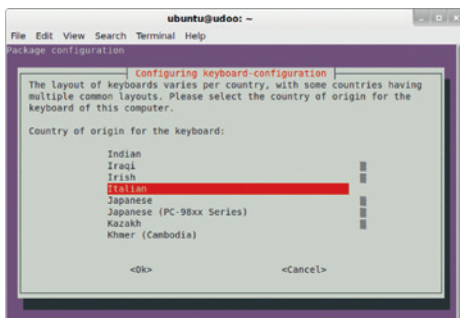


Fig. 6

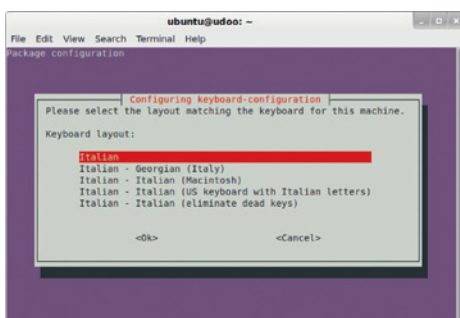


Fig. 7

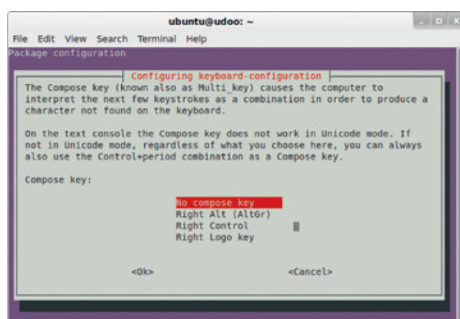


Fig. 8

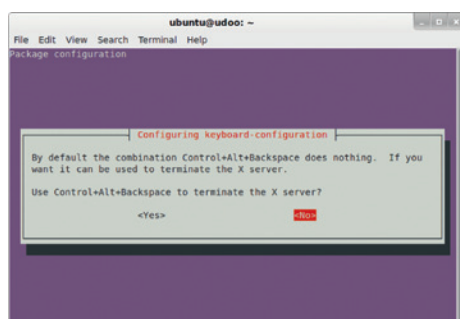
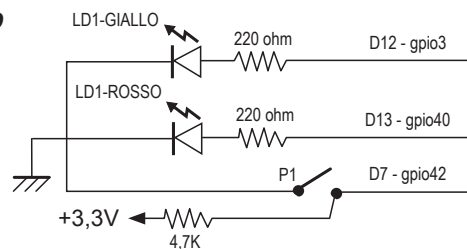


Fig. 9

volta scaricata la scompattiamo in una cartella e, se utilizziamo un PC con un sistema operativo di Microsoft, utilizziamo il tool Win32DiskImager, per trasferire l'immagine scaricata sulla micro SD Card. Noi abbiamo utilizzato una micro SD Card da 8 GB, in quanto l'immagine è di 6 GB. Al termine dell'operazione inseriamo la Micro SD Card nell'apposito slot della scheda UDOO, colleghiamo il cavo HDMI alla scheda ed al monitor e una tastiera ed un mouse alle porte USB disponibili. Colleghiamo il cavo di rete alla porta Ethernet. Controlliamo tutti i collegamenti ancora una volta e ... diamo tensione. A tal scopo abbiamo utilizzato l'alimentatore da 12V acquistato dallo shop di UDOO, con il connettore a barilotto da 5 mm di diametro esterno e 2,5 mm interno. Potete utilizzare qualsiasi alimentatore con le stesse caratteristiche che possa erogare una corrente di almeno 2A. Diamo il tempo al sistema operativo di caricarsi e ci troveremo di fronte alla videata visibile sullo sfondo di Fig. 4, dove abbiamo "riassunto" la navigazione principale del desktop. Una prima piacevole sorpresa: a sinistra possiamo vedere l'icona con il logo di Arduino. Sì, è proprio l'IDE di sviluppo per la sezione Arduino della scheda. Nessuna necessità, dunque, di collegamenti via USB con il PC per la programmazione della sezione Arduino. La scheda UDOO è un vero e proprio PC autonomo. È dotata di un suo desktop dal quale possono essere lanciati e utilizzati gli applicativi disponibili. All'inizio la dotazione di pacchetti è abbastanza scarsa ma decisamente sufficiente per gestire la scheda da subito. Ovviamente possiamo installare ciò che desideriamo con il gestore di pacchetti "Synaptic Package Manager". Questo, come tutti i pacchetti installati, si esegue

Fig. 10



dal menu "Applications", in alto a sinistra sul desktop. L'utente predefinito nella distribuzione è "ubuntu" ed anche la password è "ubuntu". Con la combinazione di tasti Ctrl+Alt+T, apriamo una finestra di terminale sul monitor. Possiamo utilizzarlo per impostare l'utente root con il comando:

```
sudo passwd
```

annotiamoci la password perchè, come abbiamo già descritto in diversi articoli, l'utente root (Super User) è l'utente con i privilegi di accesso su ogni componente del sistema. Perdere la relativa password impedisce la gestione e amministrazione del sistema. In aggiunta, recuperare la password di root, in un sistema GNU/Linux, è pressoché impossibile. Approfittiamo della finestra di terminale aperta per configurare la tastiera italiana. Nella finestra di terminale digitiamo:

```
sudo dpkg-reconfigure keyboard-configuration
```

come visibile in Fig. 4 (videata in basso). Il carattere "-" dovrebbe corrispondere al segno di "apice". Seguiamo tutto il percorso di configurazione come mostrato da Fig. 5 a Fig. 9 e al termine ci troveremo con una configurazione della tastiera più "confortevole". Ora sperimentiamo la gestione dei pin di I/O da parte del sistema operativo GNU/Linux. In queste prove utilizzeremo i pin unicamente come ingressi ed uscite digitali. È possibile utilizzare alcuni pin anche per le altre

funzionalità come quelle elencate di seguito, ma questo richiede la modifica di file di configurazione del kernel e la ricompilazione dello stesso a partire dai sorgenti, un argomento un po' troppo complesso per essere trattato in poche pagine di un articolo.

- UART: uart1, uart3, uart4, uart5
- sd1
- SPI: spi1, spi2, spi5
- i2c1
- spdif
- timer capture
- timer compare
- WATCHDOG FUNCTIONALITIES: watchdog reset, watchdog out
- clock out
- PWM: pwm1, pwm2, pwm3, pwm4
- digital audio

Rimaniamo su un terreno più semplice. Nella configurazione del sistema operativo comunemente distribuita, i pin sono tutti configurati come ingressi/uscite digitali ed il driver di accesso ne permette la gestione per mezzo dei file di device con lo stesso metodo utilizzato per Raspberry Pi. Nella cartella:

```
/sys/class/gpio
```

sono presenti le sottocartelle di gestione di ciascuno dei pin di I/O gestibili, il nome delle cartelle è composto dal prefisso "gpio" più il numero del pin interessato. Ciascuna sottocartella contiene, tra gli altri, i file di gestione del pin, in particolare:

- direction – in questo file il valore "in" indica che il pin è di ingresso, il valore "out" che è di uscita;
- value – contiene il valore corrispondente al livello del pin. Per i pin di ingresso può

essere solo letto, mentre per i pin di uscita può essere anche impostato. Il valore "0" indica il livello basso ("0V"), mentre il valore "1" indica il livello alto ("3,3V");

- edge – permette di impostare il modo di innesco di un evento collegato allo stato del pin. Il valore "none" disabilita la funzione, il valore "rising" innesca l'evento sul fronte di salita del segnale sul pin, il valore "falling" innesca l'evento sul fronte di discesa mentre il valore "both" innesca l'evento su entrambi i fronti.

Per una prova reale della gestione del GPIO sia in GNU/Linux che in Arduino - ma anche in condivisione - abbiamo realizzato il piccolo circuito visibile nello schema di Fig. 10. Lo abbiamo montato utilizzando lo shield di sperimentazione di Arduino con i componenti montati su una bread-

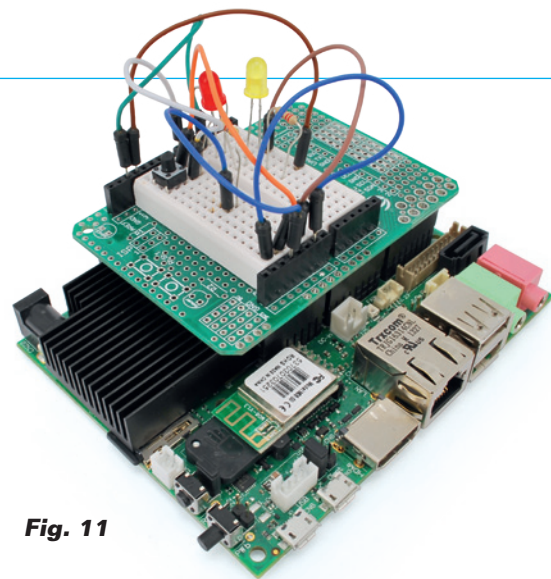


Fig. 11

board millefori. Il risultato finale è visibile nella foto di Fig. 11. Prima proviamo a digitare i comandi dalla finestra di terminale per gestire il LED rosso. Impostiamo il pin GPIO40 come output:

```
echo "out" > /sys/class/gpio/gpio40/
direction
```

Accendiamo il LED:

```
echo "1" > /sys/class/gpio/gpio40/
value
```

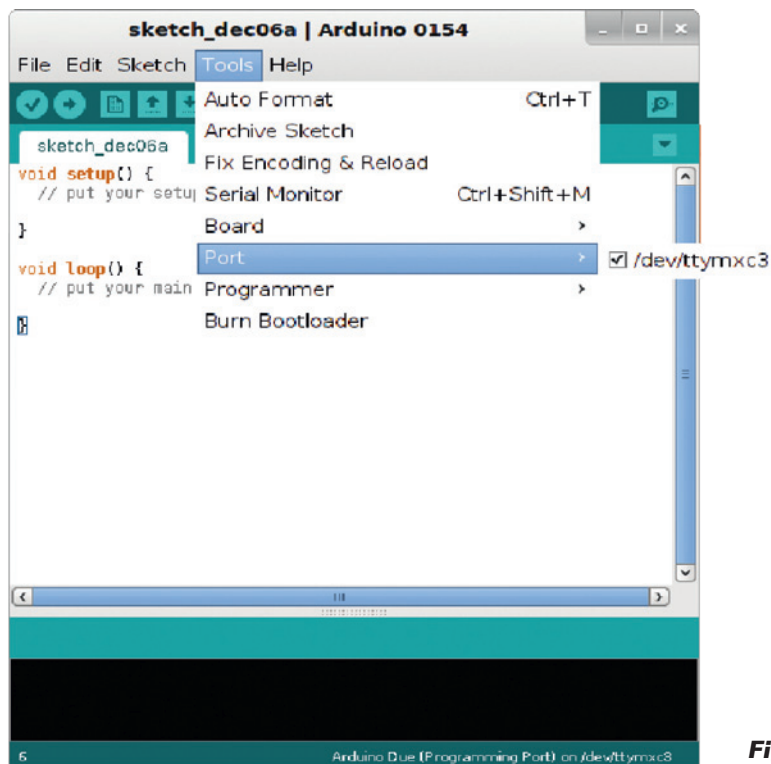
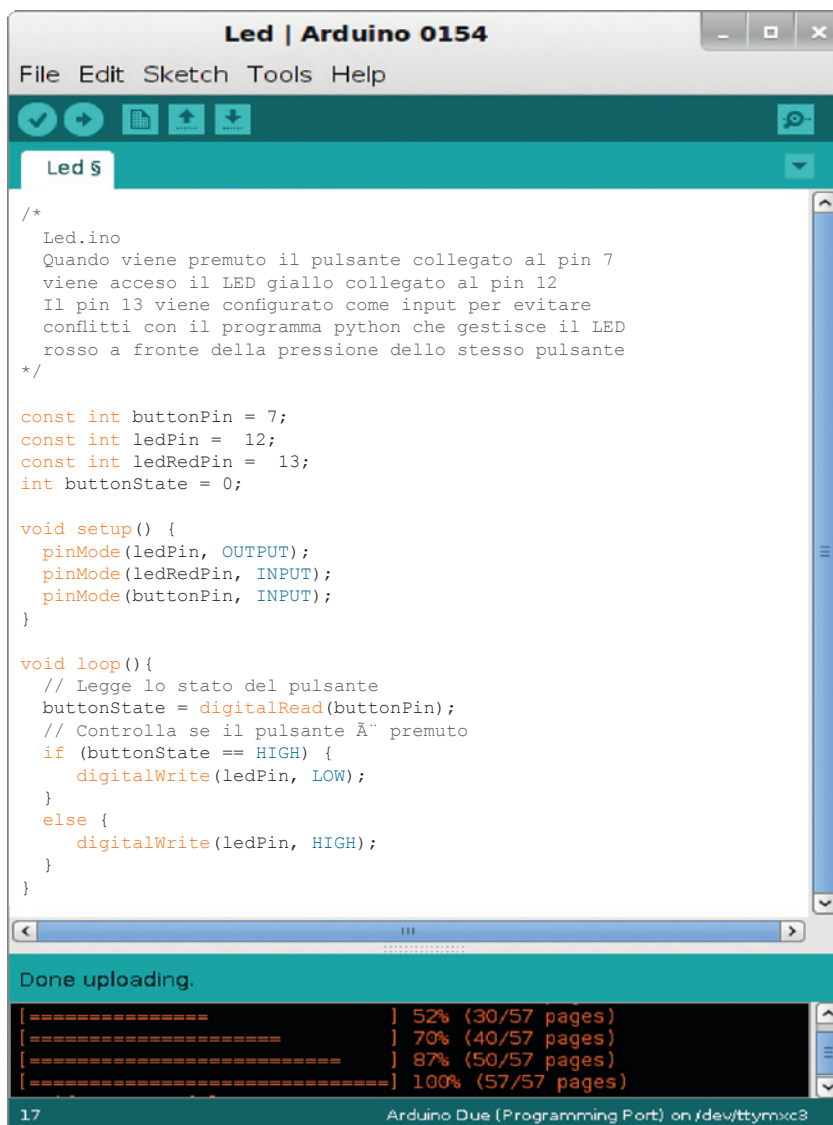


Fig. 12

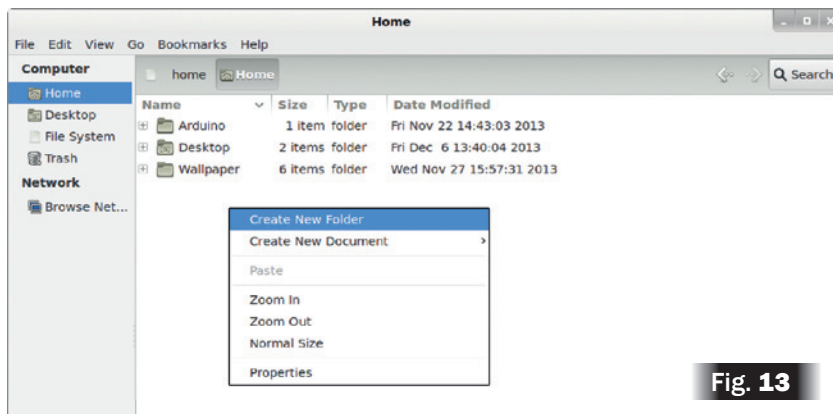


Spegniamo il LED:

```
echo "0" > /sys/class/gpio/gpio40/
value
```

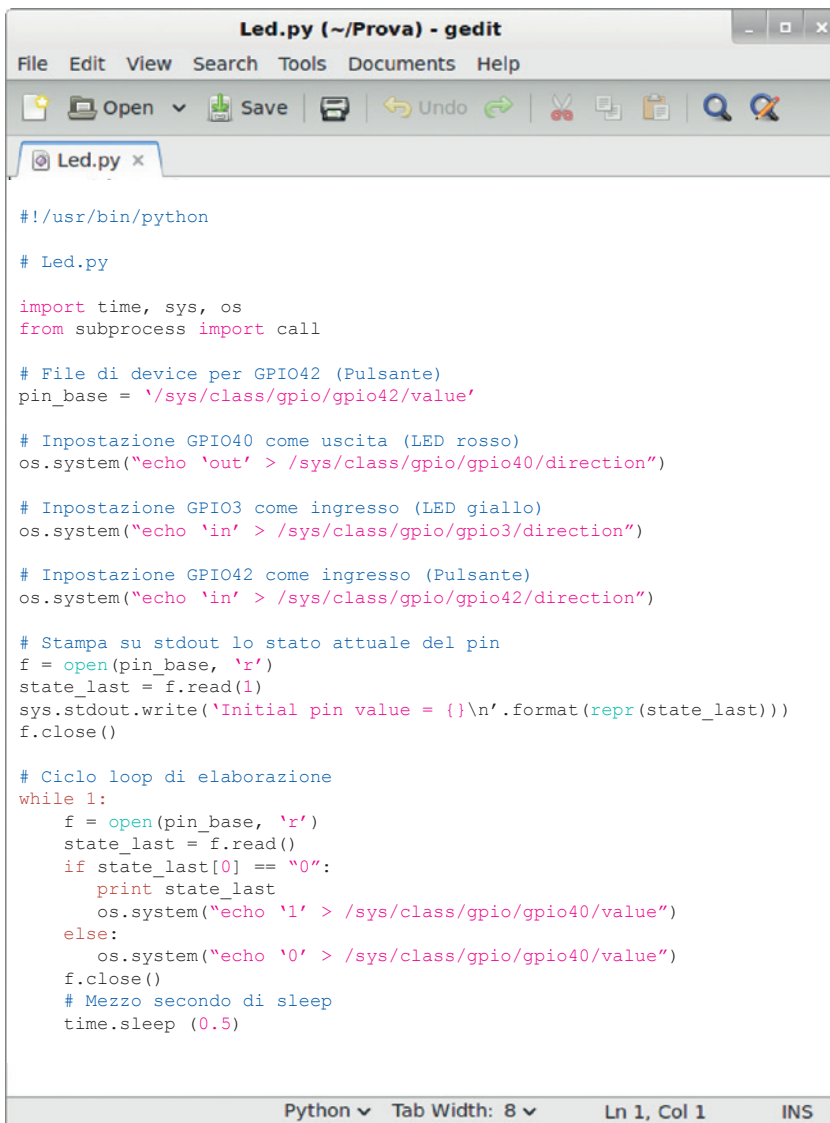
I programmi che vi presentiamo

hanno il duplice scopo di familiarizzare con l'utilizzo dell'IDE di sviluppo per Arduino incluso nella distribuzione GNU/Linux e con la condivisione dei pin tra i due ambienti. Per fare questo



Listato 1

dobbiamo prima conoscere la corrispondenza tra i pin di Arduino ed i pin del processore Freescale. Ci aiutiamo ancora con lo schema di Fig. 2, dal quale possiamo verificare che il pin 7 di Arduino al quale abbiamo collegato il pulsante P1 corrisponde al pin "gpio42". Vogliamo fare in modo che la pressione del pulsante P1 faccia accendere contemporaneamente il LED giallo, controllato dallo sketch di Arduino ed il LED rosso controllato dal programma Python in esecuzione su GNU/Linux. Per precauzione è bene impostare il LED gestito in output da Arduino come input per GNU/Linux e viceversa, in modo da evitare conflitti ... elettrici. Per questo motivo abbiamo identificato le corrispondenze anche dei pin di controllo dei LED. Il LED giallo, controllato in output dal pin 12 di Arduino, corrisponde al "gpio3" in GNU/Linux. Il pin "gpio40", che controlla in output il LED rosso da programma Python, corrisponde al pin 13 di Arduino. Nel Listato 1 presentiamo lo sketch di Arduino che permette di accendere il LED giallo quando viene premuto il pulsante P1. Apriamo l'IDE cliccando sull'icona sul desktop di GNU/Linux. Scriviamo il programma all'interno dell'IDE, oppure possiamo scaricarlo dal sito www.elettronica.in.it. Controlliamo nel menu "Tools">>"Board" che sia selezionata l'opzione "Arduino Due (Programming Port)" e nel menu "Tools">>"Port" che sia spuntata la porta di comunicazione "/dev/ttyMX3" (Fig. 12). Salviamo lo sketch con il nome "Led.ino" e compiliamolo cliccando sull'icona a forma di segno di spunta ("Verify"). Se non vi sono errori di compilazione carichiamo lo sketch su Arduino con il pulsante "Upload" (quello col segno di "freccia a destra"). Possiamo provare immediatamente il fun-



```
#!/usr/bin/python

# Led.py

import time, sys, os
from subprocess import call

# File di device per GPIO42 (Pulsante)
pin_base = '/sys/class/gpio/gpio42/value'

# Inpostazione GPIO40 come uscita (LED rosso)
os.system("echo 'out' > /sys/class/gpio/gpio40/direction")

# Inpostazione GPIO3 come ingresso (LED giallo)
os.system("echo 'in' > /sys/class/gpio/gpio3/direction")

# Inpostazione GPIO42 come ingresso (Pulsante)
os.system("echo 'in' > /sys/class/gpio/gpio42/direction")

# Stampa su stdout lo stato attuale del pin
f = open(pin_base, 'r')
state_last = f.read(1)
sys.stdout.write('Initial pin value = {}\\n'.format(repr(state_last)))
f.close()

# Ciclo loop di elaborazione
while 1:
    f = open(pin_base, 'r')
    state_last = f.read()
    if state_last[0] == "0":
        print state_last
        os.system("echo '1' > /sys/class/gpio/gpio40/value")
    else:
        os.system("echo '0' > /sys/class/gpio/gpio40/value")
    f.close()
    # Mezzo secondo di sleep
    time.sleep(0.5)
```

zionamento premendo il pulsante. Dobbiamo vedere accendersi il LED giallo. Ora pensiamo al programma Python per GNU/Linux, visibile nel **Listato 2**.

Creiamo una cartella "Prova" in

/home/ubuntu (**Fig. 13**). Entriamo nella cartella e creiamo il file "Led.py", cliccando col tasto destro del mouse e selezioniamo "New">>"File". Copiamo il file dal listato o scarichiamolo dal

Listato 2

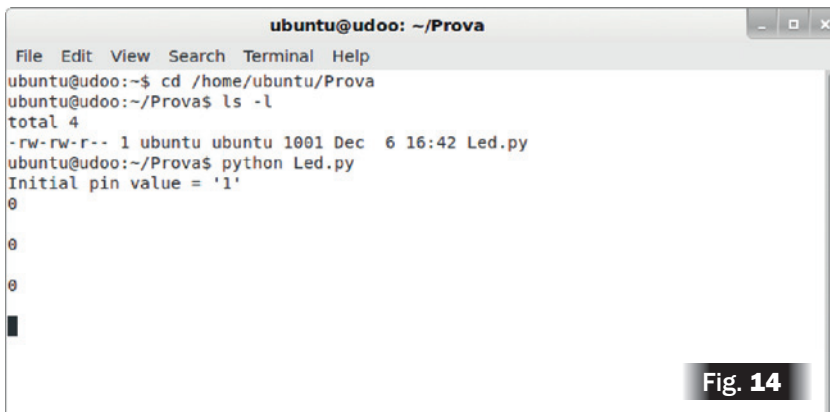
sito citato in precedenza, salviamo e chiudiamo. Per eseguire il programma, nella finestra di terminale, entriamo nella cartella /home/ubuntu/Prova e digitiamo il comando:

python Led.py

Proviamo a premere il pulsante sulla basetta di prova e vedremo accendersi entrambi i LED.

CONCLUSIONI

Non abbiamo avuto tempo di approfondire le ulteriori numerose funzionalità della scheda UDOO. In particolare l'accesso alla porta seriale dall'esterno della scheda, sia per comunicare con GNU/Linux che per programmare il microcontrollore Arduino dall'esterno. Allo stesso modo è possibile pilotare l'accesso alle porte micro USB. Tutto questo configurando opportunamente i jumper presenti sulla scheda. Ci ripromettiamo inoltre di verificare la connettività WiFi e la possibilità, per la versione Quad, di collegare un disco fisso SATA da utilizzare anche per il boot del sistema operativo. A questo punto pensiamo che ci troveremo di fronte ad un vero e proprio PC, in grado a nostro parere di sostituire i PC tradizionali utilizzati in diversi ambiti, anche professionali come supervisione di sistemi di automazione e controllo, integrazione dei sistemi di produzione con i sistemi gestionali, raccolta ed elaborazione dati, server web, database e file transfer, sistemi di visione e di sicurezza ed infiniti altri. Per quanto abbiamo visto finora, dobbiamo fare i complimenti ai progettisti di questa scheda che consigliamo vivamente a quanti hanno bisogno di realizzare applicazioni di tipo "professionale" con prestazione al di sopra della media.



```
ubuntu@udoo: ~/Prova
File Edit View Search Terminal Help
ubuntu@udoo:~$ cd /home/ubuntu/Prova
ubuntu@udoo:~/Prova$ ls -l
total 4
-rw-rw-r-- 1 ubuntu ubuntu 1001 Dec  6 16:42 Led.py
ubuntu@udoo:~/Prova$ python Led.py
Initial pin value = '1'
0
0
0
```

Fig. 14

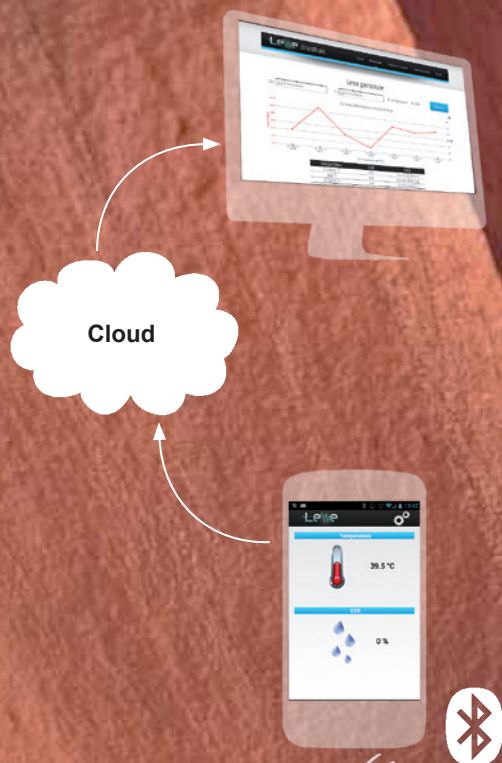
LEWE BRACCIALE BIOMETRICO

Portato al polso, rileva temperatura e sudorazione e ne trasmette le informazioni via bluetooth ad uno smartphone, collegato ad Internet e predisposto per la pubblicazione su un apposito servizio cloud.

dell'Ing. MIRCO SEGATELLO
e ALESSANDRO PASQUALINI

Lo sviluppo dei mezzi di comunicazione ha subito negli ultimi anni un impulso incredibile; in particolare, la telematica ha raggiunto livelli sempre più sofisticati, dando vita ad una vera e propria rivoluzione nella

diffusione delle informazioni, abbattendo barriere economiche, geografiche e culturali. L'utilizzo delle nuove tecnologie garantisce oggi l'opportunità di far circolare idee e progetti, dando vita ad un flusso comunicativo dove argomenti e contenuti digitali sono condivisi da milioni di utenti che, sparsi in ogni parte



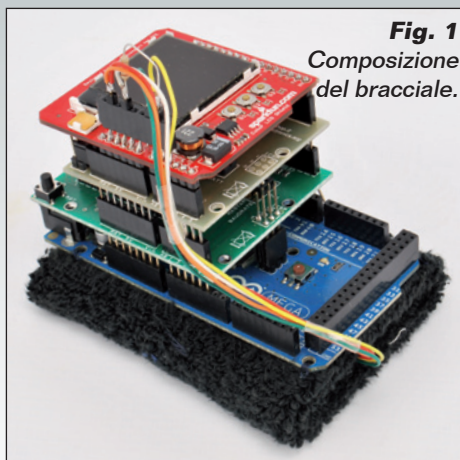


Fig. 1
*Composizione
del bracciale.*

del mondo, sono al contempo fruitori, creatori e coautori di un lavoro globale. Il progetto Lewe rientra in questo ambito e intende raccogliere e rielaborare le tecnolo-

ne di un antifurto per auto, la telegestione di una serra, oppure il braccialetto biometrico Lewe descritto in queste pagine. Il braccialetto biometrico nasce dall'idea degli autori: Alessandro, che ama smisuratamente l'arrampicata e il parkour, e Mirco, il suo personal trainer, che vorrebbe conoscere in tempo reale i dati biometrici di Alessandro semplicemente accedendo al cloud online di Lewe. Il tutto ha trovato espres-

sione in un bracciale per la lettura dei dati biometrici che saranno inviati via bluetooth ad uno smartphone Android e, tramite esso, ad un servizio cloud per l'archiviazione e l'organizzazione dei dati. In versione sperimentale, questo progetto è stato realizzato (vedere Fig. 4) mediante una scheda Arduino Mega che ospita uno shield Bluetooth di Futura Elettronica, uno shield RTC (sempre di Futura Elettronica) ed un Color LCD shield della Sparkfun; chiaramente il tutto ha un ingombro non trascurabile, ma la nostra intenzione è spiegarvi come realizzare l'applicazione, fermo restando che chi se la sentirà potrà realizzare un proprio hardware con architettura Arduino Mega (gli schemi delle board Arduino sono di pubblico dominio...) e integrare tutto in una sola scheda o al massimo due, molto compatte e applicabili (tipo misuratore di pressione) al polso mediante un apposito contenitore dotato di bracciale. Nel progetto che abbiamo messo a pun-

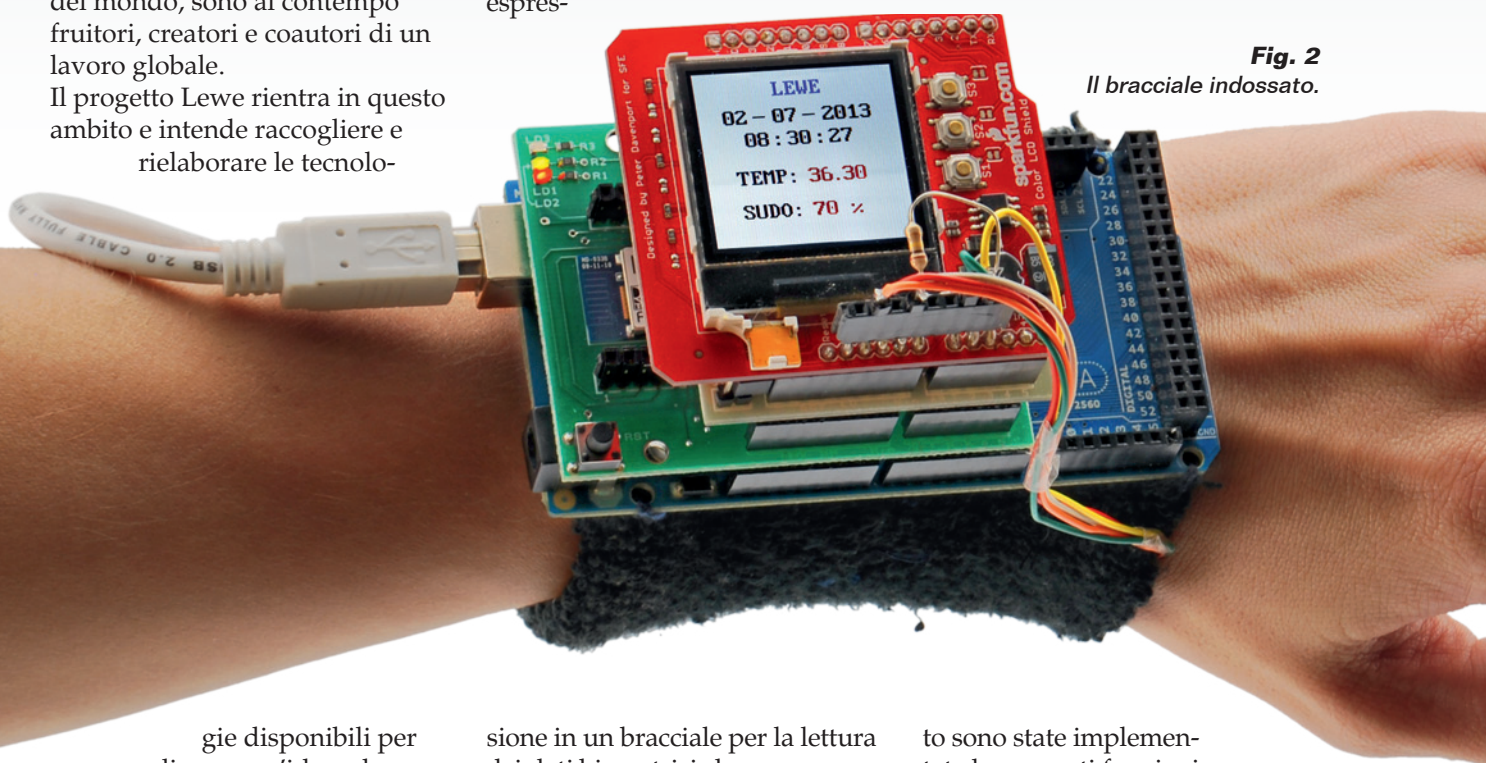


Fig. 2
Il bracciale indossato.

gie disponibili per realizzare un'idea a basso costo applicabile a tutti i settori che utilizzano sensori per la rilevazione di dati. Il progetto ha come scopo la realizzazione di un portale cloud, chiamato Lewe, del relativo protocollo di comunicazione, denominato Jack (capace di trasportare i dati in maniera affidabile) e di una serie di dispositivi di varia natura che sfruttano questa struttura per pubblicare automaticamente i dati sul web. Esempi dell'applicazione di questo protocollo potrebbero essere la realizzazio-

sione in un bracciale per la lettura dei dati biometrici che saranno inviati via bluetooth ad uno smartphone Android e, tramite esso, ad un servizio cloud per l'archiviazione e l'organizzazione dei dati. In versione sperimentale, questo progetto è stato realizzato (vedere Fig. 4) mediante una scheda Arduino Mega che ospita uno shield Bluetooth di Futura Elettronica, uno shield RTC (sempre di Futura Elettronica) ed un Color LCD shield della Sparkfun; chiaramente il tutto ha un ingombro non trascurabile, ma la nostra intenzione

to sono state implementate le seguenti funzioni:

- rilevazione della temperatura e della sudorazione del corpo;
- visualizzazione in locale dei parametri acquisiti;
- invio dei valori ad un'applicazione installata su uno smartphone;
- caricamento gratuito dei dati in un cloud online;
- organizzazione dei dati in semplici grafici che consentono di conoscere l'andamento nel tempo dei parametri vitali e di confrontarli con quelli di altri utilizzatori.

HARDWARE

Il bracciale è stato sviluppato attorno alla piattaforma Arduino Mega (Fig. 3). Per la comunicazione con lo smartphone è stato utilizzato un modulo Bluetooth RN-42 interfacciato con Arduino tramite lo shield Bluetooth Futura Elettronica, mediante la seriale hardware (pin 0 e 1). In questo progetto abbiamo preferito l'utilizzo di una seriale software sui pin 6 e 10, per permettere alla seriale hardware di comunicare liberamente con il PC anche durante la comunicazione Bluetooth, funzione,



questa, utile per il debug del software. Allo scopo, bisogna interrompere le piste uscenti dai pin 0 e 1 e connetterle ai pin rispettivamente 10 e 6 dello shield; per comodità, si possono usare i pin del connettore denominato RS232 già presente sullo shield. Per evitare che i pin 0 e 1 dello shield Bluetooth entrino in comunicazione con i contatti sottostanti della board Arduino, potete tagliarli oppure (Fig. 5) piegarli all'esterno, così da non

farli introdurre nei connettori della scheda.

Per la lettura della temperatura abbiamo utilizzato un comunissimo sensore LM35 in contenitore TO-92, la cui uscita è connessa all'ingresso analogico A0 di Arduino. Invece, per rilevare la sudorazione sono stati usati due elettrodi connessi ad un partitore di tensione facente capo all'ingresso analogico A1 di Arduino. Il principio di funzionamento del sistema di rilevamento della sudorazione è basato sulla conducibilità del sudore, il quale, essendo una soluzione salina, lascia fluire la corrente da un elettrodo all'altro; più alto è il livello della sudorazione, minore è la resistenza rilevata tra gli elettrodi. Questi due elettrodi devono avere la dimensione di una moneta da un centesimo e ovviamente avere buone caratteristiche di conducibilità e di resistenza all'ossidazione. Con il partitore di tensione

otteniamo un valore di tensione dipendente dal livello di sudorazione, che a sua volta è in relazione agli stimoli emotivi di chi lo indossa ed è un buon indicatore dello sforzo fisico dell'atleta. Ovviamente le tensioni e le correnti in gioco sono così piccole da non creare alcun pericolo per la persona che indossa il bracciale. Nello schema elettrico noterete che i sensori sono alimentati tramite il pin 5V di Arduino, ma dato che l'intero bracciale sarà alimentato a batteria, abbiamo previsto la possibilità di controllare l'alimentazione dei sensori, attivandoli solo quando serve fare la misura. L'alimentazione del sensore LM35 può essere prelevata dal pin 12, mentre il sensore di sudorazione può essere alimentato dal pin 7. Questi pin sono mantenuti sempre a livello basso ed attivati via software solo quando è necessario eseguire la misura (ciò consente di limitare il consumo di elettricità, in previsione dell'alimentazione a batteria); la corrente di uscita di 40mA indicata nei data-sheet

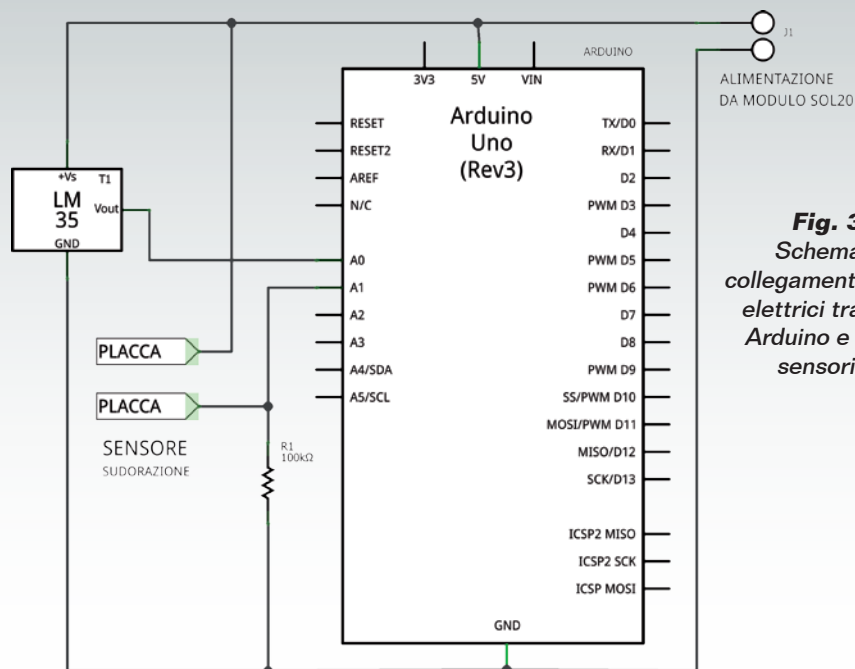


Fig. 3
Schema
collegamenti
elettrici tra
Arduino e i
sensori.

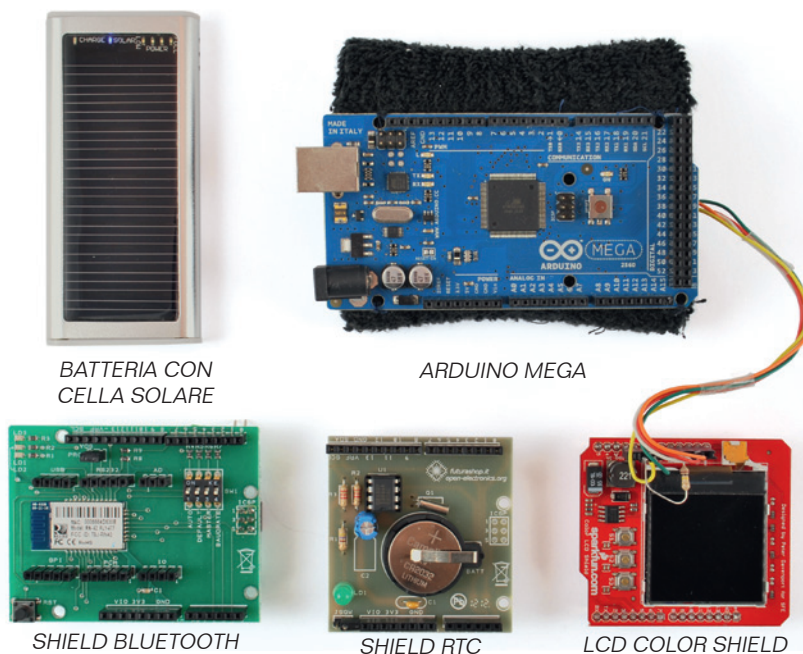


Fig. 4 - Componenti del sistema Lewe.

del microcontrollore Atmel, è sufficiente per alimentare i pin interessati dalla misura. Per correlare temporalmente i dati rilevati, abbiamo utilizzato uno shield RTC messo in comunicazione con Arduino tramite il bus I²C, mediante i pin analogici A4 e A5. Abbiamo altresì inserito uno shield LCD Color per visualizza-

re in tempo reale i valori biometrici e lo stato di funzionamento, oltre a data e ora; le misurazioni acquisite ad intervalli regolari, di durata impostabile, sono inviate in tempo reale ad uno smartphone per l'archiviazione. I pin 8,9,11,13 (trasferimento dati) assieme ai pin 3,4,5 (pulsanti) sono usati dall'LCD Color shield.

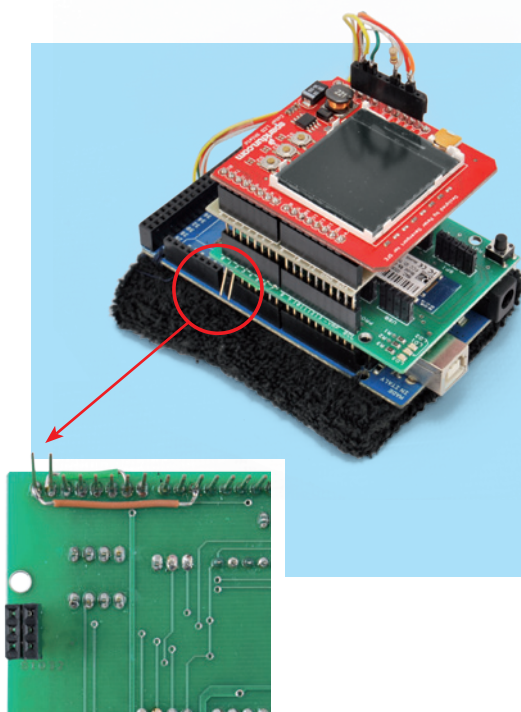


Fig. 5 - Modifica dello shield bluetooth per consentire l'utilizzo della seriale software sui pin 6 e 10. In basso, posizionamento dei contatti all'interno del bracciale.



Il bracciale, in assenza di collegamento Bluetooth con lo smartphone, può memorizzare fino a 12 misurazioni; non appena si collega, invia le 12 misure.

Tutto il sistema è alimentato da un modulo batteria, distribuito da Futura Elettronica (codice SOL20), che permette di essere ricaricato sia tramite una porta USB, sia con il piccolo pannello solare incorporato, ottimo per l'utilizzo all'aperto. All'occorrenza lo stesso modulo può intervenire per la ricarica dello smartphone.

Il modulo SOL20 fornisce in uscita una tensione di 5V stabilizzata, che possono essere utilizzati per alimentare la scheda Arduino utilizzando i pin +5V e GND; per fare questo è però necessario tagliare un cavo USB in modo che da un lato rimanga la spina USB da connettere al modulo batteria, mentre dall'altra parte si utilizzano i fili spellati per portare l'alimentazione ai pin +5V e GND di Arduino. Per evitare di manomettere il cavo è possibile usare lo stesso cavo impiegato per la programmazione di Arduino, solo che invece di inserirlo nella porta USB del PC sarà inserito nel modulo SOL20: alquanto ingombrante ma estremamente funzionale. Lo stesso sistema potrà essere usato per alimentare con batteria qualsiasi altro vostro progetto, con il vantaggio di disporre della doppia modalità di ricarica: via USB o con pannello solare integrato.

FIRMWARE

La realizzazione del firmware è stata eseguita con l'IDE di Arduino utilizzando diverse librerie standard ed alcune librerie esterne. La libreria color LCD è stata utilizzata per gestire il Color LCD shield della Sparkfun, la libreria HashMap è stata utilizza-

Fig. 6
LCD
con visualizzazione
orologio.

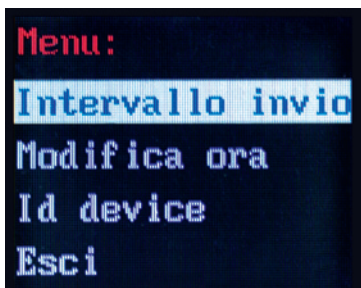
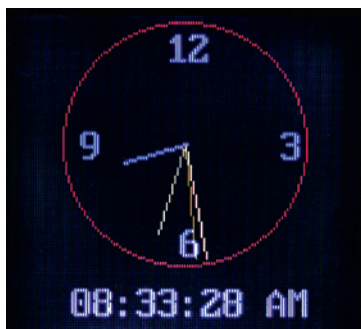


Fig. 8 - LCD
con visualizzazione menu.

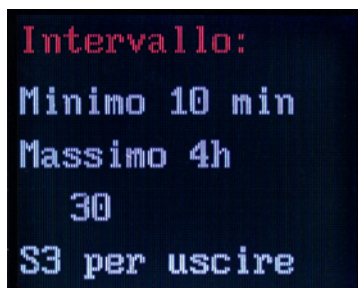
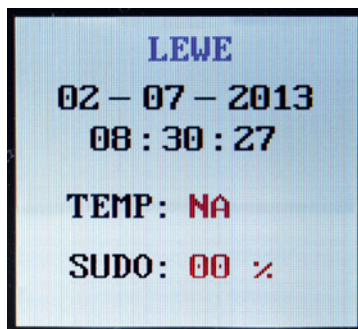


Fig. 9 - LCD
con visualizzazione impostazioni.

ta per gestire array di stringhe ed infine la libreria Wrapper è stata usata per facilitare l'estrazione di valori stringa incapsulati in array. Per l'utilizzo del modulo Bluetooth è stata creata la libreria SoftwareSerialJack, basata sulla SoftwareSerial, presente nel SDK di Arduino, per implementare una comunicazione più sicura grazie all'aggiunta del protocollo di comunicazione Jack che garantisce una connessione affidabile tra mittente e destinatario, implementando un sistema di conferme di ricezione dei messaggi. Tale protocollo è stato sviluppato per potersi appoggiare a qualunque mezzo trasmissivo e prevede l'implementazione delle funzioni **boolean available(void)**, **String getMessage(void)** e **void sendMessage(String)**.

La boolean **available (void)** restituisce un valore booleano che significa vero nel caso in cui ci siano messaggi disponibili da poter scaricare dal mezzo di trasmissione, altrimenti vale falso. La **String getMessage(void)** restituisce in formato stringa, uno alla volta, i messaggi ricevuti dal mezzo trasmissivo. Infine, la

sendMessage(String) si preoccupa di inviare nel mezzo trasmissivo i messaggi, in formato stringa, provenienti dall'istanza del protocollo Jack. Quest'ultimo mette a disposizione una classe che funge da contenitore per i dati chiamato **JData** e sfrutta le classi wrapper per immagazzinare i dati e con-

Fig. 7 - LCD
con visualizzazione
dati.

servarne il tipo. Il metodo **send(JData)** del protocollo si occupa di creare un messaggio di tipo stringa mettendo assieme il messaggio effettivo da inviare con un identificatore univoco del messaggio, il quale è composto dai valori di data e ora di acquisizione, dal valore della temperatura e della sudorazione. Il messaggio viene quindi inviato al ricevitore e da questo elaborato. Alla ricezione, viene invocata la funzione **available()** la quale passerà il messaggio alla funzione **getMessage()** che provvederà a salvare il messaggio ricevuto. Per processare il messaggio ricevuto viene usato il metodo **execute(String)**, il quale provvederà a verificare se il messaggio è conforme allo standard, quindi provvederà ad estrarre il codice univoco ed i dati contenuti nel messaggio; per confermare la ricezione del messaggio, viene spedito al mittente l'identificatore univoco. Se il messaggio non dovesse arrivare o sarà corrotto, non ci sarà la ricevuta di ritorno

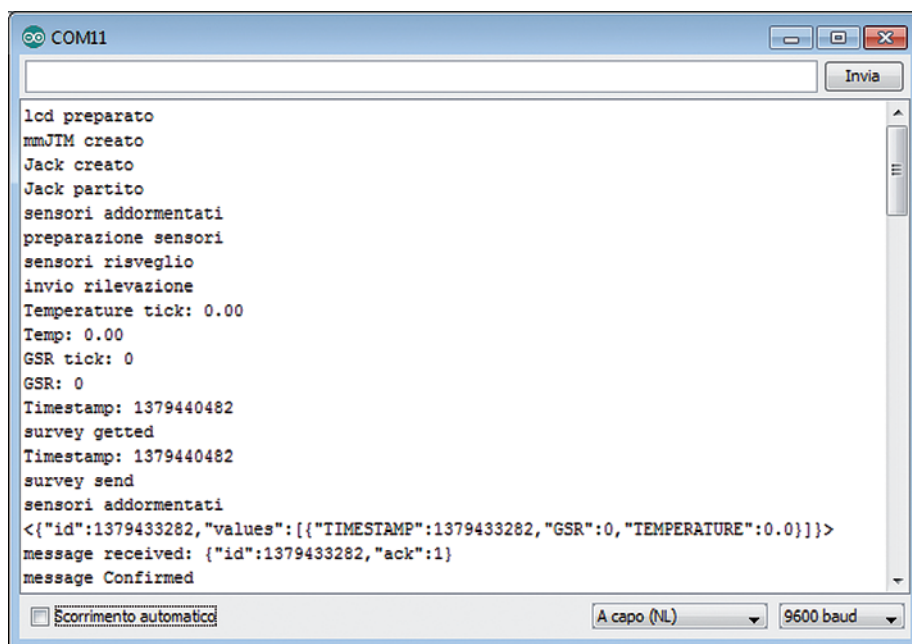


Fig. 10 - Traffico dati ricevuti da serial monitor.

ed il trasmettitore capirà che il messaggio è andato perso e lo dovrà ritrasmettere. In questo modo, anche una caduta momentanea del collegamento Bluetooth non comporterà la perdita di informazioni, le quali saranno rispedite non appena il collegamento sarà ripristinato. Sarà poi il codice orario all'interno del messaggio a permettere la ricostruzione della corretta sequenza temporale.

Per compilare correttamente lo sketch per Arduino è necessario installare le librerie aggiuntive ColorLCDShield, HashMap, Jack, RTCLib, SoftwareSerialJack e Wrapper, tutte fornite assieme ai file di questo progetto; il considerevole numero di righe fa lievitare parecchio lo spazio occupato dallo sketch il quale può girare solo su una scheda Arduino Mega 2560 che dispone di ben 256 kB di spazio per il programma; Arduino UNO non dispone di sufficiente spazio per lo sketch. Durante il funzionamento, il bracciale controlla costantemente

via USB le stringhe inviate e ricevute via Bluetooth, in modo da poter verificare la correttezza del protocollo utilizzato; il campo "id" rappresenta l'identificativo del messaggio, valore chiave utilizzato per verificare la bontà dei messaggi ricevuti. Il modulo Bluetooth utilizza tutte le impostazioni di default ed i jumper dello shield sono così posizionati: AUTO=off, DEFAULT=off, MASTER=off e BAUDRATE=on. Il modulo è quindi forzato a funzionare come slave con velocità di comunicazione di 9600bps. Se il modulo è stato precedentemente programmato, è necessario venga resettato e portato alle impostazioni di fabbrica, funzione svolta dal pulsante RESET presente sullo shield. È però importante assegnargli un nome che sia "lewe" compatibile ovvero del tipo LEWE_idDevice, dove idDevice è un numero identificativo del dispositivo. Per questa procedura abbiamo scritto un apposito sketch con il quale è possibile assegnare un nome,

diverso da quello di default, al modulo Bluetooth.

Lo sketch si chiama *RN_42_serial-monitor.ino* e permette di interfacciarsi al modulo RN-42 tramite il serial monitor di Arduino. Dopo aver caricato lo sketch su Arduino è sufficiente inviare i seguenti comandi AT:

- impostare "nessun fine riga" su serial monitor e inviare \$\$\$ per entrare in command mode (risposta: CMD);
- impostare "sia NL sia CR" su serial monitor;
- inviare D per avere come risposta la configurazione (risposta: AOK);
- inviare SN, LEWE_01 per impostare il nome al modulo (risposta: AOK);
- impostare "sia NL sia CR" su serial monitor;
- inviare --- per uscire dal command mode (risposta: END).

Lasciando in esecuzione questo sketch è possibile interfacciarsi con un dispositivo Android e provare uno scambio dati semplicemente installando una APP con funzione di Bluetooth SPP in grado cioè di gestire comunicazioni seriali via BT. Abbiamo fatto delle prove con l'applicazione "Bluetooth spp pro" gratuita, con la quale è possibile inviare e ricevere stringhe. Questa APP consente anche la modalità "Button" con la quale si possono programmare dei pulsanti per inviare stringhe predefinite, utile per gestire le uscite di Arduino o comandi più avanzati.

APPLICAZIONE PER SMARTPHONE

L'applicazione per Smartphone Android costituisce il vero cuore di Lewe, perché si occupa di ricevere i dati dal bracciale e di memorizzarli localmente nel dispositivo e, se correttamente configurata, di caricarli nel cloud

Fig. 11
Il grafico della temperatura sullo smartphone.

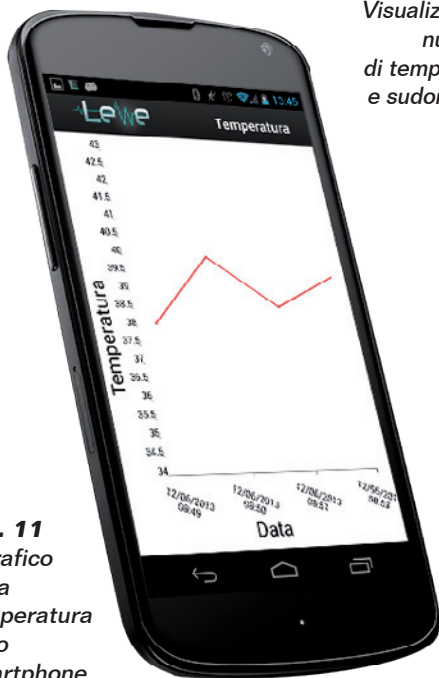


Fig. 12
Visualizzazione numerica di temperatura e sudorazione.



online. È composta da un'attività principale nella quale sono mostrati gli ultimi dati ricevuti dal bracciale ed un'attività secondaria che visualizza il grafico contenente tutti i dati memorizzati. Cliccando, poi, sull'icona dell'ingranaggio, è possibile entrare nelle impostazioni dell'app con le quali selezionare il bracciale al quale connettersi e configurare i dati di accesso al cloud.

L'applicazione è stata realizzata in Java con l'ausilio di Eclipse provvisto di ADT Plugin, che permette lo sviluppo di applicazioni Android; per i grafici si è scelto di utilizzare la libreria AChartEngine, disponibile sotto licenza Apache 2.0.

L'applicazione si avvale di quattro servizi che lavorano in background:

- LeweService;
- LewebluetoothService;
- LeweDatabaseService;
- LeweWebCloudService.

LeweService è il servizio principale dell'applicazione che viene avviato al primo start e rimane in background fino alla sua chiusura, avvia i rimanenti servizi e si preoccupa di terminarli una volta ricevuto il comando di spegnimento. Il servizio comunica direttamente con gli altri servizi *LeweWebCloudService* e *LewebluetoothService*; oltre a ciò, invia i comandi di connessione e disconnessione ai suddetti servizi, che comunicano con l'esterno.

LewebluetoothService sfrutta la classe *bluetoothChatService* per la ricezione dei dati la quale è stata modificata aggiungendo la possibilità di recuperare la connessione con il dispositivo Bluetooth nel caso in cui dovesse mancare, inoltre è stata resa compatibile per operare con il protocollo Jack.

LeweDatabaseService si occupa di gestire il database dei dati acquisiti SQLite.

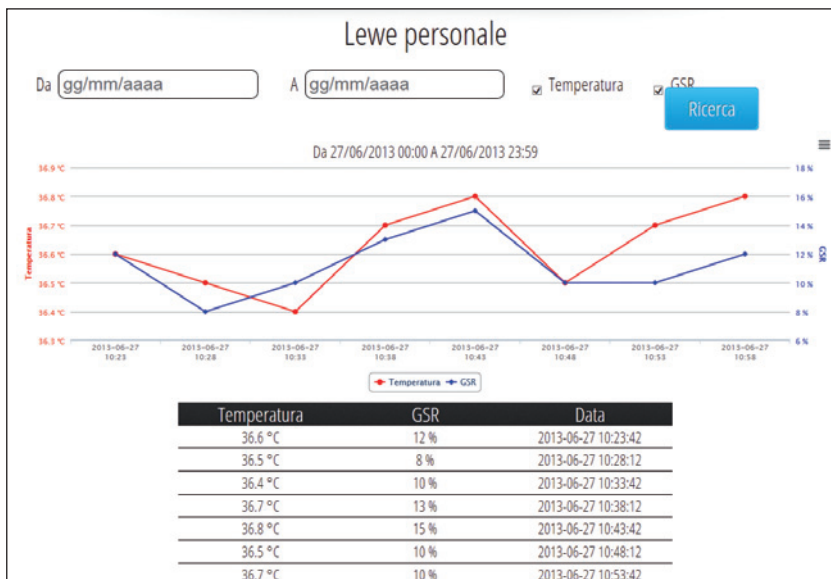


Fig. 13 - Schermata del web cloud online.

LeweWebCloudService gestisce la comunicazione tra applicazione e web cloud, in particolare si preoccupa di inviare i dati al cloud tramite richieste HTML post e di verificarne l'avvenuto caricamento tramite la conferma di ricezione dati, come previsto nel protocollo Jack.

Il web cloud online permette l'accesso ai propri dati da qualunque parte del mondo con utilizzo di credenziali per effettuare l'accesso al proprio spazio personale reso disponibile gratuitamente; se non si possiede un account, è stato previsto un servizio di registrazione. I dati vengono rappresentati sia in modalità grafica sia nella classica forma tabellare (Fig. 13); è anche presente un modulo per eseguire una ricerca per data. Se nessun parametro viene inserito, vengono automaticamente visualizzati i dati relativi all'ultimo giorno di utilizzo del bracciale.

Configurando l'applicazione con i dati di accesso allo spazio personale, è possibile iniziare a caricare i dati biometrici rilevati dal bracciale.

È possibile, inoltre, garantire l'accesso ai propri dati anche ad altre

persone compilando con il loro *username* il modulo presente nella pagina "Pubblico il mio Lewe". Accedendo alla pagina "Lewe di un amico" è possibile, infine, visualizzare i nomi degli utenti che ci hanno autorizzato e, cliccando su uno di essi, vedere i dati organizzati in maniera simile alla pagina "Il mio Lewe".

Per il caricamento dei dati è stato realizzato un sotto-cloud non accessibile direttamente, se non dopo autenticazione effettuata dall'applicazione Android per smartphone.

Tale sotto-cloud è stato realizzato sempre con l'ausilio del framework Azzurro e possiede un'implementazione del protocollo di comunicazione Jack per la ricezione e la conferma dei dati da memorizzare.

Per mettere in funzione il bracciale è sufficiente alimentarlo mentre l'applicazione **Lewe.apk** dovrà essere installata in un dispositivo Android.

Per prima cosa è necessario associare il bracciale allo smartphone accedendo alla schermata delle impostazioni, simbolo dell'ingranaggio in alto a destra (Fig. 14).



Fig. 14 - Applicazione Lewe.

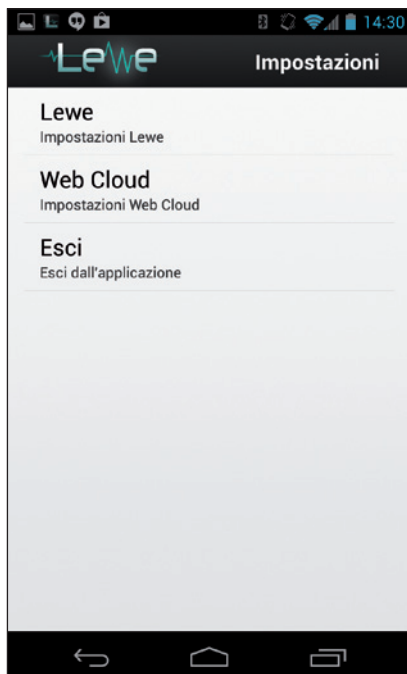


Fig. 15 - Configurazione dell'applicazione.

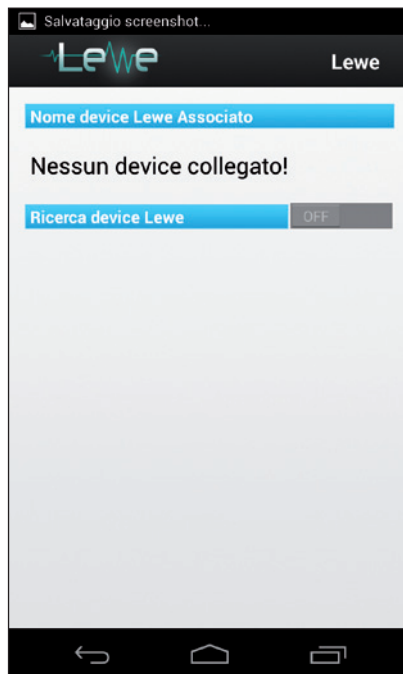


Fig. 16 - Configurazione dell'applicazione.

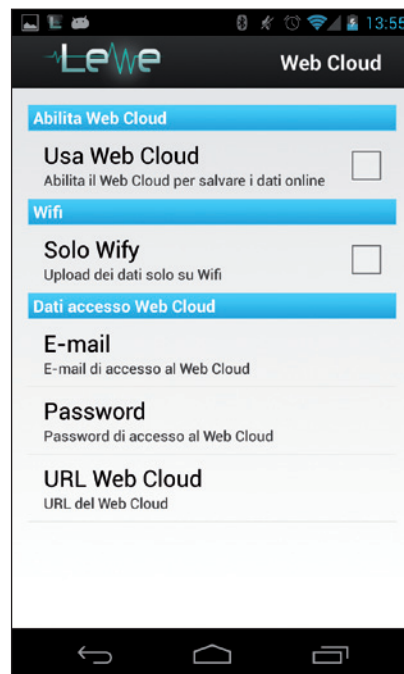


Fig. 17 - Configurazione Web Cloud.

Dal menu *Impostazioni* è possibile configurare la comunicazione con il bracciale ed il web cloud (Fig. 15). Cliccando sul pulsante Lewe e successivamente sul pulsante ON, si avvia la ricerca di dispositivi Lewe: nella casella sottostante appaiono i nomi dei bracciali che l'app rileva (Fig. 16); grazie ad un controllo sul nome Bluetooth del dispositivo vengono mostrati solo quelli che rispondono al pattern LEWE_idDevice, dove idDevice è un numero identificativo del dispositivo. Selezionando il bracciale al quale connettersi si stabilisce la connessione e l'applicazione è funzionante.

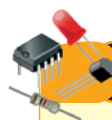
Questa operazione è necessario farla solamente al primo avvio dell'app o quando si cambia dispositivo, in quanto l'app memorizza il device Bluetooth al quale connettersi e tenterà di connettersi automaticamente ogni volta che si apre l'app. Per il cloud è necessario cliccare su "Web Cloud" e

inserire e-mail e password negli appositi campi (quelli di accesso al web cloud online) e spuntare "Usa Web Cloud" (Fig. 17). L'applicazione può funzionare anche senza cloud e quindi la sua impostazione è facoltativa. Il sistema è stato ampiamente testato con smartphone Samsung Galaxy Nexus, Galaxy Nexus 4 e Galaxy S2 con sistema operativo Android 4.0. È possibile visionare il cloud online accedendo al sito <http://www.lewe.tk>. Per l'accesso si utilizzi il form richiamabile cliccando sul pulsante "login", usando come credenziali: email *prova@prova.it* e password *prova123*.

CONCLUSIONI

Non sempre è facile sviluppare cose nuove, tanto meno quando possono interessare un gran numero di persone ed una vastità di applicazioni. Le nuove tecnologie, prime fra tutte il mobile, offrono ancora ampio margine per svi-

luppo e innovazione, a maggior ragione nei casi in cui il software incontra l'hardware e in questo Arduino ci ha insegnato molto. ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il caricabatterie solare da 1200 mAh cod. SOL20 costa 22,00 Euro ed è reperibile presso Futura Elettronica così come lo Shield Bluetooth cod. FT1032M (Euro 34,00), Arduino Mega2560 REV3 (Cod. ARDUINO-MEGAREV3, Euro 51,00) e l'RTC Shield cod. RTCSHIELDKIT (Euro 11,00). Tutti i prezzi si intendono IVA compresa.

Il materiale va richiesto a:
Futura Elettronica, Via Adige 11,
21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
<http://www.futurashop.it>

Proseguiamo il nostro viaggio alla scoperta di Flowcode, affrontando la gestione dell'USB e realizzando un convertitore seriale RS232-USB, oltre ad un tastierino numerico e ad un analizzatore di stati logici, sempre su USB. Sesta puntata.

corso di programmazione in **FLOWCODE**

di Francesco Ficili e Daniele Defilippi

Nelle scorse puntate abbiamo parlato del bus di comunicazione avanzato CAN ed imparato a gestirlo in modo semplice e intuitivo grazie a Flowcode. In questa puntata faremo altrettanto con il bus USB, spiegando come funziona la comunicazione, in che modo si possa convertire un progetto RS232 in una periferica USB in pochi semplici passi e realizzeremo due tipi differenti di periferiche, ovvero una tastiera numerica per il PC e un analizzatore di stati logici.

La porta USB

L'interfaccia maggiormente utilizzata oggi per la comunicazione tra

Personal Computer e periferiche è il bus USB. Esso ha sostituito praticamente tutte le altre precedenti interfacce sia seriali che parallele perché permette di trasferire dati ad una velocità molto elevata e collegare molti dispositivi (teoricamente fino a 127) ad un'unica porta, mediante uno o più hub eventualmente disposti in cascata. Altra caratteristica fondamentale dell'USB è la possibilità di collegare e rimuovere i dispositivi senza dover riavviare il computer, il che fa del bus un vero Plug&Play. I dispositivi collegati possono essere alimentati direttamente dal bus o, qualora richiedano correnti superiori a 500 mA (il limite erogabile da una porta USB; se si connetto-

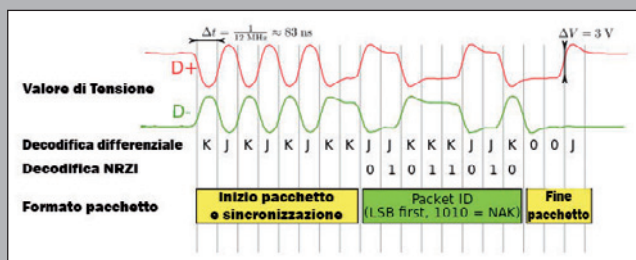


Fig. 1
Struttura
di un
pacchetto
trasmesso
sul bus
USB.

no più dispositivi, devono spartirsi questa corrente), devono avere una connessione di alimentazione a parte.

Lo standard USB 1.0 nacque nel 1996 e consentiva una velocità di connessione pari a 1,5 Mbit/s; le successive versioni offrono velocità di comunicazione maggiori: per esempio la 1.1 permette di raggiungere i 12 Mbit/s, la 2.0 i 480 Mbit/s e la più recente 3.0 permette una velocità massima di 4.8 Gbit/s. Mentre dalla prima versione alla 2.0 la connessione è rimasta invariata (quattro contatti) con la 3.0 è stato introdotto un nuovo connettore che ha i contatti sdoppiati e di fatto prevede un doppio canale dati; la connessione rimane comunque compatibile con le vecchie, in quanto una fila di contatti tocca quelli dello spinotto tradizionale. Esistono differenti connettori fisici che permettono la connessione al bus USB: A, B, mini e micro. Dal 2011 il connettore micro-USB è diventato la porta standard di comunicazione con tutti i sistemi cellulari, ma con l'avvento degli smartphone è sorta la necessità di sviluppare un protocollo che permetta di decidere all'atto della connessione quale dispositivo diventa il master e quale lo slave. La soluzione al problema si chiama USB-On-The-Go e permette di impostare, all'atto della connessione, quale dei due dispositivi è il master e quale lo slave. Questo accorgimento fa in modo, ad esempio, che uno smartphone possa essere visto come uno slave se connesso ad un PC o come master nel caso sia connesso ad una periferica di archiviazione di massa come un hard-disk. Vediamo ora quali connessioni sono disponibili su un connettore USB-mini/micro:

- **VBUS**; permette di alimentare le periferiche erogando 5V stabilizzati e una corrente massima di 500mA;
- **GND**; è la massa di riferimento per i segnali e l'alimentazione;
- **D-**; dato -;
- **D+**; dato +;

- **ID**: identifica se il dispositivo funziona da controllore (master) o periferica (slave); nel connettore di tipo A è collegato a GND per fare in modo che il dispositivo collegato alla porta mini/micro debba funzionare da periferica.

Lo standard 3.0 prevede il raddoppio delle connessioni dati in modo da poter avere due linee per la ricezione dei dati ad altrettante dedicate unicamente alla trasmissione. I dati sono trasmessi in modo pseudo-differenziale perché la linea D- ha come valore l'opposto della linea D+, ma ciò non è vero nel caso dell'identificatore del termine pacchetto.

Nell'USB, la codifica è di tipo NRZ ed il valore zero provoca l'inversione del valore logico trasmesso. Il segnale di clock è trasmesso in modo implicito durante l'invio dei dati perciò, per evitare che il bus stia fermo allo stesso valore per troppo tempo, dopo 6 bit consecutivi di valore 1 viene trasmesso un bit aggiuntivo di valore zero.

La comunicazione fisica è suddivisa in tre parti principali:

- **Pacchetto di start**: Viene inviata la sequenza di sincronizzazione 00000001. Nel caso di pacchetti più grandi la sequenza di sincronizzazione è di 32bit;
- **Pacchetto di dato**: La lunghezza dipende dal tipo di connessione, da 1 a 1024 byte, ed il primo bit trasmesso è quello meno significativo;
- **Pacchetto di Stop**: Viene trasmessa una sequenza non differenziale in cui entrambi i segnali restano al valore logico basso per due cicli e poi il segnale D+ torna al valore logico alto.

In Fig. 1 è mostrato un esempio di trasmissione dati sul bus USB nel quale possiamo identificare le tre parti principali nel quale è suddiviso, la codifica NRZ ed il frame di stop.

Lo standard USB prevede la classificazione delle periferiche in "classi" in modo da facilitare la gestione ed identificazione dei necessari driver; in questo modo i dispositivi standard come mouse, tastiere e memorie, non richiedono l'installazione di driver e sono quindi subito pronti all'uso.

Passiamo ora a scoprire come dotare un nostro progetto di una connessione USB in modo molto semplice e rapido senza neppure dover rigenerare il codice del microcontrollore.

Il chip FT232BL

Il metodo probabilmente più semplice per gestire periferiche è usare la porta seriale: ciò, sia per la semplicità del firmware da caricare nel micro che governa la periferica, sia per il software di gestione lato personal computer. Il problema è che attualmente pochi PC hanno ancora il connettore DB9 della COM e i notebook non l'hanno più da tempo. Per questa ragione possiamo realizzare la periferica gestendo unicamente la connessione RS232 ed utilizzando un componente che si occupi della conversione di protocollo da RS232 ad USB. In questo

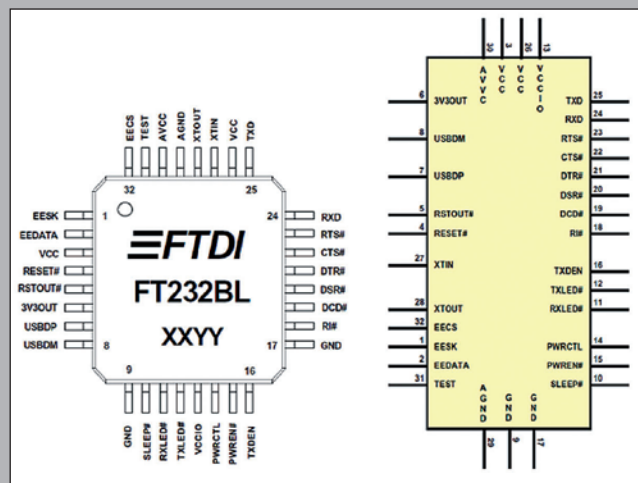


Fig. 2
Piedinatura
e schema di
riferimento per
il componente
FT232BL
di FTDI.

modo vedremo dal punto di vista del PC una seriale che potremo gestire normalmente con il microcontrollore che riceverà i dati attraverso la seriale senza doversi occupare della complessa gestione del bus USB. Questa metodologia di lavoro è molto funzionale ed utilizzata soprattutto per la rigenerazione di vecchi progetti perché non richiede la modifica del codice sorgente.

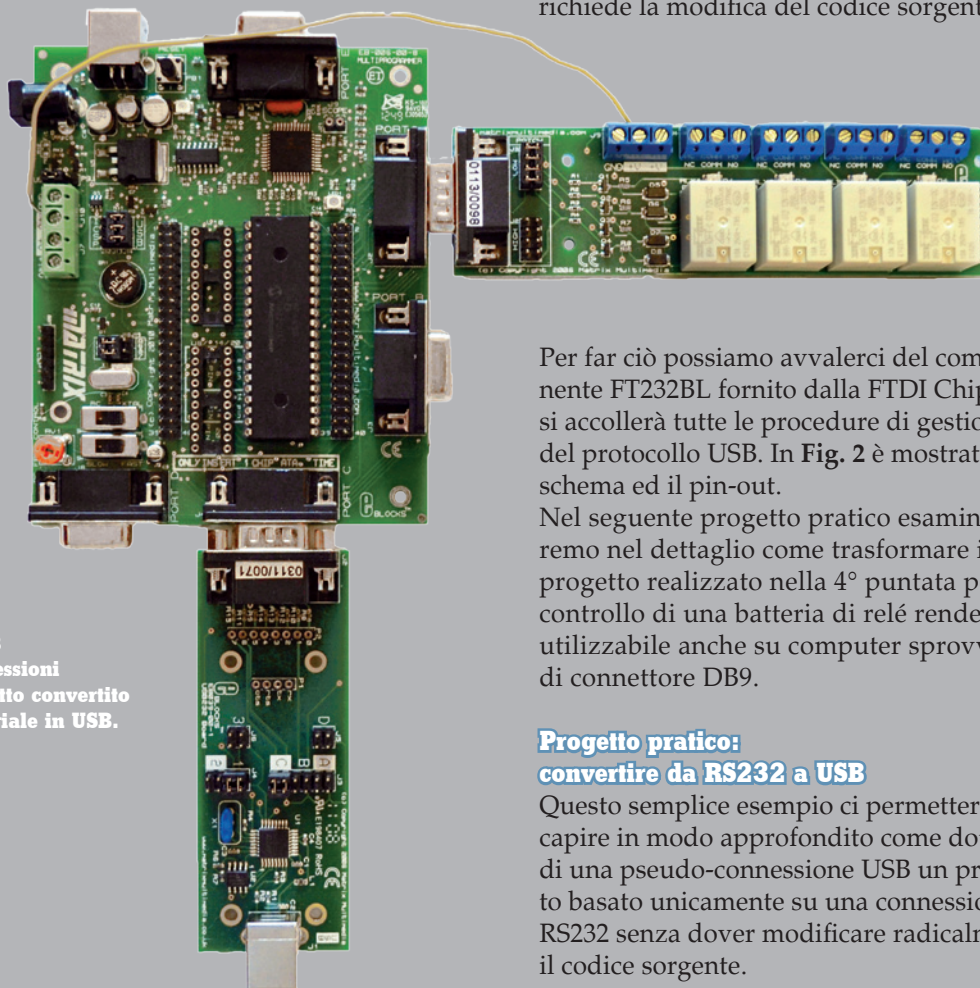


Fig. 3
Connessioni
progetto convertito
da seriale in USB.

Per far ciò possiamo avvalerci del componente FT232BL fornito dalla FTDI Chip, che si accollerà tutte le procedure di gestione del protocollo USB. In Fig. 2 è mostrato lo schema ed il pin-out.

Nel seguente progetto pratico esamineremo nel dettaglio come trasformare il progetto realizzato nella 4ª puntata per il controllo di una batteria di relé rendendolo utilizzabile anche su computer sprovvisti di connettore DB9.

Progetto pratico: convertire da RS232 a USB

Questo semplice esempio ci permetterà di capire in modo approfondito come dotare di una pseudo-connessione USB un progetto basato unicamente su una connessione RS232 senza dover modificare radicalmente il codice sorgente.

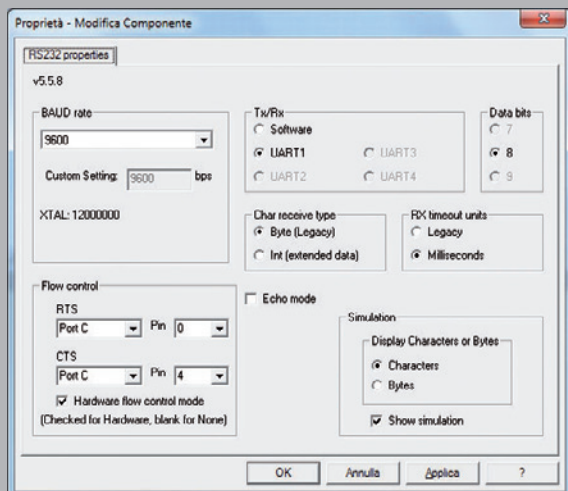


Fig. 4 - Nuova configurazione componente RS232.

I componenti necessari per la realizzazione sono i seguenti:

- EB006 "USB PICmicro multiprogrammer board" (PIC18F4580 e quarzo 12MHz);
- EB038 "Relay Board";
- EB039 "USB232 Board".

La scheda EB039 sostituisce la EB015 sulla porta C (Fig. 3) ed al posto della connessione di uscita su DB9 è presente la porta USB di tipo B.

Le modifiche, a parte quelle all'hardware, sono davvero poche. Vediamo dunque come realizzare il nostro progetto passo passo: apriamo il progetto "RS232.fcf", che avevamo salvato al termine della 4^a puntata, e modifichiamo le impostazioni del componente RS232 come mostrato in Fig. 4. La connessione RS232 deve essere impostata a 9.600 baud, 8bit e il controllo di flusso deve essere gestito mediante il pin RTS sulla porta C0 e il pin CTS sulla porta C4. Ricompiliamo il progetto e ricarichiamo il tutto sulla scheda di sviluppo, per vederne i risultati. Collegando il connettore USB tipo B, mediante un cavo, al connettore USB tipo A del nostro PC, il sistema operativo rileverà un nuovo hardware, corrispondente al componente FTDI, del quale dovremo installare il driver come indicato nel box "installiamo il driver del componente FTDI". Per questa variante del progetto è stata aggiornata l'interfaccia software presentata a corredo della 4^a puntata, permettendone l'utilizzo anche con una porta COM virtuale.

Il funzionamento rimane sostanzialmente invariato, con la differenza che ora la selezione della porta di comunicazione viene fatta tramite una combo box che rileva

automaticamente le seriali virtuali eventualmente connesse al PC.

Il componente USB in Flowcode

Flowcode permette di utilizzare principalmente tre tipi di configurazione per il bus USB:

- **Serial**; permette di gestire la porta USB come se fosse una normale connessione seriale;
- **HID**; permette di creare un dispositivo di tipo "Human interface device" come ad esempio un mouse, una tastiera, un joystick, ecc.;
- **Slave**; permette di creare un dispositivo custom.

La scelta di uno tra i sopraelencati tipi dipende dalla funzionalità che la nostra periferica implementa. La più semplice e intuitiva da utilizzare è quella di tipo seriale perché è vista dal calcolatore come una semplice porta COM, come per il caso del chip della FTDI che abbiamo precedentemente utilizzato, ed i comandi che abbiamo a disposizione dal punto di vista del microcontrollore sono quelli standard di una comunicazione seriale limitandosi all'invio e alla ricezione di caratteri e stringhe. Se volessimo realizzare un mouse o comunque un dispositivo standard di input dovremmo scegliere il tipo HID, in modo da poterlo utilizzare senza dover installare alcun driver aggiuntivo mentre negli altri casi dovremmo avvalerci del tipo slave.

Il componente USB-HID

È contenuto all'interno della barra dei componenti nel sotto-menu "periferiche" e permette di gestire un dispositivo di tipo "Human interface device". Dopo aver inserito il componente all'interno del pannello possiamo esaminare le proprietà del componente. Essendo abbastanza complesso, a differenza degli altri componenti più semplici, necessita di tre pagine di configurazione. Iniziamo ad analizzare la prima, illustrata nella Fig. 5.

Il parametro VID è il cosiddetto "Vendor Identifier" ed è, in pratica, l'identificativo univoco del venditore della periferica; viene assegnato a fronte del pagamento

Tabella 1

Nome macro	Descrizione
<code>RetVal = Initialise_HID()</code>	È utilizzata per inizializzare il driver USB per la comunicazione ed attivare il servizio di enumerazione delle periferiche. Restituisce zero se la procedura termina senza errori.
<code>SetDataByte(idx, value)</code>	È utilizzata per scrivere uno specifico byte sul vettore che verrà inviato via USB. In ingresso necessita dell'indice del byte da scrivere ed il valore che si vuole scrivere. Questa macro non invia fisicamente i dati ma li immagazzina nel buffer di trasmissione senza trasmetterli.
<code>SendData()</code>	Quando viene chiamata trasmette sul bus USB i dati che sono stati caricati nel buffer dalla funzione <code>SetDataByte</code> .
<code>SendDataDirect(String)</code>	Invia i dati contenuti all'interno della stringa passata come parametro sul bus USB.
<code>NumBytes = CheckRx()</code>	Aggiorna i dati che sono stati ricevuti dal canale USB. Restituisce la lunghezza dei dati presenti all'interno del buffer.
<code>Value = ReceiveByte(idx)</code>	È utilizzata per leggere un byte indentificato dall'indice passato come parametro dal buffer di lettura dei dati ricevuti. Il primo dato è identificato dall'indice zero.
<code>String = ReceiveString(NumBytes)</code>	È utilizzata per leggere l'intera stringa ricevuta sul bus USB. Richiede che gli sia passato come parametro il numero di byte da leggere e caricare sulla stringa restituita dalla funzione.

della quota associativa al consorzio USB. Coloro che producono in grandi quantità dispositivi USB hanno un loro VID, mentre le piccole aziende e gli hobbisti possono utilizzare il VID del costruttore del chip, che nel nostro caso è quello della Microchip. Per l'uso personale del dispositivo realizzato, non è necessario avere un VID valido. Il parametro PID è l'identificativo del dispositivo realizzato. A fronte dell'acquisto di un VID possiamo quindi avere a disposizione 65.535 differenti prodotti. Se volessimo vendere piccole/medie quantità di dispositivi USB potremmo domandare alla Microchip, mediante un particolare richiesta scritta disponibile sul sito www.microchip.com, l'autorizzazione ad utilizzarne uno.

All'interno del campo Name e Manufacturer possiamo inserire il nome della nostra applicazione ed il nostro nome mentre come versione indichiamo 1.0.

In **Fig. 6** vediamo le altre configurazioni specifiche al dispositivo che intendiamo realizzare. I parametri principali che dobbiamo impostare sono:

- **Maximum current**; corrente massima che il dispositivo è abilitato ad assorbire tramite la tensione di alimentazione dell'USB;
- **Transmitpacketsize**; dimensione dei pacchetti trasmessi al computer dal dispositivo durante l'interrupt sul bus;
- **Transmitpacketperiod**; numero di millisecondi che intercorrono tra due trasmissioni di dati;
- **Receivepacketsize**; dimensione dei pacchetti ricevuti dal dispositivo durante l'interrupt del bus;
- **Receivepacketperiod**; numero di millisecondi che intercorrono tra due ricezioni di dati;
- **Subclass**; è la sottoclasse dei dispositivi USB ed è valida unicamente per mouse e

Fig. 5

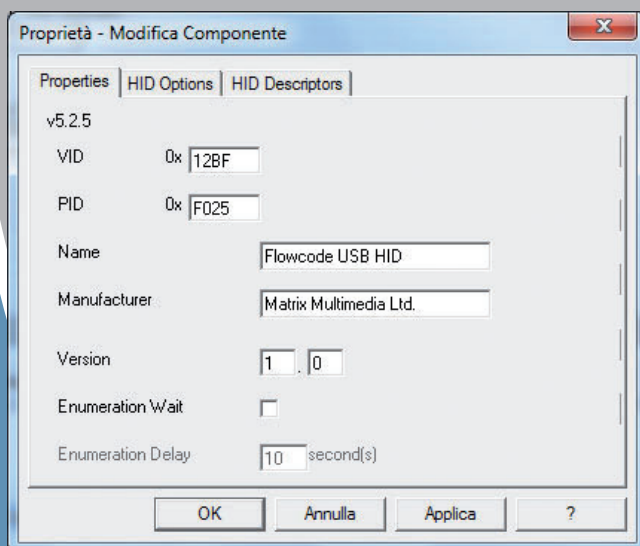


Fig. 6

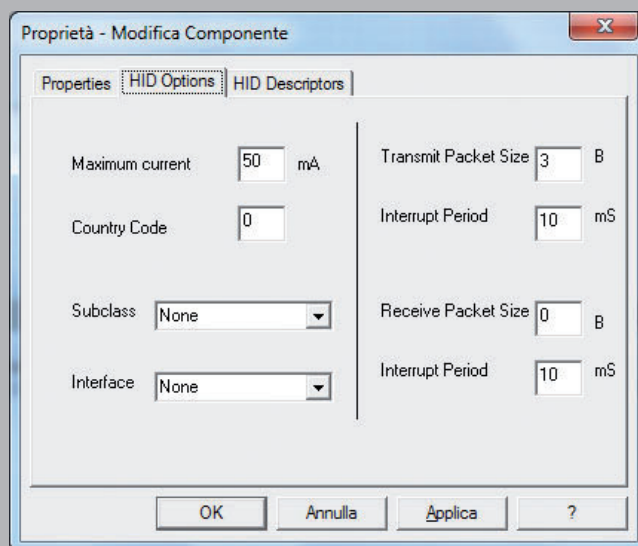
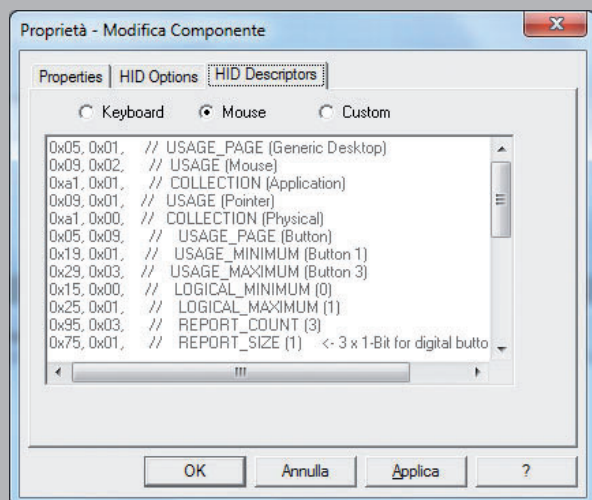


Fig. 7



tastiere (HID);

- **Interface**; è utilizzata per definire la classe del dispositivo.

Il tab "HID Descriptor" visibile in Fig. 7 permette di descrivere in modo dettagliato la comunicazione ed il significato di ogni singolo byte trasmesso. La descrizione dei dispositivi è standard quindi, qualora volessimo ad esempio realizzare un gamepad, ci basterà cercarne la sua descrizione su Internet ed inserirla all'interno della nostra finestra. In modo automatico, appena connesso il dispositivo al computer, verrà riconosciuto come tale senza bisogno di altre configurazioni ed installazioni. Le macro che Flowcode ci mette a disposizione sono riassunte nella **Tabella 1**. Grazie a queste semplici funzioni possiamo realizzare qualsiasi tipo di dispositivo HID ma per meglio capirne il funzionamento proviamo ora a mettere in pratica questi concetti realizzando un tastierino USB.

Progetto pratico: realizziamo un tastierino numerico USB

Il dispositivo che stiamo realizzando sarà riconosciuto in automatico dal PC al quale verrà collegato come una normale tastiera USB. Per realizzarlo abbiamo bisogno dei seguenti componenti:

- EB006 "USB PICmicro multiprogrammer board" (PIC18F4450 e quarzo 12 MHz);
- EB014 "Keypadboard";
- EB055 "USB interfaceboard".

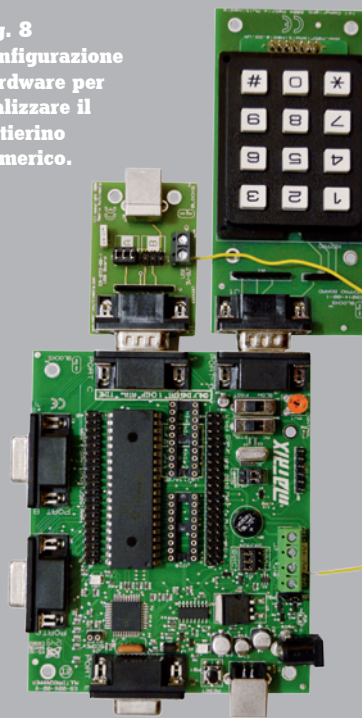
Collegiamo la scheda EB014 alla porta D e

la EB055 alla porta C, come mostrato in Fig. 8. Per poter utilizzare il componente senza doverlo alimentare esternamente, come le normali tastiere USB, basta collegare il morsetto V+ della scheda EB006 al morsetto +5 della scheda EB055. In questo modo il microcontrollore viene alimentato direttamente dalla porta USB. La connessione è effettuata mediante il filo giallo mostrato nella stessa Fig. 8.

È inoltre necessario, come visto nelle precedenti puntate, configurare la scheda in modo da avere la frequenza adatta per poter utilizzare correttamente il bus USB. Ogni volta che è necessario gestire un bus a frequenza fissa bisogna stare molto attenti nel configurare correttamente tutti i parametri; nel nostro caso la configurazione è mostrata in Fig. 9. La frequenza di oscillazione del PLL è quindi 96 MHz, ma le periferiche interne lavorano a 24 MHz in modo da poter supportare la connessione USB Low-Speed.

Ora che l'hardware è pronto, possiamo dedicarci alla realizzazione del software di controllo: per prima cosa dobbiamo capire quali dati vengono scambiati tra PC e tastiera in modo da poterla emulare; allo scopo, possiamo andare sul sito ufficiale USB e scaricare i file "Hut1_12v2.pdf" e "HID1_11.pdf". Analizzandoli, riusciamo a capire cosa viene trasmesso e cosa è ricevuto. In Fig. 10 sono riassunti i dati trasmessi tra PC e tastiera USB. Sempre nel file, troveremo la codifica

Fig. 8
Configurazione
hardware per
realizzare il
tastierino
numerico.



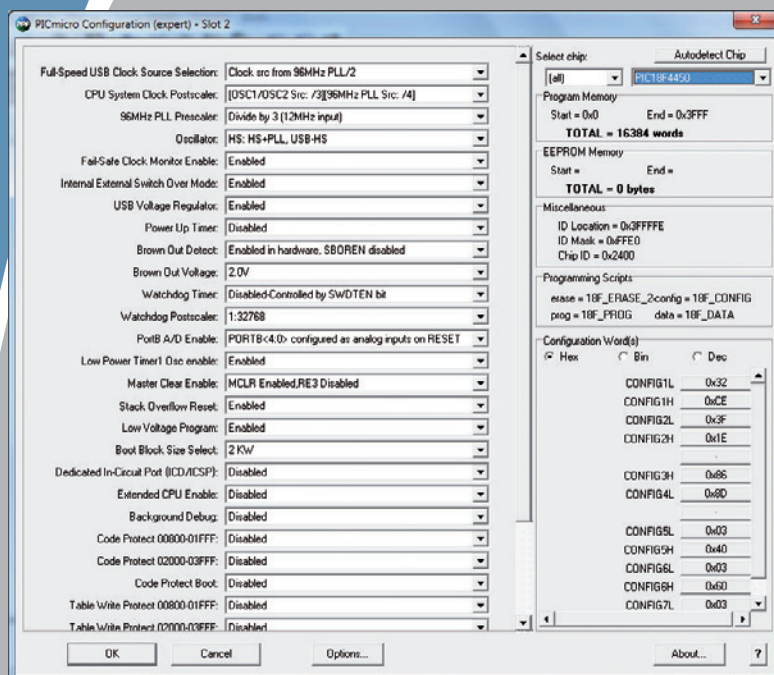


Fig. 9 - Configurazione del microcontrollore per la realizzazione del tastierino numerico.

dei caratteri da trasmettere perché questi non seguono la normale tabella ASCII. I valori numerici dall'uno al 9 partono dal valore esadecimale 0x1E mentre lo zero ha codifica 0x27. In questo semplice esempio trasmetteremo unicamente un carattere per volta e non saranno quindi possibili combinazioni di tasti premuti in contemporanea. Per questa ragione ci basterà valorizzare unicamente il byte 2, lasciando gli altri al valore predefinito 0x00.

Ora che abbiamo tutte le informazioni necessarie possiamo trascinare nel pannello il componente Keypad ed il componente USB-HID. Il componente Keypad richiede una corretta configurazione delle connessioni. Nel nostro caso dovranno essere impostate come mostrato in Fig. 11.

Per quanto riguarda la configurazione del componente HID, è necessario scegliere nella pagina HID descriptors l'opzione Keyboard mentre configurare il tab HID options come indicato in Fig. 12.

L'applicazione potrebbe sembrare complessa, per il fatto che utilizza il bus USB, ma il diagramma di controllo risulta semplice e non richiede neppure di realizzare macro di supporto. Come primo passo è necessario inizializzare la periferiche HID e successivamente, mediante un ciclo infinito,

scandire il tastierino numerico e trasmettere periodicamente, mediante la macro di aggiornamento della periferica HID, le informazioni sul tasto premuto. Se nessun tasto è premuto, viene trasmesso il valore 0x00 altrimenti il codice relativo al tasto. Per evitare di sovraccaricare il bus di dati non utili, qualora il tasto premuto risulti sempre lo stesso, i dati vengono trasmessi unicamente a seguito di una variazione del tasto premuto ed il tempo minimo di ritardo tra una trasmissione e l'altra è di 10 ms. Per ovviare a problemi di inizializzazione è stato aggiunto un ritardo di inizializzazione pari a 100 ms.

Le variabili utilizzate sono le seguenti:

- DatiUSB[8]: è un vettore di 8 byte nel quale vengono memorizzati i dati da trasmettere sul bus;
- EsitoInizializzazione: variabile byte che serve per memorizzare l'esito dell'inizializzazione della periferica USB;
- Keypad: memorizza il valore letto dal Keypad in una variabile di tipo byte;
- Keypad_precedente: è utilizzata per memorizzare lo stato precedente letto sul keypad ed è di tipo byte.

Il flusso di programma completo è mostrato in Fig. 13.

Fig. 10 - Riassunto dati scambiati con la tastiera USB.

Byte	Description
0	Modifier keys
1	Reserved
2	Keycode 1
3	Keycode 2
4	Keycode 3
5	Keycode 4
6	Keycode 5
7	Keycode 6

Note Byte 1 of this report is a constant. This byte is reserved for OEM use. The BIOS should ignore this field if it is not used. Returning zeros in unused fields is recommended.

The following table represents the keyboard output report (1 byte).

Bit	Description
0	NUM LOCK
1	CAPS LOCK
2	SCROLL LOCK
3	COMPOSE
4	KANA
5 to 7	CONSTANT

Note The LEDs are absolute output items. This means that the state of each LED must be included in output reports (0 = off, 1 = on). Relative items would permit reports that affect only selected controls (0 = no change, 1 = change).

Installiamo il driver del componente FTDI

Per poter funzionare correttamente, il PC al quale colleghiamo il convertitore FTDI, deve avere installati i driver necessari. Per far ciò inseriamo il CD fornito assieme alla scheda EB001 a fronte della richiesta del PC di installare il driver e indichiamo di volerlo cercare nel computer (Fig. A); inseriamo l'indirizzo alla cartella "\drivers\EB-O39" contenuta all'interno del nostro lettore CD e procediamo premendo il pulsante "Avanti".

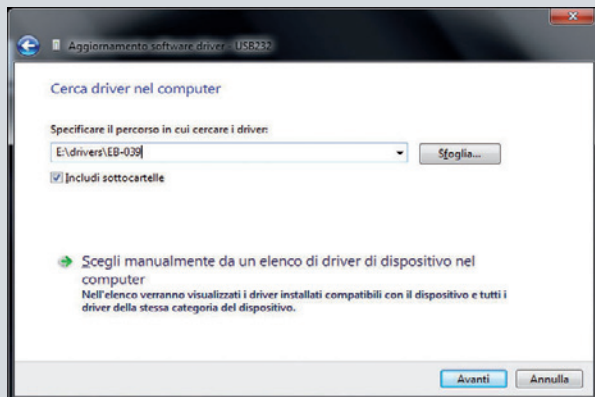


Fig. A - indicazione del percorso di ricerca del driver USB.

Al termine della procedura di installazione otterremo l'indicazione che il driver è stato aggiornato ed è quindi pronto all'uso come indicato in Fig. B.

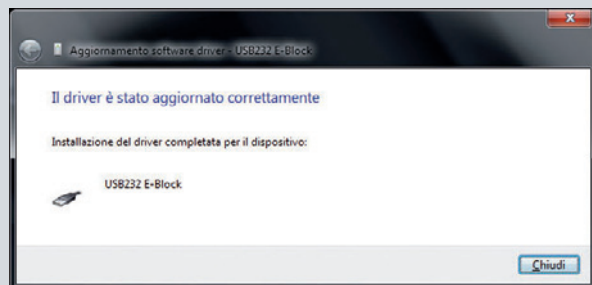


Fig. B - Installazione driver completata con successo.

Il componente USB-Serial

Questo tipo di controllo permette di gestire in modo molto semplice una connessione USB perché è vista sia lato PC che lato Sistema embedded come una comunicazione di tipo seriale RS232. Il componente si trova nella barra dei componenti all'interno del menu "periferiche" dove avevamo precedentemente visto gli altri componenti USB. Come possiamo osservare nella finestra di dialogo visibile in Fig. 14, i parametri che possiamo configurare sono gli stessi che avevamo visto per il componente HID; il

pulsante "Generate Driver File" che si trova in essa serve per generare il file .inf che viene richiesto quando colleghiamo la periferica ad un PC per l'installazione del driver. È necessario rigenerare il driver ogni qualvolta, nella finestra di dialogo, modifichiamo i parametri di configurazione. Le macro a disposizione sono elencate in Tabella 2. Con queste semplici macro possiamo realizzare qualsiasi tipo di periferica USB che emuli una comunicazione seriale. Per meglio capirne il funzionamento passiamo ora alla realizzazione di un semplice analizzatore di stati logici.

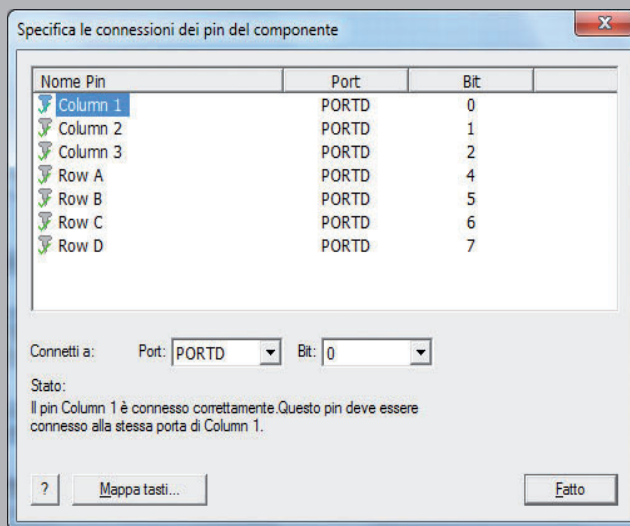


Fig. 11 - Configurazione Keypad.

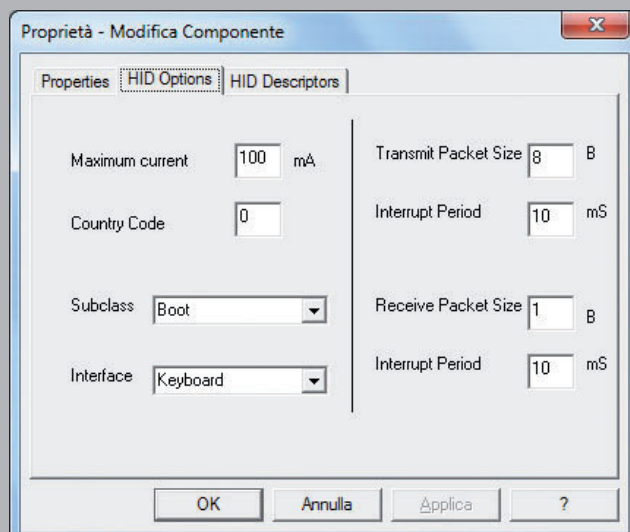


Fig. 12 - Configurazione USB-HID.

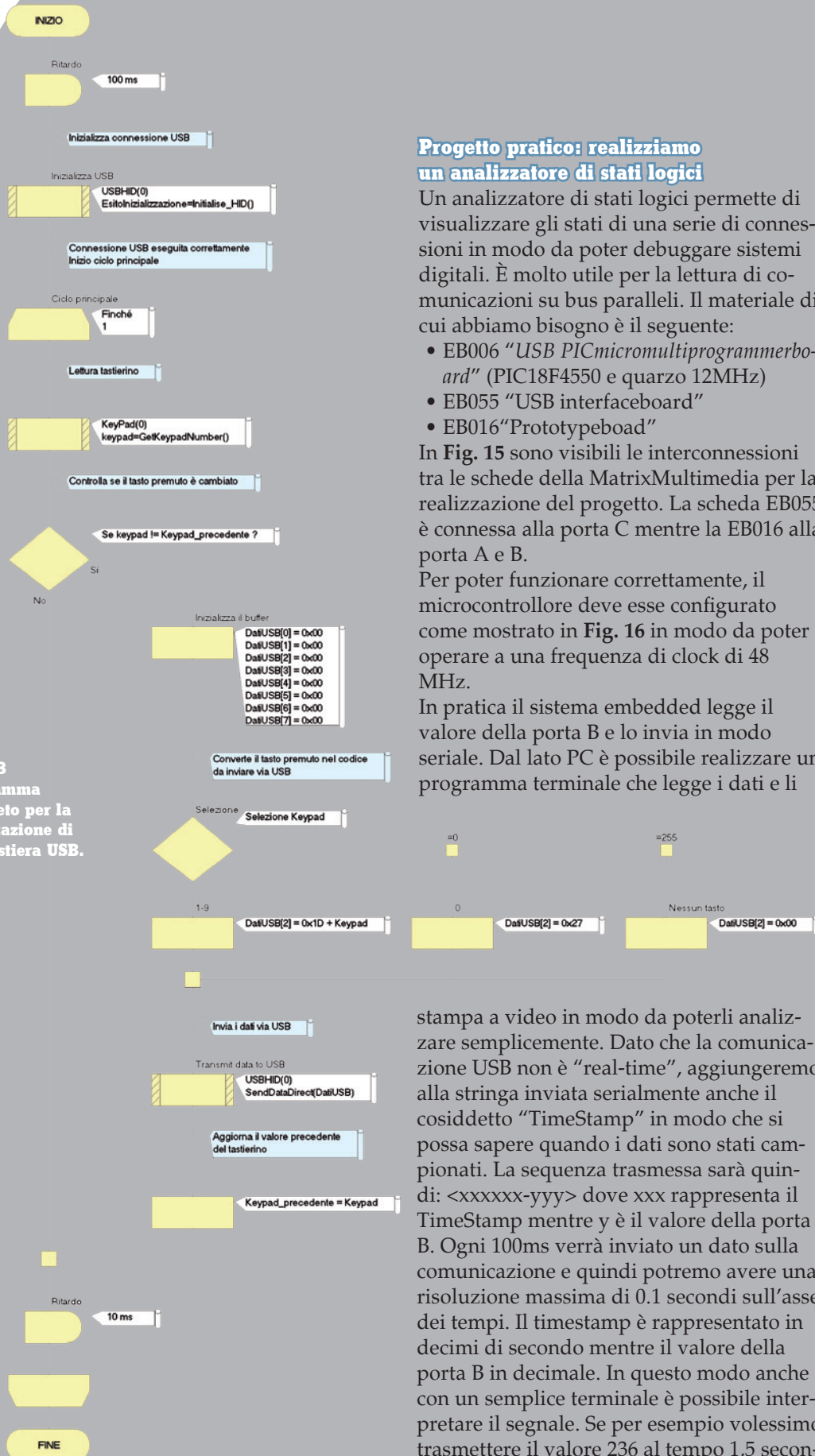


Fig. 13
Diagramma completo per la realizzazione di una tastiera USB.

Progetto pratico: realizziamo un analizzatore di stati logici

Un analizzatore di stati logici permette di visualizzare gli stati di una serie di connessioni in modo da poter debuggare sistemi digitali. È molto utile per la lettura di comunicazioni su bus paralleli. Il materiale di cui abbiamo bisogno è il seguente:

- EB006 "USB PICmicromultiprogrammerboard" (PIC18F4550 e quarzo 12MHz)
- EB055 "USB interfaceboard"
- EB016 "Prototypeboard"

In Fig. 15 sono visibili le interconnessioni tra le schede della MatrixMultimedia per la realizzazione del progetto. La scheda EB055 è connessa alla porta C mentre la EB016 alla porta A e B.

Per poter funzionare correttamente, il microcontrollore deve essere configurato come mostrato in Fig. 16 in modo da poter operare a una frequenza di clock di 48 MHz.

In pratica il sistema embedded legge il valore della porta B e lo invia in modo seriale. Dal lato PC è possibile realizzare un programma terminale che legge i dati e li

stampa a video in modo da poterli analizzare semplicemente. Dato che la comunicazione USB non è "real-time", aggiungeremo alla stringa inviata serialmente anche il cosiddetto "TimeStamp" in modo che si possa sapere quando i dati sono stati campionati. La sequenza trasmessa sarà quindi: <xxxxxx-yyy> dove xxx rappresenta il TimeStamp mentre y è il valore della porta B. Ogni 100ms verrà inviato un dato sulla comunicazione e quindi potremo avere una risoluzione massima di 0.1 secondi sull'asse dei tempi. Il timestamp è rappresentato in decimi di secondo mentre il valore della porta B in decimale. In questo modo anche con un semplice terminale è possibile interpretare il segnale. Se per esempio volessimo trasmettere il valore 236 al tempo 1,5 secon-

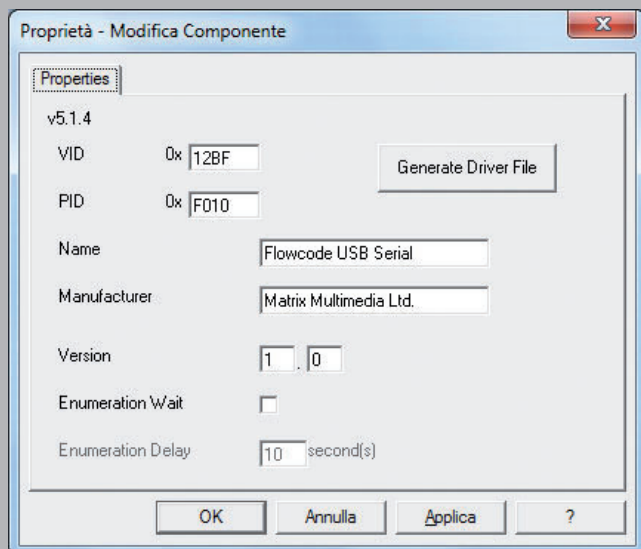


Fig. 14
Finestra delle
proprietà del
componente
USB seriale.

di invieremo la stringa "<000015-236>. Il tempo massimo è di 99999.9 secondi (27,7 giorni) dopo il quale, si ha la ripetizione del valore di TimeStamp.

Generare la stringa a lunghezza fissa, cioè con l'aggiunta dello zero per mantenere la lunghezza fissa, è probabilmente la parte più difficile di tutto il progetto perché il comando per la conversione da numero a stringa inserisce unicamente le cifre significative. Per questa ragione realizzeremo una macro che avrà il compito di convertire un numero e di portarlo a lunghezza fissa. La macro si chiama "ConvertiInStringaFissa" e necessita dei seguenti parametri:

- ValoreNumerico: è il valore che vogliamo convertire;
- LunghezzaFissa: è la lunghezza della stringa in uscita.

Il primo ciclo si occupa della formattazione della stringa mediante l'aggiunta del carat-



**Fig. 15 - Configurazione
HW per analizzatore
di stati logici.**

tere che identifica lo zero mentre il secondo carica all'interno del valore di ritorno il valore convertito mediante la funzione Length\$ con allineamento a destra come mostrato in Fig. 17.

Ora che la nostra macro è pronta possiamo passare alla realizzazione del programma principale. Per prima cosa inseriamo all'interno del pannello il componente USB-Seriale e lo configuriamo come visto nel paragrafo precedente. Come di routine suddividiamo il programma in due parti principali: l'inizializzazione del sistema ed il ciclo principale. Nell'inizializzazione inseriamo l'azzeramento delle variabili e la macro "Intitalize_Serial()" per la connessione USB. Il ciclo principale si occupa invece della sequenza di lettura della portaB, Conversione del valore letto e del TimeStamp, generazione della stringa da trasmettere e infine trasmissione del dato su bus USB. Per fare in modo che i dati ven-

Tabella 2 - Macro disponibili per la gestione della connessione USB-Seriale.

Nome Macro	Descrizione
Initialise_Serial()	È utilizzata per inizializzare la comunicazione USB avviando il servizio di enumerazione in modo che il PC possa identificare il dispositivo. Restituisce come valore di ritorno 0 qualora venga eseguita correttamente mentre 255 in caso di errore.
Send_Byte(Byte)	Trasmette il byte passato come parametro inserendolo nel buffer di scrittura.
Send_String(String, Length)	Invia la stringa passata come parametro "String". Il parametro "Length" deve contenere la lunghezza dei caratteri da trasmettere contenuti nella stringa.
Read_Byte(timeout_ms)	Legge dal buffer di lettura un byte e lo restituisce come valore di ritorno. Il parametro "timeout_ms" serve per stabilire un timeout massimo, in millisecondi, per il quale la macro deve attendere in attesa di un dato. Qualora venga posto al valore 255 l'attesa non termina fino a quando un dato non è disponibile nel buffer di ricezione. Nel caso nessun dato venga ricevuto entro il timeout il valore di ritorno è 255.
Read_String(timeout_ms, String, Length)	Legge dal buffer di lettura una sequenza di caratteri la cui lunghezza è fissata dal parametro "Length" e la restituisce come valore di ritorno della funzione. Come per la macro Read_Byte il parametro "timeout_ms" serve per permettere alla macro di restituire il controllo alla funzione chiamante senza che nessun dato sia stato ricevuto.

Fig. 18
Diagramma
del "Main"
dell'analizzatore
di stati logici.

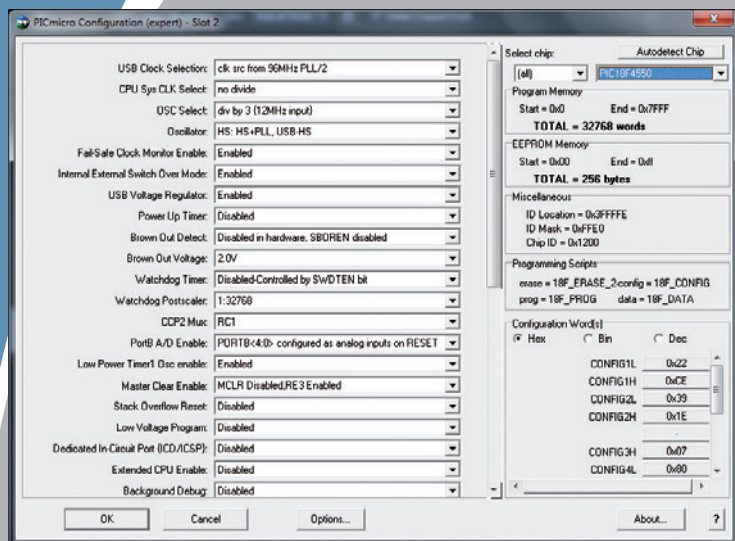


Fig. 16 – Configurazione microcontrollore per analizzatore di stati logici.

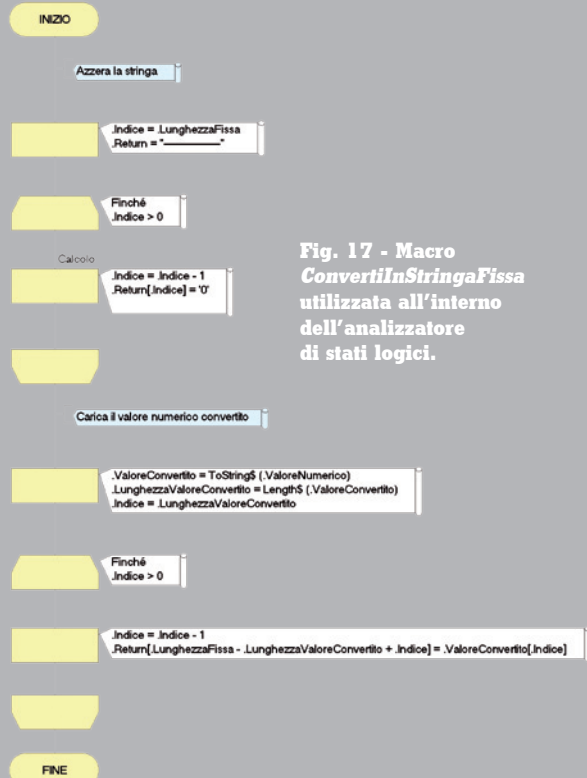


Fig. 17 - Macro *ConvertiInStringaFissa* utilizzata all'interno dell'analizzatore di stati logici.

gano campionati e trasmessi ogni 100ms è necessario inserire un ritardo di 100ms. Il ciclo principale è, come sempre, racchiuso all'interno di un ciclo infinito per evitare che il programma termini dopo il primo invio di dati. I più esperti nella realizzazione di sistemi embedded si saranno sicuramente accorti del fatto che utilizzare un delay per cadenzare una misurazione per periodi così lunghi porta ad un accumulo dell'errore sul timestamp. Per ovviare a questa problematica è comunque sufficiente utilizzare un timer. In questo esempio è stato scelto di non utilizzarlo unicamente per poterci soffermare maggiormente sull'impiego del bus di comunicazione piuttosto che sulle altre problematiche di contorno che, a questo punto del corso, dovrete essere già in grado di risolvere in modo autonomo. In **Fig. 18** è mostrato il diagramma principale per la realizzazione dell'analizzatore di stati logici.

Per testare il programma possiamo utilizzare un qualsiasi terminale per porte COM che supporti le comunicazioni seriali virtuali.

Bene, anche con questa puntata abbiamo concluso. Nelle prossime conosceremo altri componenti forniti da Flowcode, tra i quali lo ZigBee e l'Ethernet.