

Mensile di progettazione elettronica, attualità scientifica, novità tecnologiche.

€ 6,00

Elettronica In

oltre l'elettronica

www.elettronica.in.it

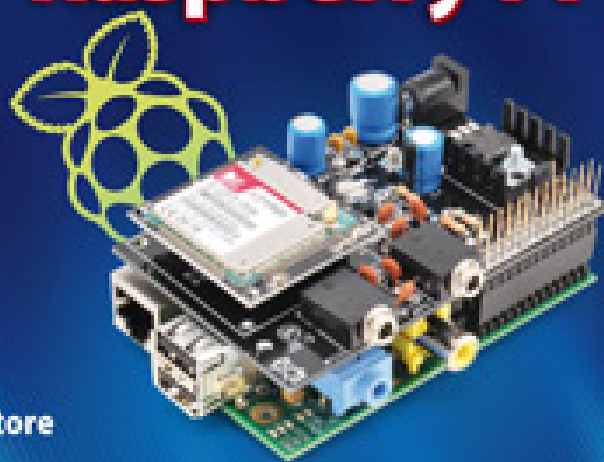
Anno XIX - n. 179
Settembre 2013



**ARRIVA
IN ITALIA**

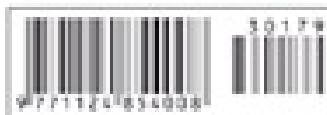
Maker Faire

**Shield GSM / GPRS / GPS
per Raspberry Pi**



- Antifurto Bluetooth
- Tecnologia Bodycom
- Combinatore GSM vocale con SIM900
- myKIT: misuratore a distanza
- ArduLego sensor
- Corso mbed

**Frequenzimetro
digitale a microcontrollore**



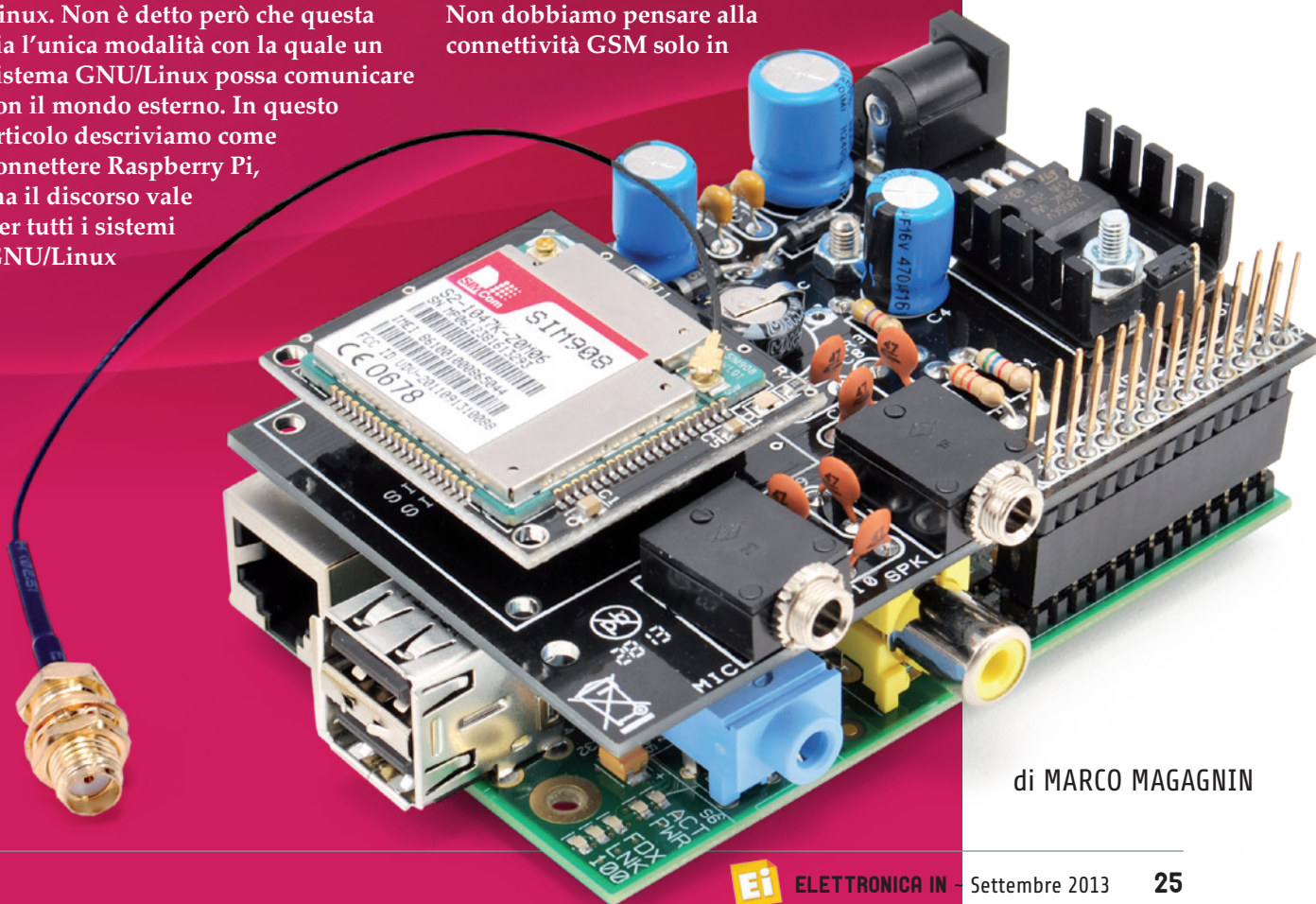
SHIELD DI ESPANSIONE GSM/GPRS/GPS PER RASPBERRY PI

Il sistema operativo GNU/Linux nasce con la "vocazione" della connettività Internet. I protocolli TCP/IP (Transport Control Protocol – Internet Protocol) sono legati a doppio filo alla storia di UNIX prima, e dei suoi successori poi, sino all'attuale GNU/Linux. Non è detto però che questa sia l'unica modalità con la quale un sistema GNU/Linux possa comunicare con il mondo esterno. In questo articolo descriviamo come connettere Raspberry Pi, ma il discorso vale per tutti i sistemi GNU/Linux

dotati di porta seriale, ai moduli della famiglia SIM9XX. In questo modo è possibile estendere le funzionalità di Raspberry Pi per applicazioni che richiedano connettività mobile e, impiegando il modulo SIM908, anche le funzionalità GPS.

Non dobbiamo pensare alla connettività GSM solo in

Shield di espansione per Raspberry Pi col quale controllare i moduli GSM/GPRS SIM900 e SIM908 (con GPS), per realizzare applicazioni che possono essere controllate e che possono comunicare in remoto tramite rete telefonica mobile.



di MARCO MAGAGNIN

applicazioni strettamente “mobile” come quelle su automezzi o in luoghi non collegati alla rete internet. Pensiamo semplicemente ad un sistema di domotica, di antifurto o di controllo di un impianto non presidiato.

Nel caso venga a mancare l’energia elettrica, nella maggioranza dei casi, a queste applicazioni verrebbe a mancare anche la connettività internet. In questo caso, l’unica possibilità di inviare un messaggio di avviso sullo stato delle cose. Ovviamente il sistema di controllo deve essere dotato di un sistema di batteria tampone della massima affidabilità, ed in grado di poter comunicare con il



sistema GNU/Linux. Ci stiamo pensando. Per ora torniamo al nostro shield GSM/GPS e a come utilizzarlo con Raspber-

ry Pi. Come già anticipato nell’articolo nel numero 178 della rivista, gli shield dedicati a Raspberry Pi sono progettati per poter essere impilati l’uno sull’altro e poter gestire più funzioni contemporaneamente, in modo da poter integrare diversi “servizi” in un’unica soluzione integrata. Potrebbe essere il caso, appunto, di un sistema domotico, dotato di sensori e attuatori, che utilizzi l’LCD per l’attivazione e disattivazione, ma che renda possibile la stessa operazione anche da remoto, sia via rete internet che da GSM. Inoltre a questo possono essere aggiunti servizi di sicurezza per anziani, attivabili sia in locale che da remoto, con funzionalità vocali o di invio SMS a numeri predefiniti. Possiamo aggiungere la presa di immagini con videocamera, associata all’apertura delle porte e con riconoscimento facciale, o di movimento, con l’invio di messaggi e delle stesse immagini in caso di ... sospetto. L’unico limite è la fantasia e ... la capacità di progettare e realizzare le applicazioni. Applicazioni di questa portata superano le possibilità dei microcontrollori in genere, anche se questi sono utilizzabili con profitto nella gestione dei sensori e dei dispositivi periferici. Tornando sul piano concreto, chiaramente i diversi shield non devono utilizzare gli

I moduli GSM/GPRS utilizzati

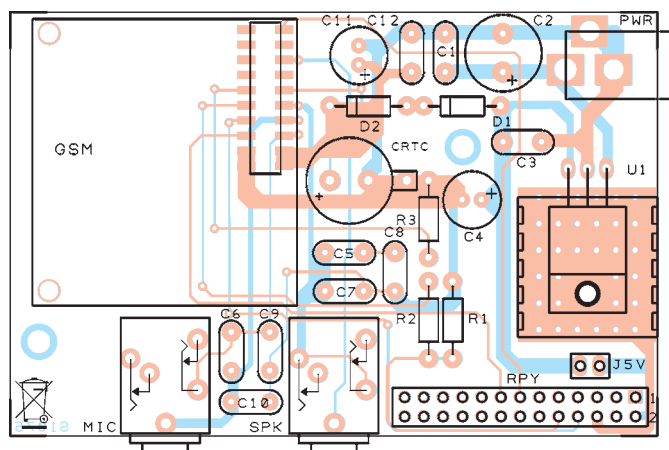
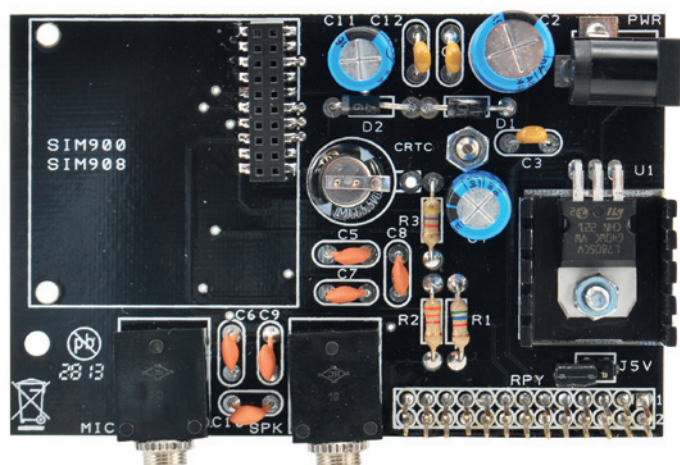


Lo shield per Raspberry Pi presentato in questo articolo può ospitare sia il modulo SIM900 (GSM/GPRS) che il modulo SIM908 (+GPS). Più precisamente il modulo SIM900 è un Quad-band compatibile GSM/GPRS, operante sulle frequenze di 850/900/1800/1900 MHz, con SMS e accessi a Internet mediante stack TCP/IP integrato, gestito da un processore AMR926EJ-S che si occupa anche della gestione della comunicazione telefonica mediante comandi AT. Il modulo, compatibile GSM fase 2/2+, è di classe 4 (2 W) a 850/ 900 MHz e di classe 1 (1 W) a 1800/1900 MHz; SIM900 integra



un’interfaccia analogica, un A/D converter, un RTC, un bus SPI, uno I²C, ed un modulo PWM. Il modulo si alimenta a tensione continua di valore compreso fra 3,4 e 4,5 V, assorbe un massimo di 0,3 A (continuativi) in trasmissione. È dotato di connettore per antenna esterna.

Il SIM908 integra un ricevitore GPS (con antenna separata) a 42 canali capace di una sensibilità di -160 dB in tracking e -143 dB in acquisizione, di una precisione nella localizzazione di soli 2,5 m e di tempi di operatività di 1 s per l’Hot Start e 30 secondi per il Cold Start. Il set di comandi AT è esteso per l’accesso al GPS. I manuali sono reperibili in PDF in internet cercando “SIM900 AT command” e “SIM908 AT command”.



Elenco Componenti:

R1: 5,6 kohm
 R2: 3,3 kohm
 R3: 4,7 kohm
 C1: 100 nF multistrato
 C2: 470 μ F 25 VL elettrolitico
 C3: 100 nF multistrato
 C4: 470 μ F 16 VL elettrolitico
 C5÷C9: 47 pF ceramico
 C10: 47 nF ceramico
 C11: 470 μ F 16 VL elettrolitico

C12: 100 nF multistrato
 U1: 7805
 D1: 1N4007
 D2: 1N5819
 CRTC: Condensatore basso profilo 0,1F

Varie:
 - Plug alimentazione
 - Presa jack stereo 3,5 mm da CS (2 pz.)
 - Dissipatore

- Vite 8 mm 3 MA
 - Vite 12 mm 3 MA
 - Dado 3 MA (2 pz.)
 - Jumper
 - Strip maschio 2 poli
 - Presa MMS SMT 2x10 vie passo 2 mm
 - Connettore 13x2 RaspberryPi
 - Torretta M/F 18 mm
 - Circuito stampato

foro vicino al condensatore CRTC e fissarlo con il dado M3;

- Posizionare il connettore a 26 pin nelle piazzole identificate con RPY, tenendo il lato con i terminali femmina verso il basso;
- Applicare lo shield su Raspberry Pi facendo appoggiare su quest'ultima il distanziale esagonale. Unire i due connettori da 26 pin, e fissare il distanziale con la vite M3;
- Una volta che le schede sono allineate e distanziate correttamente è possibile saldare sullo shield i terminali maschio del connettore.

Sempre per garantire la possibilità di "impacchettamento" posizionare con attenzione l'aletta di raffreddamento dell'integrato U1, in modo che non vada a toccare l'eventuale shield montato al di sopra.

UTILIZZO PRATICO DELLO SHIELD

Come prima operazione monteremo sullo shield uno dei moduli GSM compatibili e poi, se non ancora fatto, monteremo lo shield sul connettore di Raspberry Pi, facendo attenzione che la parte inferiore dello shield non vada in contatto con i connettori USB o Ethernet. In caso di dubbio proteggeremo i connettori stessi con del nastro isolante. Colleghiamo le periferiche, la rete e l'alimentazione a Raspberry Pi nel solito modo. Colleghiamo l'alimentazione allo shield FT1075 e ... diamo tensione.

In questo articolo vi proponiamo un approccio didattico all'utilizzo dello shield GSM in modo da dare la possibilità di sperimentare il più possibile prima di iniziare a scrivere applicazioni più o meno complesse. Ricordiamo che utiliz-

zando GNU/Linux possiamo realizzare applicazioni che svolgono molte funzioni, condividendo risorse come lo shield GSM. Ad esempio possiamo scrivere un'applicazione che resti in attesa di eventi esterni come l'arrivo di una chiamata voce od un messaggio SMS, ma sia anche in grado di reagire ad eventi interni come l'attivazione di ingressi di I/O, sia digitali che analogici, oppure alla richiesta di esecuzione di chiamate voce, invio di SMS, singoli o di massa, invio di file, messaggi di posta elettronica ecc. Nel frattempo si può desiderare di acquisire dati dalla sezione GPS del modulo. Per fare questo, come già descritto per lo shield LCD, è necessario progettare un'architettura modulare in grado di condividere le risorse senza creare condizioni di conflitto. Ne risulta un'architettura composta da

moduli server che interfacciano e controllano le risorse e i moduli operativi ciascuno specializzato in una specifica funzione che possono essere richiamati ed eseguiti al verificarsi di specifiche condizioni e richiedere servizi di comunicazione ai moduli server, che provvederanno a soddisfarli secondo una logica che garantisca le priorità ed eviti i conflitti. Per fare questo è necessario approfondire fin nei dettagli le esigenze funzionali dell'applicazione ed è il motivo per il quale, prima di addentrarci, nei prossimi numeri, nella progettazione di applicazioni client/server, vi proponiamo un paio di modalità che permettano di sperimentare in modo agile l'utilizzo dello shield.

La prima modalità permette di riutilizzare con poche modifiche i programmi scritti per Arduino, in modo da prendere dimestichezza con lo shield per eredità dal mondo Arduino.

In questo modo si potranno utilizzare gli esempi proposti nei numeri scorsi della rivista nella serie di articoli "Gestire i moduli GSM con Arduino". Per questa modalità utilizzeremo la libreria arduPi realizzata da Libelium Comunicaciones Distribuidas S.L., messa a disposizione in licenza GNU General Public License e reperibile da CookingHacks che vi abbiamo già presentato nell'articolo "Shield di espansione ADC per Raspberry Pi" nel numero 174 della rivista.

La seconda modalità è una modalità diretta. Ci colleghiamo alla porta seriale con un programma TTY e comandiamo il modulo GSM dando e ricevendo direttamente i comandi di gestione. Questo ci permette di provare le sequenze di comandi necessari a realizzare una specifica funzionalità senza la necessità di scrivere e testare ogni volta un programma.

INVIAMO UN SMS IN STILE ARDUINO

Della libreria arduPi abbiamo già parlato nel numero 174 della rivista. Ci limitiamo a ricordare brevemente le modalità di installazione ed utilizzo. Per installare la libreria arduPi è sufficiente scaricarla dall'indirizzo:

http://www.cooking-hacks.com/skin/frontend/default/cooking/images/catalog/documentation/raspberry_arduino_shield/arduPi_1-5.tar.gz

La scompattiamo nella cartella /home e, per comodità, ne modifichiamo il nome in "ArduPi". Sempre per comodità metteremo nella stessa cartella i programmi che andremo a realizzare. Prima di compilare la libreria apriamo il programma arduPi.cpp e, se diversa, sostituiamo alla riga 64 il riferimento alla porta seriale esistente con la riga:

```
serialPort="/dev/ttyAMA0";
```

Ora andiamo nella cartella ArduPi (Fig. 2) con il comando

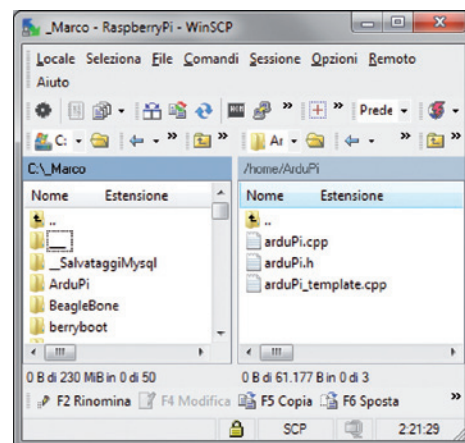
```
cd /home/ArduPi
```

e compiliamo la libreria con il comando (Fig. 3):

```
g++ -c arduPi.cpp -o arduPi.o
```

Per quanto riguarda i programmi vi proponiamo tre moduli. Due permettono di accendere e spegnere il modulo GSM e ci saranno molto utili in futuro.

Il terzo ci permette di inviare un SMS ad un telefono cellulare. Per



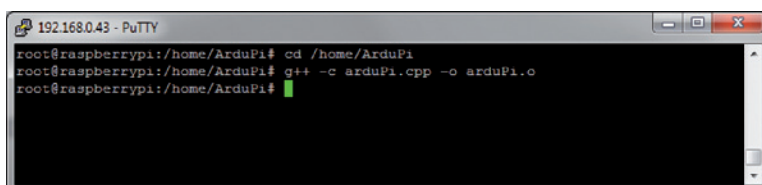
accendere il modulo è necessario alzare il livello del pin GPIO27 per due secondi e poi portarlo di nuovo a livello basso, stessa operazione per spegnerlo, se questo è acceso. Per il reset è necessario alzare e mantenere a livello alto per mezzo secondo il pin GPIO22, per poi riportarlo a livello basso. I programmi sono scaricabili liberamente dal sito www.elettronica.in.

Prima di tutto, per poter utilizzare la porta seriale, dobbiamo eseguire una modifica al processo di boot per disattivare la console seriale, che altrimenti impegnerebbe la porta seriale.

DISABILITIAMO LA CONSOLE SERIALE

Pressoché tutti i sistemi GNU/Linux attivano, alla partenza, la funzionalità console, cioè la possibilità di collegarsi alla linea seriale con un terminale - reale o virtuale - per gestire tutto il sistema tramite questa interfaccia di comunicazione. Un ricordo dei tempi nei quali ci si collegava ai computer via modem utilizzando le normali linee telefoniche. A volte è ancora l'unica possibilità di accesso ai sistemi embedded o server. Per

Fig. 3



Listato 1

```

/*
 * GSMAccendi
 */

//Include Arduino library
#include "arduino.h"

int resetModulePin = 9;
int onModulePin = 8;

void switchModuleOn(){
    digitalWrite(onModulePin,HIGH);
    delay(2000);
    digitalWrite(onModulePin,LOW);
}

void resetModule(){
    digitalWrite(resetModulePin,HIGH);
    delay(500);
    digitalWrite(resetModulePin,LOW);
    delay(100);
}

int main (){
    Serial.begin(115200);
    delay(2000);
    pinMode(resetModulePin, OUTPUT);
    pinMode(onModulePin, OUTPUT);
    Serial.flush();
    printf ("zero\n");
    Serial.print("AT");
    delay(1000);
    if (Serial.available()==0)
    {
        printf ("uno\n");
        resetModule();
        delay(2000);
    }
    Serial.print("AT");
    delay(1000);
    if (Serial.available()==0)
    {
        printf ("due\n");
        switchModuleOn();
    }

    return (0);
}

```

disabilitare la console utilizziamo WINScp, andiamo nella cartella "boot" con il comando:

```
cd /boot
```

e apriamo, con un doppio clic, il file /boot/cmdline.txt eliminiamo, facendo attenzione, i parametri di configurazione (Fig. 4):

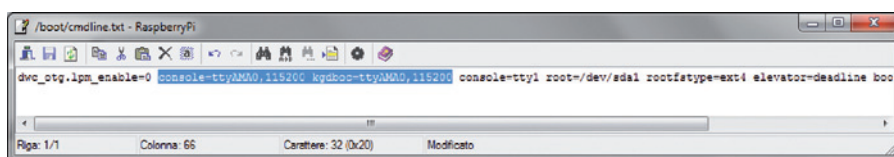


Fig. 4

```
console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
```

Salviamo il file e chiudiamolo.

ESEMPI DI PROGRAMMI (ON/OFF DEL MODULO E INVIO SMS)

Vediamo la logica del programma di accensione del modulo, ispezionabile nel Listato 1. Come prima considerazione potete verificare che i programmi sono codificati per eseguire un'unica sequenza di istruzioni, contrariamente ai programmi per Arduino, che funzionano all'infinito all'interno di un ciclo loop(). Questo permette di utilizzare i singoli programmi come moduli di un'applicazione di livello superiore.

Il processo di accensione del modulo GSM prevede i seguenti passi:

- Controllo, mediante l'invio sulla seriale di un comando "AT" per verificare che il modulo non sia già acceso;
- In caso di risposta negativa (all'invio di un comando AT, in situazioni normali si riceve di ritorno la stringa "OK"), come ulteriore controllo, viene eseguita la procedura di reset con il successivo invio di un secondo comando "AT";
- In caso di ulteriore risposta negativa viene eseguita la procedura di accensione che consiste nel portare il pin GPIO27 a livello alto per due secondi.

Lo scopo dei controlli preliminari è dovuto al fatto che il processo di accensione e spegnimento è lo stesso, quindi, se si esegue il programma di accensione senza

controlli, se il modulo GSM è già acceso, lo si spegne. Per compilare il programma GSMAccendi si utilizza il comando:

```
g++ -lrt -lpthread GSMAccendi.cpp
arduino.o -o GSMAccendi
```

Per eseguire il programma, dopo essersi posizionati nella cartella Arduino con il comando:

```
cd /home/ArduPi
```

si digita:

```
./GSMAccendi
```

Possiamo seguire la procedura di accensione del modulo GSM osservando il LED verde, all'accensione rimane acceso per circa due secondi, poi si spegne ed inizia a lampeggiare con intervalli di circa mezzo secondo, segno che il modulo è alla ricerca della rete GSM. Quando la rete è agganciata il LED lampeggia con una frequenza molto più bassa, circa un lampeggio al secondo.

Nel Listato 2 è visibile il programma per lo spegnimento del modulo, che esprime una logica simile al programma precedente:

- Controllo, mediante l'invio sulla seriale di un comando "AT" per verificare che il modulo non sia già spento;
- In caso di risposta negativa, come ulteriore controllo, viene eseguita la procedura di reset con il successivo invio di un secondo comando "AT";
- In caso di risposta positiva viene eseguita la procedura di spegnimento che consiste nel portare il pin GPIO27 a livello alto per due secondi.

Anche in questo caso lo scopo dei controlli preliminari è dovuto al fatto che il processo di accensione e spegnimento è lo stesso, quindi, se si esegue il programma di

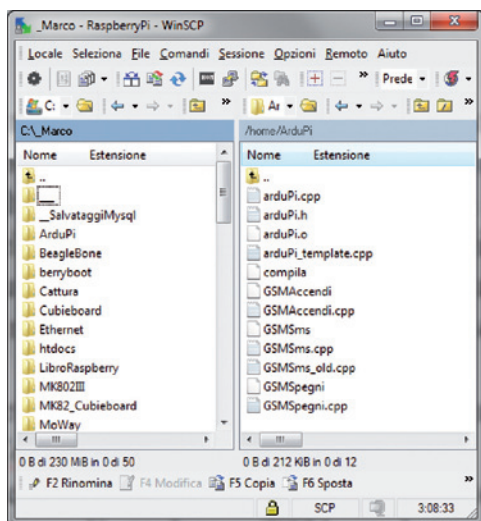


Fig. 5

- Infine vengono inviati i caratteri CTRL-Z per chiudere la chiamata.

Per compilare il programma si digita il comando:

```
g++ -lrt -lpthread GSMsms.cpp
arduPi.o -o GSMsms
```

Per eseguire il programma si esegue il comando:

```
./GSMsms
```

Prima di eseguire il programma è necessario eseguire il programma di accensione del modulo GSM, per sincerarsi che il modulo sia effettivamente acceso e funzionante. Quando si andrà a sviluppare l'applicazione di controllo, sarà compito di quest'ultima gestire la sequenza corretta di chiamata dei moduli, con i relativi controlli di stato e gestione delle condizioni anomale.

Per sperimentare altri programmi potete riferirvi agli esempi presenti nella serie di articoli "Gestire i moduli GSM con Arduino", eventualmente trasformandoli in moduli monofunzionali, come quelli descritti in questo articolo. Nella Fig. 5 è visibile lo stato della cartella dopo la scrittura e compilazione dei programmi. Nella Fig. 6 è visibile la sequenza dei comandi di compilazione e di esecuzione dell'accensione del modulo GSM, invio SMS e spe-

gnimento senza controlli, se il modulo GSM è già spento, lo si accende. Per compilare il programma GSMSpegni si utilizza il comando:

```
g++ -lrt -lpthread GSMSpegni.cpp
arduPi.o -o GSMSpegni
```

Per eseguire il programma, sempre dopo essersi posizionati nella cartella ArduPi, si utilizza il comando:

```
./GSMSpegni
```

Nel Listato 3 vediamo il programma che permette di inviare un SMS.

- Nella variabile phone_number[] si inserisce il numero di telefono da chiamare;
- AT+CMGF=1 imposta la modalità SMS in formato testo;
- AT+CMGS= invia il messaggio SMS impostato nell'istruzione successiva;

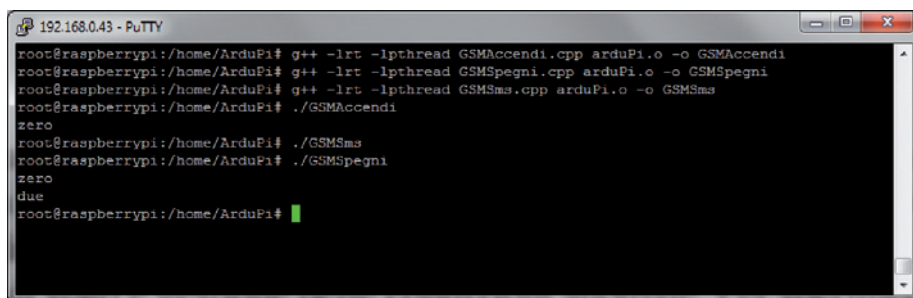


Fig. 6

gnimento del modulo.

Vediamo ora la seconda modalità, quella che permette di interagire direttamente con la porta seriale mediante il programma di emulazione TTY "CuteCom".

E siccome abbiamo impostato Raspberry Pi per essere gestita in remoto tramite protocollo SSH, continuiamo nella stessa direzione in modo da utilizzare l'emulatore TTY con le stesse modalità.

INSTALLIAMO L'EMULATORE TTY SERIALE

Listato 2

```
/*
 * GSMSpegni
 */

//Include ArduPi library
#include "arduPi.h"

int resetModulePin = 9;
int onModulePin = 8;

void switchModuleOff() {
    digitalWrite(onModulePin, HIGH);
    delay(2000);
    digitalWrite(onModulePin, LOW);
}

void resetModule() {
    digitalWrite(resetModulePin, HIGH);
    delay(500);
    digitalWrite(resetModulePin, LOW);
    delay(100);
}

int main () {
    Serial.begin(115200);
    delay(2000);
    pinMode(resetModulePin, OUTPUT);
    pinMode(onModulePin, OUTPUT);
    Serial.flush();
    printf("zero\n");
    Serial.print("AT");
    delay(1000);
    if (Serial.available() == 0)
    {
        printf("uno\n");
        resetModule();
        delay(2000);
    }
    Serial.print("AT");
    delay(1000);
    if (Serial.available() > 0)
    {
        printf("due\n");
        switchModuleOff();
    }

    return (0);
}
```

Listato 3

```

/*
 * GSMSms
 */

//Include ArduPi library
#include "arduPi.h"

int timesToSend = 1;      // Numero di SMS da inviare
int count = 0;
int ok = 0;
int numCar = 0;

char phone_number[]="3.....9"; // * Numero telefono dest. SMS

void setup(){
    Serial.begin(115200);    // UART baud rate
    delay(2000);
    Serial.println("AT+CMGF=1");    // sets the SMS mode to text
    delay(100);
}

void loop(){
    while (count < timesToSend){
        delay(1500);
        Serial.print("AT+CMGS=\"");    // send the SMS number
        Serial.print(phone_number);
        Serial.println("\"");
        while(Serial.read()!='>');
        Serial.print("Se arriva funziona"); // the SMS body
        delay(500);
        Serial.write(0x1A);    //sends ++
        Serial.write(0x0D);
        Serial.write(0x0A);
        delay(5000);
        count++;
    }
}

int main (){
    setup();
    while(1){
        loop();
        if (count == timesToSend)
        {
            break;
        }
    }
    return (0);
}

```

Per la gestione della porta seriale abbiamo scelto di utilizzare il programma "CuteCom". Lo installiamo con i soliti comandi:

```

apt-get update
apt-get upgrade
apt-get install cutecom

```

INSTALLIAMO XMING

Xming è un server X open source per Windows che permette di visualizzare sul desktop del PC Windows le finestre grafiche delle applicazioni che girano sul sistema GNU/Linux in remoto. Non viene trasferito l'intero

desktop ma unicamente la o le finestre grafiche dei singoli applicativi, che possono essere posizionate a piacere sul desktop del PC locale, assieme alle finestre del PC stesso o di altri sistemi remoti. Vi lasciamo immaginare gli aneddoti sorti al riguardo, nel mondo degli amministratori di sistema, sui comandi dati guardando la finestra di amministrazione di un

sistema e digitando in quella di un sistema diverso. Comunque la funzionalità è affascinante.

X è l'applicazione basata sul modello client/server che permette di esportare su una macchina il front-end grafico di un'applicazione in esecuzione in un sistema diverso.

Si appoggia sul protocollo SSH e sull'applicazione Putty (o Kitty) che abbiamo presentato nei numeri precedenti della rivista. La denominazione client/server spesso confonde gli utilizzatori di X perché i termini sembrano invertiti: sul desktop Windows dell'utente è in esecuzione il "server" che viene utilizzato dai programmi applicativi (client). X fornisce un servizio di visualizzazione ai programmi, in quest'ottica agisce da server, mentre il programma applicativo (che può anche essere gestito tramite connessione remota) utilizza i servizi, agendo come un client.

Il protocollo di comunicazione tra server e client opera in modo trasparente rispetto alla rete: entrambi possono risiedere sulla stessa macchina o su macchine diverse, anche con architetture e sistemi operativi differenti; il server e il client possono anche comunicare in modo sicuro attraverso la rete sfruttando un tunnel cifrato (spesso viene utilizzato SSH).

Il server X può essere:

- Un programma di sistema che controlla l'output video di un computer;
- Un componente hardware dedicato (i cosiddetti X terminal: calcolatori dotati solo dell'hardware necessario ad eseguire il server X, pensati per visual-

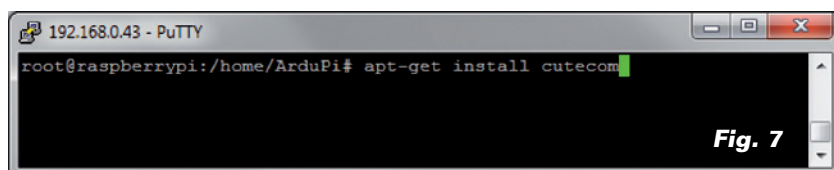


Fig. 7

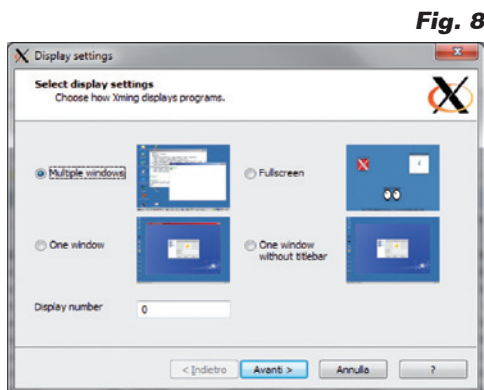


Fig. 8

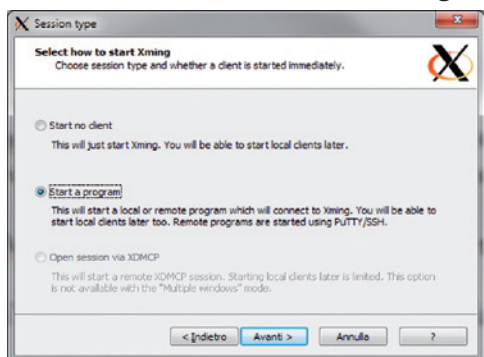


Fig. 9

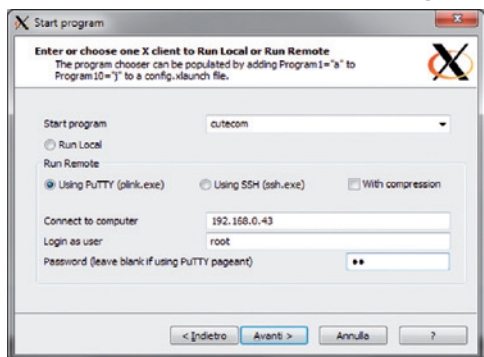


Fig. 10

lizzare applicazioni eseguite su appositi server applicativi);

- Un'applicazione che mostra dati su una finestra di un altro sistema grafico.

Xming è scaricabile da:

<http://sourceforge.net/projects/xming/>

Lo scarichiamo e installiamo come un qualsiasi applicativo Windows.

La nostra attenzione va al programma XLaunch, lo troviamo nel menu di Windows nella scelta Xming, che ci permette di lanciare in remoto quante singole applica-

zioni vogliamo in finestre separate di Windows. Una volta installato Xming lanciamo CuteCom in remoto sul desktop del nostro PC. Apriamo XLaunch (Fig. 8). Selezioniamo "Multiple Windows" e "Avanti". Nella videata successiva selezioniamo "Start a program" (Fig. 9) e ancora avanti. Ora selezioniamo "Using Putty" (Fig. 10). Nel campo Start Program digitiamo "cutecom", in "Connect to computer" inseriamo l'indirizzo IP di Raspberry Pi. In "Login as user" digitiamo "root" e nel campo password la relativa password. Ancora "Avanti" e "Avanti" anche nella pagina successiva (Fig. 11). Arriviamo all'ultima pagina dove confermiamo con "Fine" (Fig. 12). Diamo "OK" e "NO" in successione agli eventuali messaggi e ... aspettiamo. Alla fine otteniamo la finestra di Fig. 13.

GESTIAMO IL MODULO GSM DALL'EMULATORE TTY

La finestra di CuteCom è divisa in "zone" funzionali. In alto sono presenti i campi ed i pulsanti per configurare, attivare e disattivare la comunicazione seriale, in basso il campo di testo per inserire i comandi da inviare sulla porta seriale, un'area con la "history" dei comandi inviati ed alcuni comandi di configurazione che permettono di:

Fig. 11

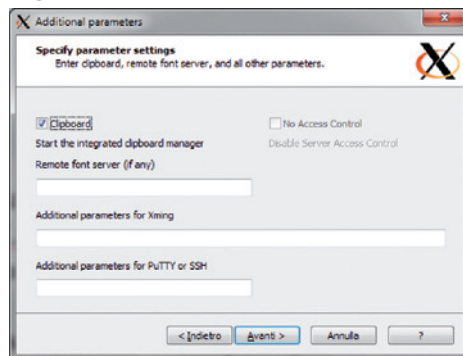
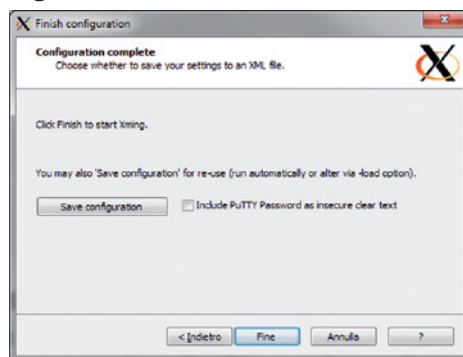


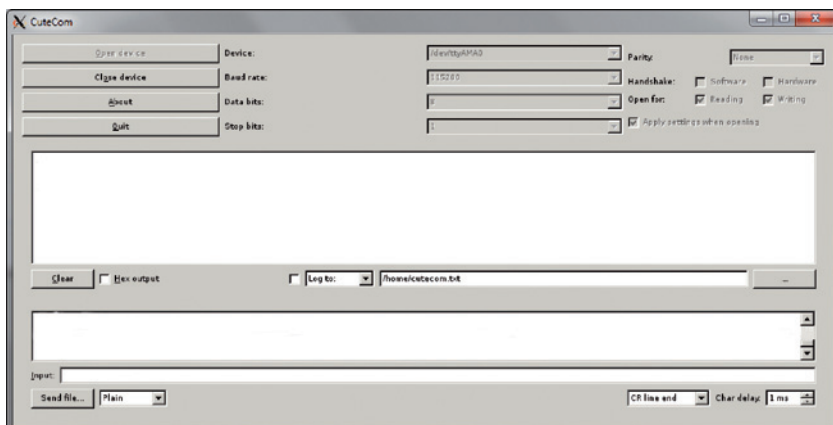
Fig. 12



- Attivare e disattivare un log;
- Aggiungere al comando caratteri di controllo come il fine linea;
- Trasmettere e/o ricevere i dati in formato ascii o esadecimale.

Infine al centro possiamo vedere l'area dove vengono evidenziati i messaggi in arrivo dalla porta seriale. La prima operazione da fare è configurare il nome di device (e relativo percorso) della porta seriale alla quale vogliamo connetterci, nel nostro caso:

Fig. 13



10^a FIERA REGIONALE DELL'ELETTRONICA SCANDIANO

(REGGIO EMILIA)

26/27 OTTOBRE

CENTRO FIERISTICO 2013



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

ORARI

Sabato 26 Ottobre dalle 9 alle 18,30
Domenica 27 Ottobre 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

/dev/tty/AMA0

Se non la troviamo nella tendina dei valori preimpostati, la scriviamo direttamente nel campo di configurazione. Configuriamo la porta impostando 115200 per la velocità, 8 per la lunghezza dei caratteri, 1 per lo "Stop bit" e "none" per la parità. Non selezioniamo alcuna opzione per l'handshake. Spuntiamo invece entrambi i check box per l'apertura in lettura e scrittura. Lasciamo senza spunta i check box "Hex output" e "Log to". Nel campo di testo vicino a "Send file" selezioniamo "Plain", nel penultimo campo in basso a destra selezioniamo "CR line end" e nell'ultimo "1 ms". Bene, possiamo iniziare. Prima però assicuriamoci che il modulo SIM9XX abbia una SIMCard inserita, sia acceso e connesso alla rete mobile verificando che il LED lampeggi lentamente. In caso contrario lo accendiamo eseguendo il programma:

./GSMAccendi

Ora possiamo premere il pulsante "Open Device" in alto a sinistra nella finestra di CuteCom. Se tutto è in ordine vedremo diventare grigia l'area di configurazione in alto e potremo iniziare ad inviare comandi inserendoli nell'apposito campo in basso. Iniziamo con "AT", lo digitiamo nel campo apposito e premiamo il pulsante "Send file", dovremmo ricevere in risposta "OK", in caso contrario potrebbe essere che il modulo non si trovi in modo comandi. Attiviamo il "Command Mode" con il comando "+++".

Facendo riferimento al manuale per il modulo che stiamo utilizzando SIM900 o SIM908, possiamo provare l'effetto di ciascun comando, sia per la configurazione del modulo che per l'esecuzione di sequenze di comandi operativi.

Qualche esempio. Colleghiamo un altoparlante autoalimentato all'uscita audio ed un microfono all'ingresso MIC oppure utilizziamo una cuffia con microfono con gli spinotti separati. Proviamo ad eseguire e a ricevere una chiamata.

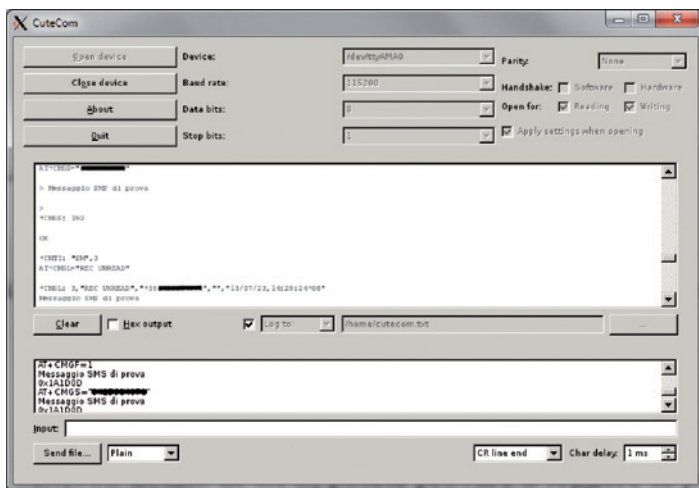
Prima diamo il comando:

AT+CALM=0

che permette di attivare la suoneria (è possibile anche impostare la musicchetta, consultate il manuale). Ora diamo il comando

ATD<numero_di_telefono> seguito da ";", ad esempio
 ATD+393.....7;.

Fig. 14



Quando il telefono chiamato accetta la chiamata potremo parlare con il nostro interlocutore, per interrompere la chiamata dal lato nostro diamo il comando:

ATH

Se un telefono esterno ci chiama, sentiremo la suoneria nell'altoparlante e ad ogni squillo vedremo la scritta "RING" nell'area centrale. Per rispondere alla chiamata digitiamo il comando:

ATA

Per inviare un SMS digitiamo la sequenza di comandi:

AT+CMGF=1

AT+CMGS="<numero_di_telefono>"

ricordando di inserire il numero di telefono tra apici. Attendiamo il messaggio di ritorno

">"

Digitiamo il testo del messaggio, per esempio:

Messaggio SMS di prova

questa volta senza apici, infine selezioniamo, nel penultimo campo in basso a destra, "Hex input" e inseriamo:

0x1A1D0D e invio

(equivale a CTRL-Z per chiudere il messaggio SMS).

Per proseguire ricordiamo di riportare il valore del penultimo campo in basso a destra al valore "CR

line end". Per leggere la lista dei messaggi SMS ricevuti e non ancora letti:

AT+CMGL="REC UNREAD"

Otteniamo (Fig. 14):

+CMGL: 3,"REC UNREAD","+393.....1","",13/07/23,14:29:24+08"

Messaggio SMS di prova

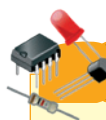
dove "3" è l'indice del messaggio. Possiamo leggere il messaggio singolo, utilizzando l'indice, anche con il comando:

AT+CMGR=3

Infine, se avete installato un modulo SIM908 dotato di GPS, potete provare a leggere la posizione (con l'antenna all'aperto) utilizzando il comando:

AT+CGPSINF=0

Otterrete come risultato una stringa composta dai campi mode, longitudine, latitudine, altezza, data/ora, TTFF (Time To First Fix), numero di satelliti, velocità e direzione di bussola. Potete sbizzarrirvi ad utilizzare i comandi nativi per prototipare le funzionalità che potrete poi codificare in programmi strutturati. Aiutatevi con gli esempi presenti negli articoli della serie "Gestire i moduli GMS con Arduino" nei numeri 175 e seguenti della rivista. Approfondiremo la realizzazione di applicazioni strutturate e modulari nei prossimi articoli. Per ora buon divertimento e ... occhio alla bolletta.



per il MATERIALE

Lo shield di espansione GSM/GPS/GPRS per Raspberry Pi è disponibile in scatola di montaggio al prezzo di Euro 13,90 (cod. FT1075K). Il kit non comprende il modulo GSM che può essere acquistato separatamente (cod. FT900M, Euro 49,00) nè l'antenna (cod. ANTGSMSTL-F01, Euro 8,00). È anche possibile utilizzare lo shield col modulo GSM/GPS Cod. FT971 (Euro 65,00) al quale vanno collegate le due antenne (ANTGPS-SMA, Euro 14,00 e ANTGSMSTL-S01, Euro 8,00) mediante due connettori Cod. CAVOUFLSMA, Euro 6,00 cad. Tutti i prezzi si intendono IVA compresa.

Il materiale va richiesto a:

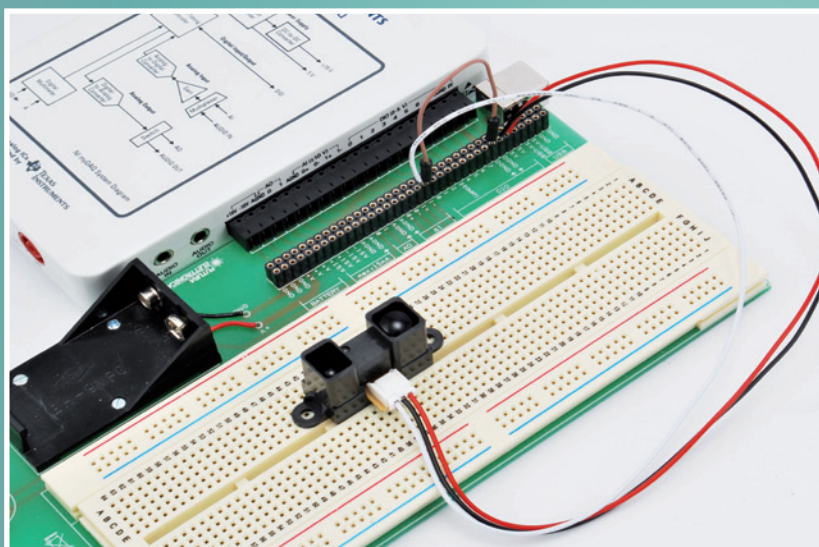
Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287 - www.futurashop.it

myKIT

MISURA DI DISTANZA CON SENSORE A INFRAROSSI

..... di DAVIDE SCULLINO

Impariamo a scrivere un programma il LabVIEW in grado di effettuare misure di distanza con un sensore ad infrarossi e mettiamo in pratica il tutto utilizzando myKIT e myDAQ.



Continuiamo la serie di esperimenti con il set myKIT realizzando questo mese un semplice misuratore di distanza.

Questa esercitazione spiega come eseguire la misura della distanza di un oggetto utilizzando un sensore a infrarossi, la scheda di acquisizione myDAQ e il software LabVIEW di National Instruments. I dati saranno acquisiti con DAQ Assistant che viene installato in LabVIEW con il driver NI DAQmx.

Utilizzando un sensore a infrarossi Sharp IR Ranger (GP2Y0A02YK), eseguire misure della distanza di un oggetto dal sensore è molto semplice. Il sensore Sharp IR Ranger (GP2Y0A02YK) è un metro a infrarossi capace di misurare la distanza di un

oggetto da circa 15 cm a 150 cm; è composto da un trasmettitore che, mediante un LED IR, emette impulsi infrarossi con lunghezza d'onda di $850 \text{ nm} \pm 70 \text{ nm}$, e da un fotodiodo IR che capta la luce riflessa dall'oggetto intercettato. DAQ Assistant viene utilizzato nel VI per acquisire i valori di tensione in uscita dal sensore. Per stimare la distanza reale dell'oggetto dal sensore saranno utilizzati due metodi: il primo utilizza un grafico di ricerca fornito dal costruttore del sensore, mentre il secondo sfrutta un'equazione di

quinto grado svolta eseguendo un'interpolazione lineare delle misure ricavate dallo stesso grafico di ricerca.

ISTRUZIONI PER IL CABLAGGIO

Il sensore utilizza 3 fili: rosso (+5V), nero (ground) e giallo o bianco (segnale di uscita). Il filo rosso deve essere collegato all'ingresso +5V del dispositivo myDAQ, il filo nero al ground AGND, e il filo del segnale all'ingresso AI1+. L'ultimo collegamento da effettuare è quello dell'ingresso AI1- al ground AGND.

PANNELLO FRONTALE LABVIEW

L'implementazione del Pannello Frontale in LabVIEW non presenta particolari difficoltà. Il pannello consiste in due grafici, due indicatori (*Linear Fit* e *Lookup Table*) e un LED verde che indica quando la tensione acquisita è nel range ammissibile per essere convertita in una distanza veritiera [20÷150 cm]. I grafici e gli indicatori

numerici mostrano la distanza rilevata, convertita dalla tensione acquisita dal sensore utilizzando i due metodi previsti.

Fig. 1 - Sensore a infrarossi Sharp IR Ranger.

CODIFICA DELLA STRATEGIA

Come si vede dal diagramma a blocchi, DAQ Assistant acquisisce il valore di tensione dalla porta AI1+, che viene eventualmente limitato utilizzando una funzione di saturazione. Successivamente si determina la distanza dell'oggetto attraverso un'operazione di ricerca nel grafico ricavato dal datasheet del sensore o sfruttando un'equazione di quinto grado ottenuta per interpolazione dei dati dello stesso grafico. Infine visualizzerete i due risultati su due grafici e due indicatori distinti, mentre su un altro indicatore potrete visualizzare la tensione acquisita.

COME FUNZIONA

Va premesso che la tensione e la distanza non hanno una corrispondenza lineare e che il grafico tensione/distanza di riferimento (**Fig. 2**) è ottenuto dal datasheet del sensore. Utilizzando un foglio di calcolo e i valori del grafico di riferimento (**Fig. 2**), sono stati tracciati questi valori e, dopo averne eseguito un'interpolazione lineare, è stata calcolata un'equazione di quinto grado che permette di convertire la tensione in distanza. Tenete presente che l'interpolazione lineare avrebbe prodotto valori anomali ai due estremi del range di misura del sensore, perciò è stato aggiunto il blocco di saturazione *Range and Coerce* prima del calcolo della distanza.

Una volta che DAQ Assistant ha acquisito il valore di tensione, questo viene visualizzato su un indicatore del Pannello Frontale e passato a un blocco di "saturazione". Questo controlla se il valore in ingresso è contenuto tra due valori di tensione ammissibili [0,37-2,8V] che corrispondono al range di misura del sensore [20÷150 cm] e se ciò si verifica si accende un LED verde sul Pannello Frontale. In caso contrario questo blocco satura il valore ricevuto in ingresso al valore massimo o minimo di soglia impostati. In seguito, il valore di tensione viene passato ad un *Formula Node* dove è presente l'equazione di quinto grado che ne compie la conversione in distanza [cm] e fornisce il risultato a un grafico e a un indicatore numerico. Lo stesso valore di tensione passato al *Formula Node* viene passato anche a un altro frammento di codice che, utilizzando i valori del grafico tensione[V] - distanza[cm] (**Fig. 2**), lo ricrea in LabVIEW per convertire la tensione all'ingresso in una distanza. Anche questo risultato sarà visualizzato in un grafico e in un indicatore sul pannello frontale.

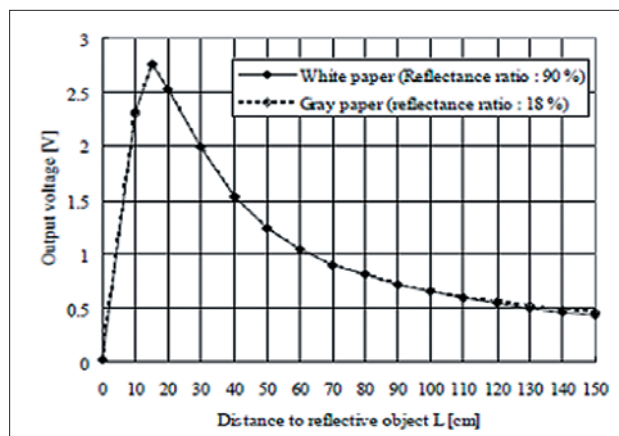


Fig. 2 - Grafico di ricerca fornito dal costruttore del sensore.

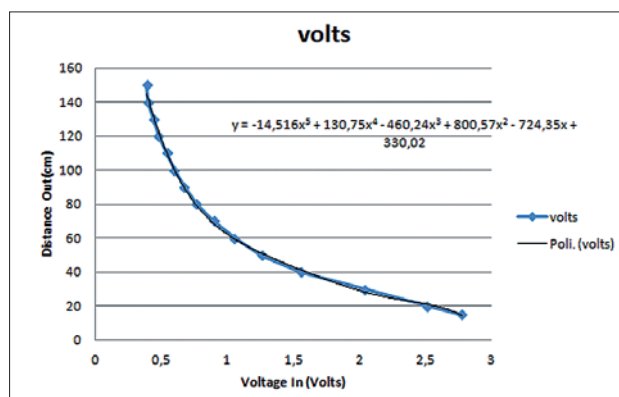


Fig. 3 - Equazione di quinto grado ottenuta per interpolazione lineare dei dati del grafico di ricerca di Fig. 2.

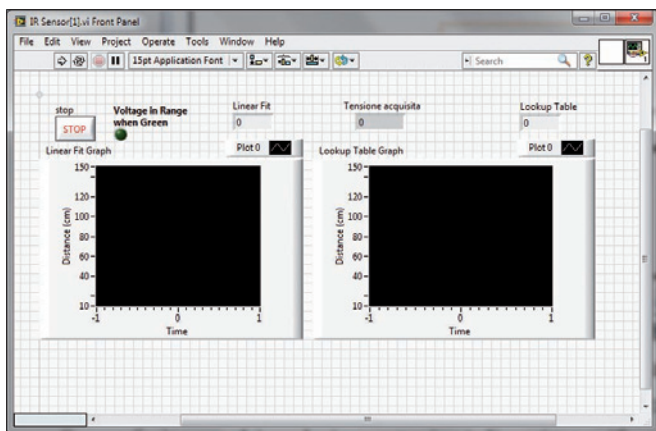


Fig. 4 - Pannello frontale LabVIEW.

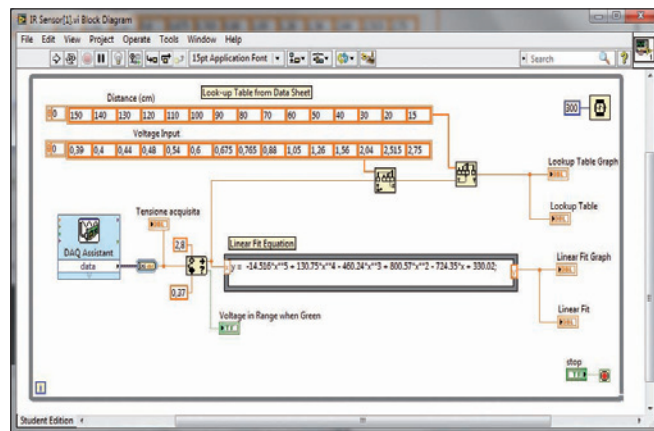


Fig. 5 - Diagramma a blocchi LabVIEW.

Tutto il codice è stato poi inserito in un ciclo *While* che permette l'acquisizione e la visualizzazione continua di dati. Questo codice continua a funzionare fino a che non viene premuto il pulsante Stop sul pannello frontale. L'esecuzione del blocco *Wait* (in alto a destra), fa in modo che l'iterazione del ciclo avvenga ogni 300 ms, pertanto la frequenza di campionamento è di circa 3 campioni al secondo, ossia di 3 Hz.

COSTRUZIONE DEL PANNELLO FRONTALE

Ora vi occuperete della costruzione del pannello frontale. Per prima cosa avviate LabVIEW, aprite un nuovo VI e andate sul Pannello Frontale. Se cliccate con il tasto destro del mouse compare la palette *Controls*, dove sono elencati tutti i tipi di controllo e di indicatori, ma anche decorazioni, tabelle, ecc. Bloccatela nella posizione meno ingombrante facendo clic sulla "puntina" in alto a destra della palette, in modo da non doverla richiamare ogni volta. Adesso dalla voce *Modern* della palette *Controls* selezionate *Boolean* e inserite nel Pannello Frontale un pulsante Stop e un Round LED. Poi selezionate *Numeric* e inserite tre indicatori numerici (*Numeric Indicator*), infine selezionate *Graph* e inserite due *Waveform Chart*. Ora, con gli strumenti *Positioning* e *Labeling* disponete e rinominate gli indicatori e il controllo Stop, come visibile in Fig. 6. Per quanto riguarda i due grafici, rinominate con lo strumento *Labeling* l'asse Y in *Distanza (cm)* e, dopo aver deselezionato *autoscale y* (tasto destro sul nome dell'asse desiderato), impostate il valore massimo (150) e minimo (10) sull'asse Y.

COSTRUZIONE DEL DIAGRAMMA A BLOCCHI

Adesso implementerete il diagramma a blocchi. Se non lo visualizzate, potete farlo selezionando *Window»Show Block Diagram* o digitando <Ctrl+E>. Gli oggetti del Pannello Frontale appaiono nel diagramma a blocchi come terminali. 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 questo simbolo . Ma è preferibile visualizzarlo così: . Ciò permette di ridurre lo spazio che occupa sul diagramma a blocchi. Per fare ciò, selezionate *Tools»Options.* e, sulla sinistra della finestra che si aprirà, impartite il comando *Block Diagram* e alla voce *General* disabilitate *Place front panel terminals as icons.* Ora, prima di iniziare ad implementare il diagramma a blocchi, osservate che 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

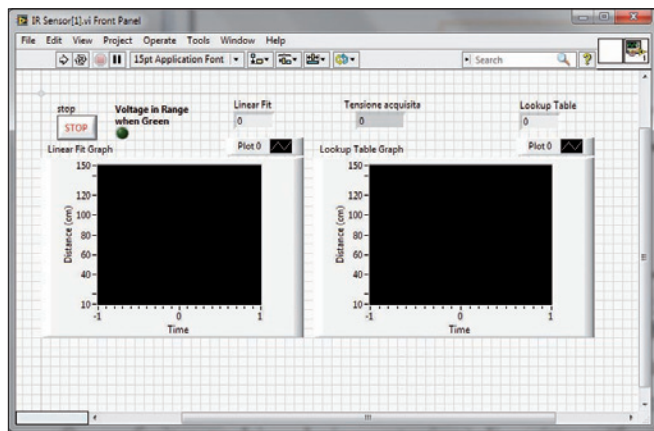


Fig. 6 - Pannello Frontale LabVIEW completato.

Tabella 1

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

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. Per prima cosa inserite e configurate il blocco di DAQ Assistant per acquisire il segnale di tensione proveniente dal sensore.

Le seguenti istruzioni vi aiuteranno passo-passo nella configurazione di DAQ Assistant.

1. Assicuratevi che myDAQ sia collegato al PC;
2. Premete Ctrl-Spazio per aprire la finestra *Quick Drop* (al primo utilizzo, sarà necessario circa un minuto);
3. Cercate la voce DAQ Assistant e fate doppio clic su di esso quando appare nella lista;
4. Trascinatelo sul diagramma a blocchi;
5. Quando appare un nuovo riquadro per le attività di configurazione rapida, selezionate:
 - a. Acquire Signals;
 - b. Analog Input;
 - c. Voltage;
 - d. Dev 1 (NI myDAQ) * Nota: Se si dispone di un altro hardware NI installato, myDAQ non sarà Dev1;
 - e. ai1;
 - f. Finish.

6. Impostate il corretto range di tensione di ingresso su *Signal Input Range*:

- a. Max: 5.3V;
- b. Min: -0.3V;

7. Cambiate *Timing Settings*:

- a. 1 Sample (On Demand);

8. Fate clic su OK.

Bene, ora procedete inserendo alla destra del blocco DAQ Assistant (**Fig. 8**) il blocco *Convert from Dynamic Data Type* situato nella palette *Functions* alla voce *Express»Signal Manipulation* e quando si aprirà la finestra per la configurazione, alla voce *Conversion* selezionate *Single Scalar* e fate clic su OK. Successivamente, inserite il blocco *In Range and Coerce* che troverete alla voce *Programming»Comparison*. Utilizzando lo strumento *Wiring*, fate clic con il pulsante destro del mouse sul "quadrato vuoto" del blocco *In Range and Coerce* (quando sarete sopra a questo ingresso il connettore lampeggia e appare il nome dell'ingresso: in questo caso *upper limit*), selezionate *Create»Constant* e inserite il valore 2,8. Ora eseguite la stessa operazione per il "quadrato pieno" (*lower limit*) e inserite il valore 0,37. I valori che avete inserito sono le soglie di tensione oltre le quali avviene la saturazione e il LED verde si spegne a segnalare che non si è più all'interno del range di misura ammissibile. In seguito, sempre con lo strumento *Wiring*, collegate l'uscita *Data* di DAQ Assistant al blocco *Convert from Dynamic Data Type* e quest'ultimo all'indicatore numerico "Tensione acquisita" e all'ingresso centrale del blocco *In Range and Coerce* (se avete difficoltà nel collegamento è consi-

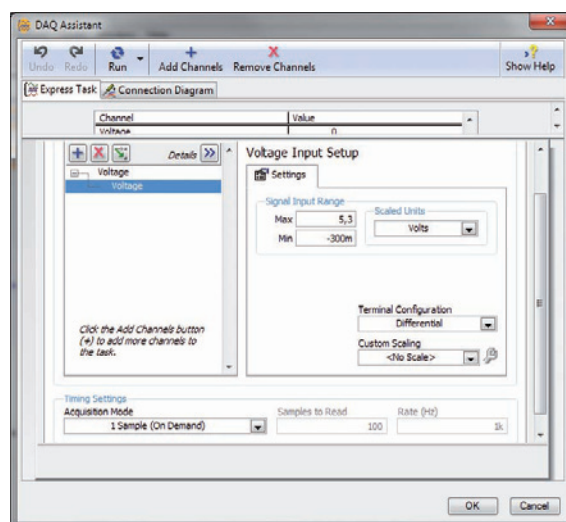


Fig. 7 - Configurazione DAQ Assistant.

gliabile aprire la finestra *Context Help*). Adesso inserite il *Formula Node* che trovate sulla palette *Functions* alla voce *Programming»Structures*. All'interno del formula node, con lo strumento *Labeling* inserite la seguente equazione:

$$y = -34.159 \cdot x^{*5} + 298.19 \cdot x^{*4} - 1004.3 \cdot x^{*3} + 1638.3 \cdot x^{*2} - 1333.7 \cdot x + 497.11;$$

Cliccando con il tasto destro del mouse sui bordi sinistro e destro del *Formula Node*, inserite un ingresso nominandolo "x" (*add input*) e un'uscita nominandola "y" (*add output*). Ora, con lo strumento *Wiring* collegate l'uscita con la freccia nera del blocco *In Range and Coerce* all'ingresso "x" del *Formula Node* e l'uscita con il punto di domanda all'indicatore LED "Voltage in Range when Green". Con il medesimo strumento, collegate l'uscita "y" del formula node al grafico "Linear Fit Graph" e all'indicatore "Linear Fit" (Fig. 8). Passerete ora all'implementazione del codice che effettua una "ricerca" nel grafico tensione\distanza del datasheet del sensore. Create prima di tutto due array di costanti con i valori presi dalla tabella del datasheet. Per fare ciò, selezionate *Array constant* nella palette *Functions* alla voce *Programming»Array*. Ora trascinate all'interno di ciascuno di essi una *DBL Numeric Constant* che trovate alla voce *Programming»Numeric*. Con lo strumento *Positioning* portatevi sul bordo laterale o sull'angolo in basso a destra dell'array ed estendetelo in modo tale da poter inserire quindici elementi. Successivamente, portate il cursore del mouse tra un elemento a caso e il successivo ed estendete un po' le caselle degli elementi dell'array per visualizzare meglio i dati che inserirete. Infine, con lo strumento *Labeling* rinominate i due array in "Distance (cm)" e "Voltage Input" e inseritevi all'interno i valori "150-140-130-120-110-100-90-80-70-60-50-40-30-20-15" e "0,39-0,4-0,44-0,48-0,54-0,6-0,675-0,765-0,88-1,05-1,26-1,56-2,04-2,515-2,75" rispettivamente (Fig. 8). Adesso che avete completato gli array di costanti, nella palette *Functions*, alla voce *Programming»Array*, trascinate sul diagramma a blocchi un blocco *Interpolate 1D Array* e un blocco *Threshold 1D Array*. Ora con lo strumento *Wiring* collegate il tutto come in Fig. 8, anche il grafico "Lookup Table Graph" e l'indicatore numerico "Lookup Table". Per concludere selezionate *While Loop* nella palette *Functions* la voce *Programming»Structures* e inglobate tutto il codice inserito finora all'interno del ciclo while. Collegate con lo strumento *Wiring* il terminale del controllo Stop al *Conditional Terminal* del ciclo *While* nell'angolo in basso a destra. Come ultima cosa, inserite il blocco *Wait(ms)* situato nella palette *Functions* alla

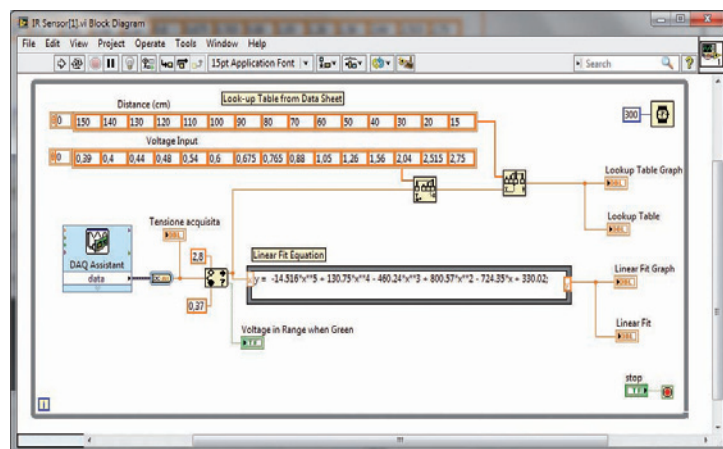


Fig. 8 - Diagramma a blocchi LabVIEW.

voce *Programming»Timing* e collegatelo a una costante con valore 300: fate ciò cliccando col tasto destro del mouse sul terminale d'ingresso del blocco *Wait(ms)*, selezionando *Create»Constant*, e inserendovi il valore 300. Questo per fare in modo che il ciclo *While* vada in esecuzione ogni 300 ms, così che DAQ Assistant acquisisca circa tre campioni al secondo.

I commenti che appaiono nei riquadri potete inserirli a vostro piacimento utilizzando lo strumento *Labeling*, in quanto non interferiscono con il codice grafico ma ne aumentano la comprensione. Se preferite cambiare le impostazioni di scrittura (font, dimensione, colore, ecc.), sia nel *Pannello Frontale* sia 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 utilizzando 15pt Application Font). Quando avrete finito di costruire il vostro VI, andate su *File»Save as* e scegliete un nome da assegnare e una destinazione per salvare il vostro VI. Ora cliccate sul pulsante *Run* nella barra degli strumenti per met-

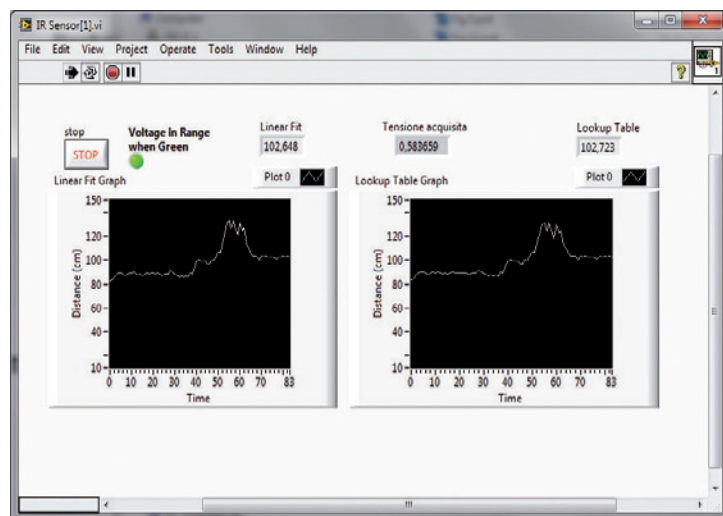


Fig. 9 - VI in funzione.

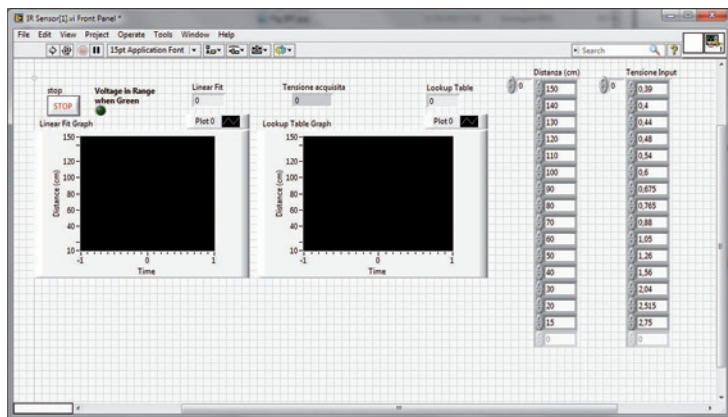


Fig. 10 - Pannello Frontale modificato per effettuare la calibrazione del sensore a infrarossi.

tere il VI in esecuzione. 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 se un VI non va in esecuzione, vuol dire che degli errori non lo rendono eseguibile. Se 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*, dove 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.

CALIBRAZIONE DEL SENSORE

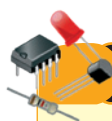
Il grafico di ricerca fornito dal costruttore (Fig. 2) è stato realizzato in condizioni ottimali difficilmente riproducibili, quali ad esempio l'ambiente di misura, la perfetta inclinazione del sensore a infrarossi rispetto all'oggetto e l'indice di riflessione di quest'ultimo. Può quindi esservi utile eseguire una calibrazione in campo del sensore, riscrivendo la tabella di riferimento utilizzata nel software per renderla più consona al vostro ambito di utilizzo. Per fare questo è consigliabile visualizzare e utilizzare come controlli sul Pannello Frontale i due array utilizzati per costruire la tabella di riferimento nel diagramma a blocchi. Andate quindi sul diagramma a blocchi, cliccate col tasto destro del mouse in corrispondenza di uno dei due array e nel menu a tendina che comparirà selezionate *Change to Control*. Ripetete la stessa operazione per l'altro array. Successivamente

Fig. 12 - Calibrazione del sensore a infrarossi.



andate sul pannello frontale e, con lo strumento *Positioning*, sistemate i due controlli array uno accanto all'altro ed estendeteli per visualizzare i quindici elementi che contengono. Il vostro Pannello Frontale ora si presenterà circa come in Fig. 10.

Mandate in esecuzione il VI e posizionate l'oggetto a una distanza conosciuta rispetto al sensore. Ora dall'indicatore della tensione acquisita ricavate il valore corrispondente alla distanza (ponete l'oggetto il più possibile parallelo al sensore e cercate una posizione in cui il valore di tensione acquisito sia il più possibile stabile). Adesso, anche mentre il VI è in esecuzione, con lo strumento *Labeling* inserite il valore di tensione rilevato nella casella dell'array *Voltage Input* corrispondente alla distanza in cui si trova l'oggetto e premete *Invio* (in alternativa fate clic sul pulsante che comparirà in alto a sinistra appena inserirete in un controllo un nuovo valore). Ripetete quest'operazione per ogni distanza indicata nell'array *Distance* (cm). Una volta completata quest'operazione, potete inserire i nuovi valori di tensione anche sul foglio di calcolo chiamato *Sharp Renger* allegato a questa esercitazione. Ciò vi permetterà di ottenere una nuova equazione di quinto grado, ottenuta per interpolazione, da inserire nel *Formula Node* del diagramma a blocchi. Adesso il VI da voi creato potrà funzionare al meglio nel vostro ambito di utilizzo. ■



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



Maker Faire

il festival del DIY
sbarca a **ROMA**

di ARSENIO SPADONI

Tutti a Roma per la prima edizione della Maker Faire che si svolgerà dal 3 al 6 ottobre presso il Palazzo dei Congressi. La giornata di giovedì 3 ottobre è dedicata ad una serie di conferenze ed incontri per capire e spiegare il movimento dei "Makers"; venerdì 4 è interamente dedicata alle scuole e agli studenti mentre durante il week-end del 5 e 6 ottobre chiunque potrà venire a scoprire gli innovativi progetti dei 200 makers provenienti da tutta Europa.

“Il far da sé in versione open source, con la condivisione delle informazioni e delle esperienze tramite Internet, è una tendenza che sta coinvolgendo un numero crescente di americani che si ritrovano ogni anno a San Mateo, in California, per presentare le proprie realizzazioni durante un fine settimana che non è solo un festival

dell'innovazione e della tecnologia, ma anche un'occasione per confrontarsi con persone con gli stessi interessi e passioni.” Così scrivevo più di tre anni fa dopo una visita alla Maker Faire di San Mateo, in California, dove questa manifestazione è nata. Evidentemente questa tendenza non era solo americana tanto che, proprio da



quella edizione, le Maker Faire si sono moltiplicate, prima in altre città degli Stati Uniti e poi in tutto il mondo, con quello stesso format e, soprattutto, con quello spirito.

Quest'anno, finalmente, è la volta dell'Italia, che ospiterà l'edizione Europea della Maker Faire dal 3 al 6 Ottobre a Roma. Oltre duecento makers europei che hanno risposto alla "Call for Makers" e che sono stati selezionati dagli organizzatori esporranno le loro creazioni tecnologiche, frutto di ingegno e passione ma anche di tante notti insonni. Prototipi quasi sempre nati dalla condivisione di idee ed esperienze con altri makers e che spesso vengono prodotti artigianalmente in pochi pezzi per altri appassionati che ne percepiscono l'utilità. Progetti che qualche volta, quando l'idea è giusta, si possono trasformare in business, in imprese che creano lavoro e ricchezza. Come nel caso di Bre Pettis che proprio in quella edizione della Maker Faire di tre anni fa presentò la sua prima stampante 3D dando poi vita alla MakerBot Industries, società che quest'anno è stata acquistata per 600 milioni di dollari dalla

Stratasys, una delle due multinazionali del settore interessata, evidentemente, ad entrare nel mercato delle stampanti di fascia più bassa.

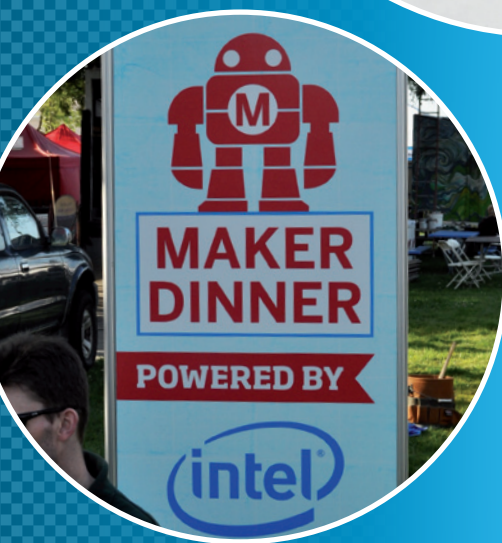
Questi nuovi artigiani tecnologici hanno spesso bisogno di luoghi dove trovare, oltre ad amici con cui condividere le esperienze, anche le attrezzature necessarie per mettere in pratica le proprie idee, macchine per il taglio laser, stampanti 3D ed altro ancora. Ecco quindi nascere i FabLab, spazi digitali da condividere con altri makers e che, alla Maker Faire di Roma, saranno anch'essi ben rappresentati.

E siccome l'attività dei makers attira l'attenzione di sempre più persone, ecco diffondersi anche i workshop e i laboratori dove i più esperti trasferiscono conoscenze ed esperienze ai ragazzi alle prime armi.

È il "fare" che in alcuni casi diventa business: non a caso la Maker Faire di Roma sarà anche la tappa conclusiva del Barcamper Makers Tour che ha attra-



Maker Faire
THE EUROPEAN EDITION



versato l'Europa e l'Italia alla ricerca di idee imprenditoriali e innovative nel settore dei Makers. Sul palco della Maker Faire i migliori 10 progetti d'impresa selezionati presenteranno la propria idea di fronte ad esperti, imprenditori, venture capitalist, giornalisti di settore, blogger, rappresentanti istituzionali e docenti universitari.

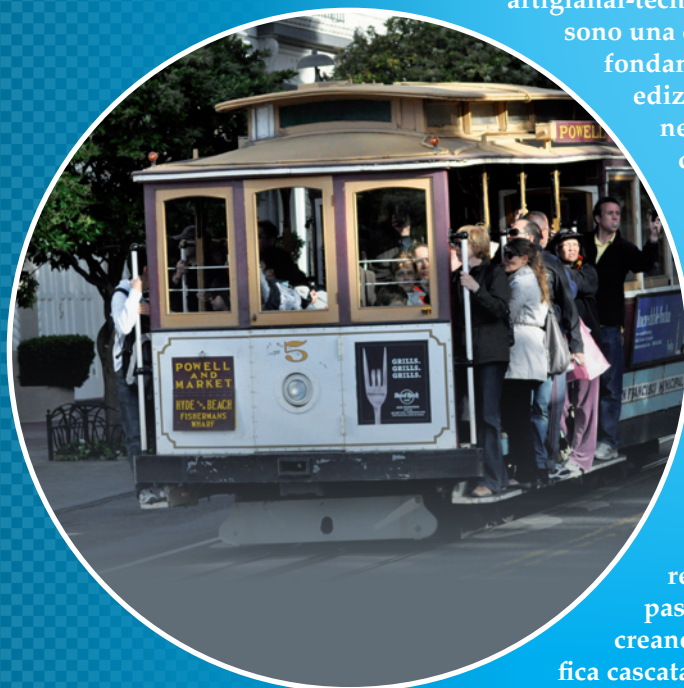
Certo, della Maker Faire originale mancheranno le attrazioni artigianal-tecnologiche che sono una componente fondamentale delle edizioni americane, lo spettacolo della tecnologia declinata nelle forme più strane e sorprendenti. Come l'incredibile concerto di Eepy-Bird dove migliaia di bottiglie di Diet Coke, reagendo con le pasticche Mentos, creano una coreografica cascata di Coca-Cola.

Oppure le scariche di centinaia di migliaia di volt prodotte da enormi generatori Tesla e modulate a ritmo di musica che colpiscono gabbie metalliche dentro le quali prendono posto coraggiosi volontari.

Per non parlare degli spettacoli di fuoco della "Fire Art", con enormi mostri sputafuoco di metallo realizzati con l'unico scopo di stupire e di gareggiare con altri appassionati di questa strana (per noi europei) forma d'arte. Ciascuno di questi mostri utilizza qualche decina di bombole di gas fissate lungo la struttura: una cosa assolutamente improponibile per i nostri modelli di sicurezza.

Mancheranno anche le pentole da 1.000 porzioni di paella di "Gerard's Paella" ma, almeno da questo punto di vista, le alternative saranno sicuramente all'altezza.

Questa prima edizione della Maker Faire si articola in quattro giornate, dal 3 al 6 Ottobre. Giovedì 3 Ottobre, dalle 10:00 alle 19:00 si susseguiranno una serie di conferenze che cercheranno di dare risposta ad una domanda apparentemente banale, "How to reMake the World": chi sono i Makes e dove è





nato il movimento che ha dato origine alla Nuova Rivoluzione Industriale? Potrete ascoltare le opinioni di Riccardo Luna, curatore di Maker Faire Rome, e di altri ospiti internazionali come David John Gauntlett, Raffello D'Andrea, Dale Dougherty e molti altri ancora. Sarà un'immersione totale nell'universo dell'elettronica open source e della stampa 3D.

La giornata di venerdì 4 Ottobre sarà interamente dedicata agli istituti scolastici e agli studenti: saranno presentati nuovi progetti destinati al mondo della formazione e proposte per rendere più divertente e innovativo l'apprendimento mediato dalle nuove tecnologie.

In particolare si parlerà di Robotica Educativa e di come i corsi e le gare di robotica siano uno strumento importante per l'apprendimento multidisciplinare nonché un valido stimolo per la creatività dei ragazzi. Infine Sabato 5 e Domenica 6 la Maker Faire aprirà le porte a tutti, appassionati di fai-da-te o semplici curiosi, adulti e bambini.

Un'occasione davvero unica per avvicinarsi al mondo dei creativi tecnologici e vedere in

funzione i loro prototipi.

Probabilmente ci sarà anche un'area commerciale, il cosiddetto "Maker Shed" dove sarà possibile acquistare prodotti tecnologici, stampanti 3D, libri e gadget di varie aziende del settore.

Noi di ElettronicaIn/Futura Elettronica saremo presenti con uno stand dove potrete acquistare tutti i nostri prodotti editoriali, kit di montaggio e prodotti tecnologici vari.

Vi aspettiamo numerosi! ■

5-6 OTTOBRE
ROMA, PALAZZO
DEI CONGRESSI



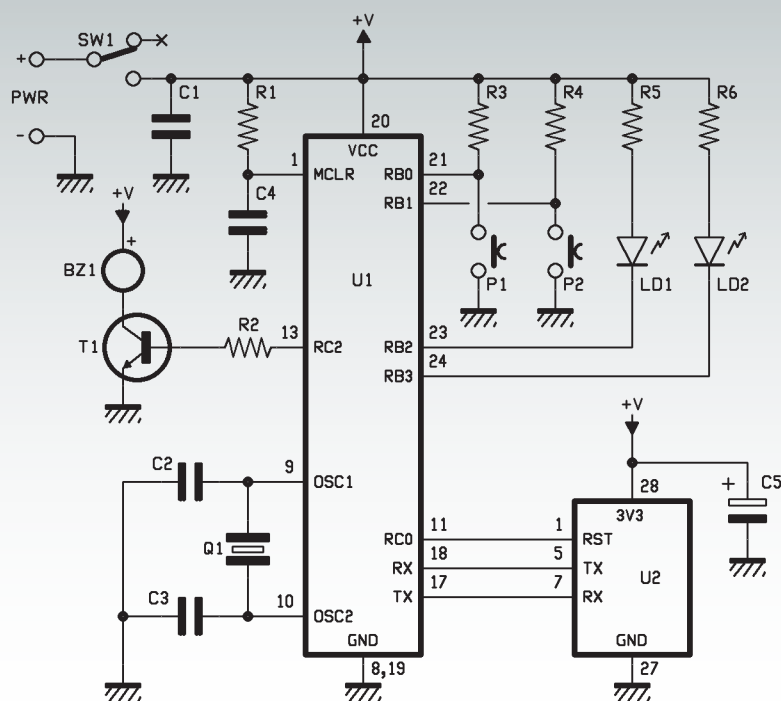
BLUETOOTH ANTI THEFT



degli Ingg.
TOMMASO
E ALESSANDRO
GIUSTO

Sistema antifurto con tecnologia bluetooth che usa un'applicazione su smartphone Android per rilevare la distanza tra noi e un oggetto di valore che vogliamo tenere sotto controllo e, nel caso in cui la distanza superi una certa soglia, segnala l'allarme tramite cicalino sonoro e vibrazione del cellulare.

È ancora tempo di vacanze e chi avrà la fortuna di poterci andare potrà fare le valigie e prepararsi alla partenza. In tema di valigie, da recenti statistiche sembra che ogni anno negli aeroporti, stazioni ferroviarie, metropolitane, ecc... (insomma: nei luoghi di partenza pubblici) per mano dei "soliti ignoti" ne spariscono diverse. In questi casi i malviventi sono facilitati nel loro compito oltre che dalla condizione di confusione tipica di questi ambienti in quei giorni, anche dal fatto che prima che il legittimo proprietario (impegnato nei vari check-in, registrazioni, code alle biglietterie, ecc...) si accorga del furto tipicamente passa diverso tempo; i ladri hanno quindi tutto il tempo necessario per "sparire dalla circolazione", aprire con calma il frutto del loro operato e appropriarsi di tutto ciò di valore che trovano. Da tutte queste considerazioni è nata l'idea del progetto che presentiamo in queste pagine. Si tratta di un dispositivo che, tramite smartphone Android, permette di determinare e monitorare la distanza tra noi e un oggetto che vogliamo controllare; nel caso in cui questa distanza



diventi eccessiva o superi determinate condizioni, ci avvisa sullo smartphone tramite messaggio a display e vibrazione. Inoltre, anche il dispositivo emetterà un allarme sonoro con un duplice effetto: "spaventare" e scoraggiare i malviventi (è noto che in condizioni di panico si commettono più errori rispetto alla calma) e attirare l'attenzione sia nostra, sia delle persone intorno a noi (non si sa mai che qualche persona di buona volontà intervenga in nostro aiuto).

Il nostro esempio è il primo caso di utilizzo reale del sistema che ci è venuto in mente ed è la tipica condizione per cui il tutto è stato progettato; nulla però vieta di adattarlo anche ad altri scopi. Abbinato a un animale domestico potrebbe permetterci di tenerlo sotto controllo più facilmente al parco, senza la necessità di doverlo seguire fisicamente per tutto il tempo; nei cantieri edili dove lavorano differenti gruppi di operai e dove si potrebbe controllare con meno stress un attrezzo di valore, e in tantissimi altri casi. La tecnologia utilizzata per

tenere sotto controllo la distanza è il Bluetooth. Abbiamo realizzato una scheda elettronica (funzionante con batteria ricaricabile) che andrà abbinata all'oggetto che vogliamo controllare (meglio se nascosto e/o inserito nell'oggetto stesso); tramite apposito software Android collegheremo il nostro cellulare al dispositivo, attiveremo l'allarme e potremo tranquillamente allontanarci. Utilizzando una connessione Bluetooth il software e la periferica ciclicamente si scambieranno dei dati come verifica della connessione; in caso che questa comunicazione fallisca entrambe le unità identificheranno che l'altra periferica è stata rimossa dal campo di copertura e pertanto entrambe attiveranno l'allarme. Il software consente di attivare e disattivare l'allarme; in questo modo se dobbiamo allontanarci per un determinato periodo lasciando l'oggetto da controllare a persone fidate, potremo disattivare il sistema evitando inutili e fastidiosi allarmi; una volta ritornati riporteremo il sistema nelle condizioni di normale funzio-

namento. È prevista inoltre una pagina di configurazione in cui è possibile impostare alla periferica il nome Bluetooth (in modo da identificarlo più facilmente nel caso in cui ci si trovi in una zona caratterizzata dalla presenza di diversi dispositivi con Bluetooth attivo), il codice PIN di accesso e il livello di potenza di trasmissione radio (per poter scegliere diversi raggi di copertura e/o allarme dell'antifurto ed adattare il funzionamento agli spazi aperti o chiusi).

Chiaramente tutte queste impostazioni sono settabili senza dover accedere fisicamente all'oggetto, grazie al link Bluetooth. La sicurezza di tutto il sistema è garantita intrinsecamente dal Bluetooth: infatti l'accesso alla periferica sarà possibile solo se si conosce il relativo Bluetooth PIN code; inoltre la comunicazione sarà cifrata proprio dai livelli più bassi del protocollo Bluetooth stesso.

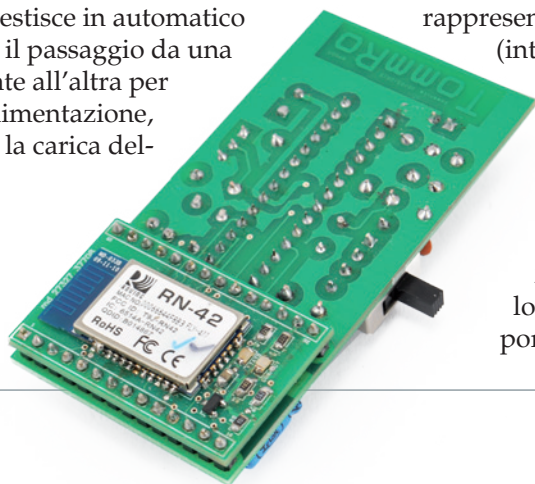
HARDWARE SCHEDA ANTI THEFT

In questa stessa pagina riportiamo lo schema elettrico del progetto; come si vede l'hardware del dispositivo è basato principalmente su un microcontrollore PIC18F2520 e sul modulo Bluetooth RN-42 (analizzato ampiamente in precedenti numeri della rivista). Le due parti sono interconnesse tramite collegamento seriale; attraverso la porta UART il microcontrollore colloquia, programma e gestisce il modulo radio.

Tutto il sistema è alimentato tramite una batteria ricaricabile agli ioni di litio con tensione nominale di circa 3,7volt e capacità di circa 1000 mAh. Come caricabatteria abbiamo scelto un hardware già pronto, affidabile e collaudato (disponibile in Futura Elettronica ed identificato dal

Protocollo di comunicazione

codice FT864M). Questo sistema dispone come input di una porta in formato mini USB da utilizzare per collegare il caricabatteria (grazie ai diffusi smartphone questo formato è ormai diventato comune e presente in molti alimentatori per cellulari; oppure la tensione di 5 VDC necessaria per la ricarica può essere prelevata dalle porte USB dei PC) e di due porte di output: la prima composta fisicamente da 2 piazzole passo 2,54mm è da utilizzare come porta di alimentazione per la nostra scheda elettronica (il formato ne rende semplice l'integrazione nei PCB; basta predisporre in fase di progettazione un pin strip 2 poli maschio); l'altra è da utilizzare per collegare la batteria ricaricabile (questa in realtà ha funzionalità sia di output che di input, in funzione del fatto che sia collegato o meno il caricabatteria). Come già detto, la tensione d'ingresso deve essere di 5 VDC (standard USB) mentre quelle di uscita sono entrambe a 3,7 volt. Il fatto di utilizzare un kit completo di questo tipo facilita notevolmente l'integrazione nei propri progetti di un sistema di carica batteria; anche per tutti i vostri progetti, ci sembra inutile "buttarsi" nel progettare una specifica sezione avendo già un qualcosa di pronto e funzionante. Inoltre, il modulo è in grado di rilevare automaticamente se è collegata un'alimentazione alla porta USB e/o una batteria, e gestisce in automatico sia il passaggio da una fonte all'altra per l'alimentazione, sia la carica del-



la batteria quando è presente una tensione alla porta USB. Durante la ricarica il LED verde del modulo rimane acceso fisso; quando la batteria è completamente carica – cioè quando la tensione ai capi della batteria raggiunge i 4,2 VDC - il LED si spegne. Analizzato il caricabatterie, torniamo allo schema elettrico della scheda dove è presente un interruttore per l'accensione/spegnimento della scheda (come si vede dallo schema collega o rimuove il polo positivo della tensione di alimentazione); è anche presente il condensatore C1 per fornire un minimo di energia in più in caso di brevi picchi nell'assorbimento di corrente del sistema generale. Come anticipato, il cervello è rappresentato dal PIC18F2520

(integrato U1) funzionante a 4MHz (quarzo Q1) collegato al modulo Bluetooth fisicamente tramite l'apposito doppio socket a passo 2,54 mm da 28 pin e logicamente tramite la porta UART (pin 17 e 18

Il protocollo di comunicazione tra scheda e smartphone è abbastanza semplice. Come già detto nei precedenti articoli riguardanti il modulo Bluetooth RN-42, da specifiche sono previste 2 distinte classi di dispositivi: master e slave. Della prima classe fanno parte tutti quei dispositivi che, di propria iniziativa, possono avviare una nuova connessione mentre della seconda classe fanno parte quelli che sono sempre in attesa di richieste di connessioni; facendo un paragone con la comunicazione TCP/IP, i master sono equivalenti ai client mentre gli slave ai server.

Nel nostro specifico progetto lo smartphone si comporta da master mentre la scheda elettronica da slave.

Il protocollo prevede delle trame così strutturate:

- primo byte: tipo di operazione (lettura o impostazione stato scheda); GETINFO_CMD (byte 0x41) per lettura; SETINFO_CMD (byte 0x42) per impostazione;
- secondo byte: identificativo tipo della risorsa. Sono previste 4 possibilità: stato attivazione allarme e attuale livello della potenza Bluetooth di trasmissione (STWRNG_CMD; byte 0x45); nome Bluetooth (NMEBTH_CMD; byte 0x46); codice PIN Bluetooth (PNCBTH_CMD; byte 0x47) e infine potenza trasmissione Bluetooth (PTRXTH_CMD; byte 0x48 usato solo in scrittura in quanto per la lettura è già compreso nel comando STWRNG_CMD).
- quarto byte: stato di attivazione allarme o livello potenza trasmissione.

Il protocollo prevede inoltre che, ad ogni richiesta di comandi proveniente dallo smartphone master, la scheda elettronica (slave) risponda con pacchetti di conferma.

del PIC che rispettivamente finiscono sui pin 7 e 5 del socket) e il pin di reset (pin 11 di U1) usato dal MCU in fase di accensione per portare l'RN-42 alle condizioni di default.

Al socket per il Bluetooth è collegato il condensatore C5 necessario per supplire ai picchi di assorbimento, ma solo del modulo RN-42 (tipicamente nelle fasi in cui la sezione radio sia attiva, cioè in trasmissione e ricezione). A differenza di C1 che è per il sistema generale, C5 è invece dedicato al solo Bluetooth e anche in fase di progettazione del PCB (come indicato nel datasheet dell'RN-42) è stato posizionato il più vicino possibile al socket. Tornando all'analisi dello schema elettrico, è presente un buzzer (BZ1) per generare l'allarme sonoro. Il componente è pilotato tramite il transistor NPN BC547 (T1) che collega o meno il polo negativo di BZ1 a massa. In fase di progettazione abbiamo fatto fatica a trovare buzzer funzionanti a 3,6 volt; in particolare ne abbiamo trovati solo del tipo senza circuito di pilotaggio

Interfaccia utente-scheda

La scheda dispone di 2 LED (verde e rosso) utilizzati per dare un'indicazione visiva del proprio stato e di 2 pulsanti con cui l'utilizzatore può interagire.

Il LED verde è utilizzato per segnalare se è attiva una connessione Bluetooth e se il software è in esecuzione. Durante il normale funzionamento, a collegamento down, il LED lampeggia con durata di circa 1 secondo; invece per tutto il tempo in cui il link è attivo il LED rimane acceso fisso.

Il LED rosso è invece usato per segnalare lo stato di attivazione dell'allarme. Allarme off corrisponde a LED rosso spento; allarme attivo a LED rosso acceso fisso.

Il pulsante P1 è usato per un breve test sonoro del cicalino: se l'allarme è off e non è attiva una connessione Bluetooth, premendo P1 viene attivato il buzzer per circa 1 secondo.

P2 è invece utilizzato per passare da un livello di potenza di trasmissione al successivo. I possibili livelli selezionabili sono 7 (da 1 a 7) in cui 1 significa livello di potenza minimo. Passando da un livello al successivo la scheda segnala il livello corrente tramite lampeggi dei LED e cicalino in numero pari al livello selezionato. Inoltre da smartphone Android, nella pagina di configurazione, è possibile verificare visivamente il valore attivo.

interno. In pratica, per attivare il suono, non è sufficiente alimentare il cicalino, ma bisogna farlo tramite onda PWM di determinati frequenza e duty cycle.

Per questo motivo la base del transistor è collegata al pin 13 del PIC, cioè a una delle due uscite (in particolare CCP1 – modulo Compare Capture Pulse 1) predisposte nel MCU per generare onde PWM. In particolare per il buzzer da noi scelto (Part Number: 254-EMB93-RO prodotto dall'azienda Kobitone) l'onda PWM da generare dovrà avere frequenza di circa $2731\text{Hz} \pm 200\text{Hz}$ (bisognerà tenerlo presente nella fase di scrittura del firmware del PIC).

Per finire sono presenti 2 LED (LD1 di colore verde e LD2 di colore rosso, con rispettive resistenze di polarizzazione R5 ed R6) la cui gestione avviene da firmware tramite i pin 23 e 24 del PIC. In particolare facciamo notare che, essendo abbastanza bassa la corrente che passa attraverso il LED, non abbiamo predisposto uno stadio di accensione tramite transistor (come invece abbiamo fatto per il buzzer che tipicamente consuma di più rischiando di "bruciare" la porta del PIC), ma abbiamo comandato il polo negativo dei

LED direttamente tramite le uscite della MCU.

Il LED verde segnala lo stato di connessione Bluetooth: rimane acceso fisso nel periodo temporale in cui la connessione è attiva; per segnalare che il firmware del PIC è in esecuzione, il LED lampeggia con frequenza di circa 0,5Hz quando la connessione è OFF.

Il LED rosso segnala invece lo stato di attivazione dell'allarme: se il sistema è abilitato (allarme attivo) il LED rosso è acceso; viceversa se il controllo dell'allarme è disattivo il LED rosso è spento.

L'ultima nota riguarda la presenza dei due pulsanti P1 e P2 (collegati ai pin 21 e 22 di U1) la cui funzionalità viene definita/gestita da firmware: quando il controllo dell'allarme è disattivo e nessuna connessione Bluetooth è in corso, premendo P1 è possibile eseguire un breve test del buzzer sonoro; invece (sempre nelle stesse condizioni di attivazione allarme e connessione) premendo P2 è possibile incrementare di un'unità (tenendo conto che raggiunto il livello massimo si torna al minimo) il livello della potenza di trasmissione (tramite diversi lampeggi dei LED e note del cicalino viene segnalato il livello selezionato).

FIRMWARE SCHEDA ANTI THEFT

Avendo già ampiamente analizzato e descritto il modulo Bluetooth RN-42 (sia come caratteristiche hardware, sia come protocollo UART di comunicazione), capire quali sono le caratteristiche e funzionalità implementate nel firmware della scheda elettronica risulta abbastanza intuitivo.

Tutte le impostazioni di configurazione della scheda (nome Bluetooth, codice PIN, livello della potenza di trasmissione radio e infine stato di attivazione dell'allarme) vengono memorizzate nella EEPROM del MCU in modo da non perderle tra un'accensione e la successiva.

Il firmware è stato scritto per PIC18F2520 utilizzando il linguaggio C. All'avvio vengono inizializzate le risorse hardware utilizzate (la porta UART di comunicazione di cui viene usato il relativo interrupt in ricezione per memorizzare i caratteri in ingresso senza appesantire troppo il firmware; i pin usati come IN/OUT digitali; il timer 1 il cui interrupt viene usato per generare un orologio software per determinare il passare del tempo per diversi timeout ed infine il timer 2 usato per generare il PWM del cicalino sonoro da hardware). In seguito viene inizializzato il modulo Bluetooth RN-42: viene selezionata la modalità di funzionamento slave, impostati alcuni timer di configurazione, la potenza di trasmissione, il nome, il PIN e la potenza di trasmissione Bluetooth (questi ultimi 3 leggendo da EEPROM l'ultimo valore memorizzato) ed infine viene richiesto il reboot del modulo in modo da rendere effettive le impostazioni (questa operazione verrà eseguita anche in seguito, ogni qualvolta dal software Android verrà richiesta la modifica di alcuni parametri). Successivamente il software suddivide l'esecuzione nel caso che sia attiva o meno una comunicazione Blue-

tooth; la condizione viene rilevata grazie ad opportuni comandi del protocollo UART.

Nella situazione in cui non sia presente alcuna connessione attiva vengono continuamente testate le seguenti condizioni:

- la richiesta di un nuovo collegamento per identificare che comunque lo smartphone Android sia in vicinanza e/o ci siano dei comandi in ingresso;
- viene verificato che l'allarme sia attivo; in questo caso se sono passati alcuni secondi con connessione continuamente OFF significa che la distanza tra scheda e smartphone è diventata superiore alla zona di copertura e pertanto è il momento di attivare l'allarme;
- se l'allarme è disattivo viene verificata la pressione di un pulsante. In caso affermativo per

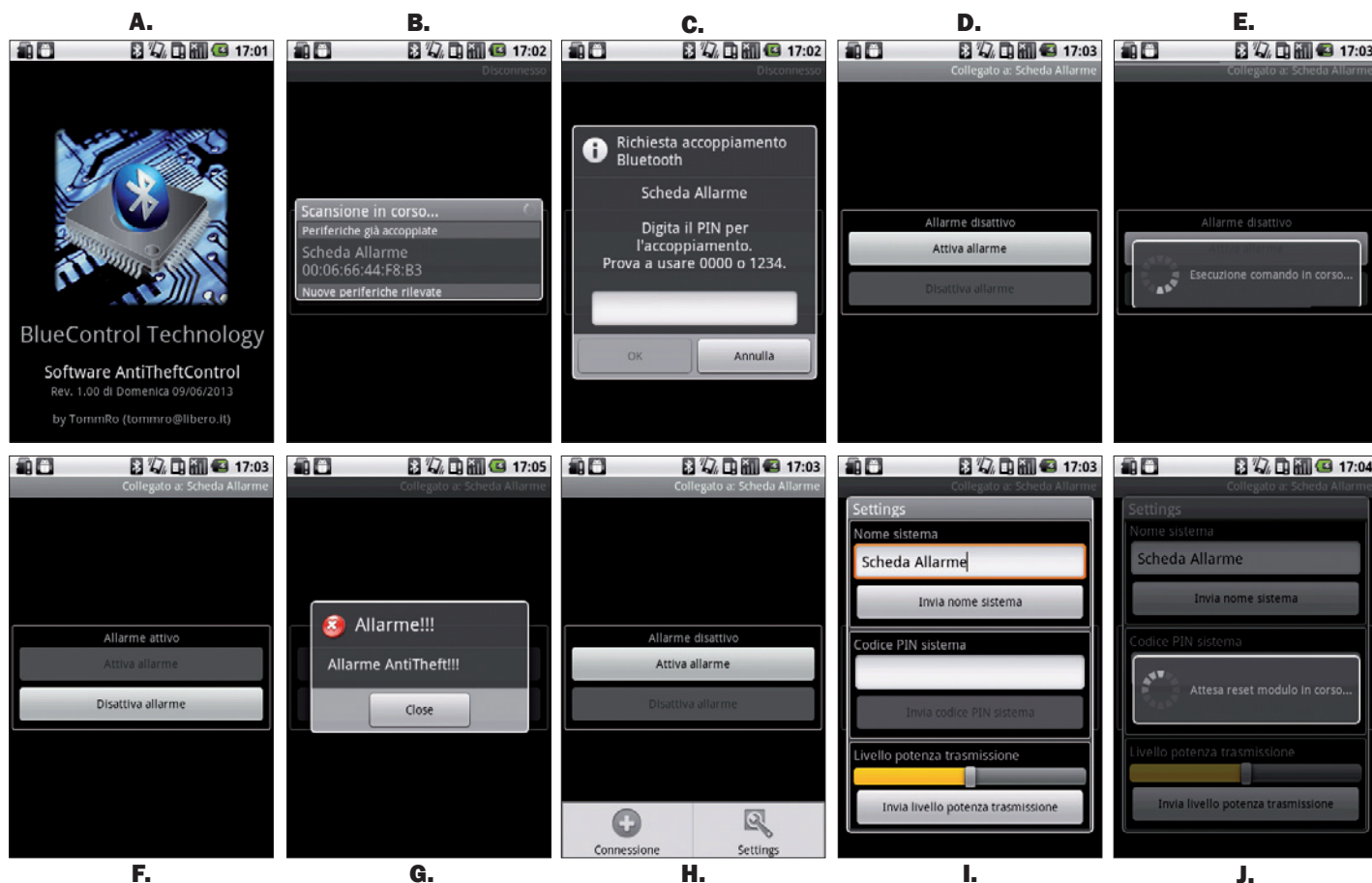
P1 (come detto in precedenza) viene effettuato un breve test del cicalino sonoro; per P2, invece, viene incrementato in modo "circolare" il livello della potenza di trasmissione (il livello selezionato viene segnalato tramite diversi lampeggi dei LED e buzzer: 1 lampeggio significa livello minimo, 7 livello massimo).

Nella situazione in cui attualmente sia presente una connessione attiva vengono invece gestiti i protocolli software di comunicazione, sia quello di "basso livello" definito dal modulo Bluetooth RN-42, sia quello del "livello superiore" da noi definito per la gestione dell'invio/ricezione dei comandi e controllo dello stato di vicinanza tra i due dispositivi (nel box Protocollo di comunicazione diamo i dettagli delle regole di quest'ultimo protocollo).

Nel firmware segnaliamo la presenza di diversi timeout (tutti gestiti dall'unico interrupt del modulo hardware Timer 1): il primo è usato come sorta di "condizione di blocco software" per rilevare che la comunicazione Bluetooth non venga impegnata inutilmente da qualche dispositivo non autorizzato; infatti nel caso in cui questo timer scada la comunicazione viene chiusa e si torna nelle condizioni di attesa nuova connessione.

Il secondo timer è invece usato (in caso di allarme attivo) per rilevare quanto tempo è passato da una connessione alla successiva; nel caso in cui questo tempo sia superiore a una determinata soglia, l'allarme deve scattare.

Infine, l'ultimo timer è usato semplicemente per generare il lampeggio del LED verde di "sistema in vita" (questo nel caso in cui la



connessione Bluetooth sia DOWN perché nel tempo in cui la connessione è ON il LED verde risulta acceso fisso).

SOFTWARE ANDROID

Analizziamo ora il software per smartphone Android; come già ampiamente spiegato, tutto il sistema si basa sull'utilizzo del Bluetooth (in particolare il 2.0) per rilevare la presenza e distanza tra i due dispositivi. Pertanto risulta ovvio che condizione necessaria per funzionare è che lo smartphone disponga di tale tecnologia (ma oramai tutti i dispositivi lo posseggono) e che prima di avviare il software, dalle impostazioni del sistema operativo il Bluetooth risulti attivo.

Il software è costituito da una pagina principale dove è possibile attivare/disattivare l'allarme e da due ulteriori pagine minori (richiamabili da menu Android) rispettivamente dove è possibile scansionare i dispositivi Bluetooth e scegliere a quale legarsi e una pagina di configurazione della scheda (nome e PIN code Bluetooth della scheda e selezione livello della potenza di trasmissione).

Avviato il software compare un primo splash screen di benvenuto (**Fig. A**) e in seguito la pagina di scansione e/o selezione del dispositivo a cui collegarsi (**Fig. B**). Selezionato il dispositivo, se è la prima volta che ci si collega, compare la richiesta di digitazione del codice PIN (di default la scheda viene programmata con 5555 ma, come vedremo più avanti, questo può essere cambiato a piacere dall'utente) necessario per il pairing Bluetooth (**Fig. C**).

Dopo che il collegamento viene stabilito, il software Android si collega alla scheda, legge lo stato di attivazione allarme ed aggiorna di conseguenza la grafica della pagina principale (**Fig. D** in caso di allar-

me disattivo). Premendo il pulsante "Attiva allarme" viene inviato il comando di attivazione (a display compare un messaggio, **Fig. E**) ed a comando eseguito la grafica della pagina viene nuovamente aggiornata tenendo conto del nuovo stato (**Fig. D** in caso di allarme disattivo, **Fig. E** in caso di allarme attivo).

A questo punto, con l'allarme attivo, il software continua ciclicamente a collegarsi alla scheda ed invia/riceve dei dati come verifica del collegamento. Se questa operazione fallisce, lo smartphone ci avvisa tramite vibrazione, allarme sonoro e messaggio sul display (**Fig. G**).

Facciamo notare che anche in caso di tentativo fallito, il software ad intervalli regolari continua la sua esecuzione di nuovi collegamenti e se questi ancora falliscono ci avvisa nuovamente oppure, in caso di collegamento ristabilito, chiude la segnalazione dall'allarme.

In caso di allarme disattivato, è abilitato il menu Android dell'applicazione (**Fig. H**). Per evitare la segnalazione di falsi allarmi, abbiamo scelto di abilitare questo menu solo in caso che l'allarme sia disabilitato. Ci sembra una scelta ovvia in quanto se l'allarme è attivo ci sembra poco probabile che si decida di modificare le imposta-

zioni. Nel menu sono presenti due voci: "Connessione" e "Settings". La prima visualizza la schermata di scansione/selezione dei dispositivi (già vista in **Fig. B**) mentre la seconda apre la schermata di impostazione (**Fig. I**). Come si vede le impostazioni programmabili sono il nome del sistema, il codice PIN e il livello della potenza in trasmissione. Per le prime due sono presenti un campo di testo ciascuno, mentre per l'ultima è presente una barra di progresso per la selezione. Ognuna delle 3 impostazioni dispone di un pulsante di invio del nuovo valore selezionato.

L'invio di una delle tre impostazioni, per rendere effettive le modifiche effettuate, richiede che il modulo Bluetooth presente sull'elettronica venga resettato (da software Android viene mostrato un messaggio di avviso, **Fig. J**).

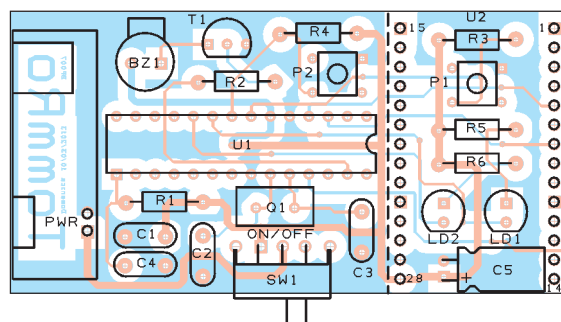
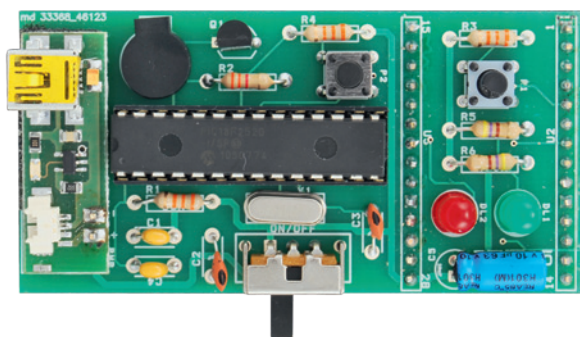
REALIZZAZIONE PRATICA

Il sistema presentato è stato progettato utilizzando solo componenti a montaggio con fori passanti (e pertanto realizzabile anche in proprio disponendo di un saldatore); unico componente in SMD è il socket adattatore RN-42 che però consigliamo di acquistare da Futura Elettronica

Nota importante per il montaggio

Evidenziamo questa operazione del montaggio in quanto anche noi, durante la costruzione del primo prototipo, siamo incappati in un errore simile che ha avuto come conseguenza la distruzione definitiva del campione (stampato e modulo Bluetooth). Il socket FT1018M viene fornito da Futura Elettronica con i 4 piccoli ponticelli sotto lo stampato predisposti ma non realizzati; questo per lasciare più libertà di configurazione all'utilizzatore finale per i propri progetti. Nel nostro caso questi ponticelli vanno chiusi (tramite delle piccole gocce di stagno) altrimenti la configurazione del modulo RN-42 non è compatibile col PIC18F2520 della scheda elettronica.

Durante montaggio del primo prototipo della scheda ci è capitato di dimenticarci e di saldare i 28 pin del socket; quando ci siamo accorti dell'errore era oramai troppo tardi e non siamo riusciti a dissaldare il socket senza rovinare le piste della scheda con il risultato che l'abbiamo rovinata definitivamente e pertanto abbiamo dovuto buttare via il tutto.



Elenco Componenti:

R1, R3, R4: 33 kohm

R2: 3,3 kohm

R5, R6: 470 ohm

C1, C4: 100 nF multistrato

C2, C3: 15 pF ceramico

C5: 10 μ F 63 VL elettrolitico

Q1: Quarzo 4 MHz

U1: PIC18F2520 (MF1084)

U2: Socket RN-42 (FT1018M)

T1: BC547

P1, P2: Microswitch

LD1: LED 5 mm verde

LD2: LED 5 mm rosso

BZ1: Buzzer 3,6V

SW1: Deviatore a slitta 90°

Varie:

- Modulo alimentazione FT864M
- Zoccolo 14+14
- Strip maschio 2 poli
- Circuito stampato

già montato e predisposto (codice articolo FT1018M).

Come si vede nelle immagini presenti nell'articolo (che per chiarezza consigliamo durante il montaggio di consultare continuamente), tutti i componenti vanno montati sul lato componenti ad eccezione del solo modulo Bluetooth che invece deve essere montato dal lato saldature (e pertanto in ordine temporale andrà montato per ultimo).

Dopo essersi procurati lo stampato e tutti i vari componenti, il primo passo è iniziare a saldare i componenti più piccoli (resistenze, condensatori, pulsanti, interruttore di accensione e LED, possibilmente in questo ordine).

Segnaliamo che alla fine la scheda andrà inserita all'interno di un contenitore di plastica; pertanto ricordiamo di effettuare il montaggio cercando di mantenere l'altezza più piccola possibile. Cercate di montare le resistenze ed i condensatori (ma tutti i componenti in generale) il più possibile aderenti allo stampato; in particolare consigliamo di montare il conden-

satore C5 piegato di 90° (quindi in posizione orizzontale); segnaliamo inoltre che, sempre per lo stesso motivo, abbiamo scelto un quarzo non tradizionale, ma ad altezza ridotta.

Terminato il montaggio di questi componenti, è il momento di montare il PIC; nel nostro esempio, per comodità, abbiamo scelto di utilizzare uno zoccolo; nel vostro caso scegliete voi se utilizzarlo oppure se saldare direttamente il PIC (in questo caso non sarà più possibile o comunque più difficoltoso aggiornare/programmare nuovamente il suo firmware).

In seguito consideriamo il caricabatterie integrato: consigliamo di montarlo attraverso un pin strip passo 2,54mm di due poli. Consigliamo inoltre, per evitare inutili oscillazioni del modulo, di incollare sullo stampato (nella zona sotto questo componente) della spugna in modo da ammortizzare i movimenti.

Come primo passaggio saldate il pin strip di due poli sullo stampato; poi incollate la spugna; in seguito inserite il modulo cari-

cabatterie nei due poli e saldate il pin strip sul lato superiore del caricabatterie.

Terminate il montaggio saldando il modulo Bluetooth. Prima però ricordatevi di realizzare sotto al modulo stesso le 4 saldature relative ai 4 ponticelli "LED STATO", "LED CONNECT", "RESET" e "BAUD 9600" altrimenti l'integrato RN-42 e il PIC non si "parlano"; inoltre il PIC, durante la programmazione, non riuscirà a resettare il Bluetooth (per i dettagli riferitvi al primo articolo relativo al Bluetooth in cui mostriamo e descriviamo lo schema del socket). Il modulo Bluetooth andrà posizionato sul lato saldature tenendo conto della numerazione dei pin (per evitare errori fate riferimento alle immagini nell'articolo). Le saldature andranno effettuate sul lato componenti; comunque non essendoci nelle vicinanze particolari componenti, l'operazione risulta abbastanza agevole. Infine, inserite la scheda nel contenitore plastico. Per evitare che l'hardware subisca inutili urti consigliamo di ritagliare della



Tile, trovare gli oggetti smarriti con una App

Tile incorpora un piccolo buzzer che ci aiuta dell'individuazione in ambienti chiusi; la batteria non può essere sostituita e ha una durata di un anno, dopodiché bisogna acquistare un altro Tile il cui prezzo è di 18,95 dollari (circa 15 euro). L'applicazione, ideata da Nick Evans e Mike Farley, è stata finanziata da una campagna di crowdfunding che ha visto l'adesione di 49.586 persone per un finanziamento totale di 2.681.297 dollari. In questo caso non è stata utilizzata una delle solite piattaforme di crowdfunding (Kickstarter, Indiegogo, ecc.) ma Selfstarter (<http://selfstarter.us/>), una soluzione open source da caricare sul proprio server per gestire in completa autonomia la propria campagna.

Manco a farlo apposta, un'idea simile a quella di questo progetto è alla base di uno dei più grandi successi di crowdfunding degli ultimi mesi: il progetto Tile (www.thetileapp.com), che consiste in un tag miniatura in tecnologia Bluetooth 4.0 completo di batteria con autonomia di un anno e in una App che gira su iPhone 4S, iPhone 5, iPad mini, iPad di terza e quarta generazione e iPod Touch di quinta generazione. Il piccolo tag può essere fissato a qualsiasi oggetto, portachiavi, borsetta, valigia, bicicletta, eccetera, mentre l'App ci consente di individuare il luogo in cui abbiamo lasciato l'oggetto. Il tutto funziona entro un raggio di 15-45 metri e con un massimo di 10 tag associati a ciascun cellulare. Tuttavia se l'oggetto è stato rubato, possiamo abilitare i nostri amici alla ricerca del tag, e quindi dell'oggetto, in modo che il raggio di ricerca si possa espandere a piacere.



spugnetta sulle forme sia dello stampato, sia del contenitore: inseritela nella scatola a modo "cuscinetto" per la scheda (a nostro avviso è meglio abbondare di qualche millimetro, tanto una volta che la scheda è inserita difficilmente andrà rimossa).

La scheda è stata progettata tenendo conto delle dimensioni della batteria ricaricabile utilizzata. Come si vede nelle foto pubblicate, sul lato saldature il modulo RN-42

è stato posizionato il più esterno possibile vicino a un bordo, lasciando verso il lato opposto lo spazio necessario per contenere la batteria che, avendo altezza simile al componente Bluetooth, creerà con questo un incastro perfetto generando un bordo inferiore praticamente piatto.

La batteria deve essere posizionata sotto la scheda (sul lato saldature dove sono presenti le parti terminali delle saldature

stesse); inoltre la stessa è inserita in un piccolo contenitore. Col tempo queste parti terminali delle saldature potrebbero generare dei piccoli tagli che potrebbero creare dei corti circuiti e quindi guastare l'elettronica. Sugeriamo pertanto di isolare il lato della batteria che sarà a contatto con la scheda incollandoci sopra un'etichetta di carta abbastanza spessa (al massimo si possono sovrapporre più etichette).

Una volta che scheda e batteria sono all'interno del contenitore, chiudete il coperchio con la vite dello stesso. Per comodità, in corrispondenza dei LED e dell'interruttore, è possibile praticare dei fori o aperture nel contenitore di plastica in modo da renderli accessibili e visibili senza dover togliere ogni volta il coperchio superiore. Stesso discorso per la presa USB di carica batteria, anche se questa si utilizzerà meno frequentemente e quindi potrebbe non essere necessario.

Infine, se volete aumentare la potenza sonora dell'allarme, praticate un'apertura in corrispondenza dell'uscita del buzzer.

COLLAUDO

Terminato il montaggio dell'hardware, vediamo come testare il sistema e collegare l'elettronica al software Android.

Prima di tutto consigliamo di collegare la batteria al relativo connettore del modulo caricabatteria e lasciare in carica il sistema almeno per circa 15 minuti (meglio fino a carica completata; segnaliamo inoltre che il connettore della batteria non entra nel relativo connettore presente sul carica-batterie, pertanto non forzate mai l'inserimento ma prima provate a girare di 180° il connettore). Per accendere il sistema spostate l'interruttore della scheda da OFF a ON (fate riferimento alla serigrafatura).

fia dello stampato) e verificate che il LED verde si accenda o comunque lampeggi segnalando così che il software risulta attivo.

A questo punto potete già effettuare un primo test tramite i due pulsanti: premete P1 e verificate che venga attivato il buzzer e che i LED rispondano correttamente; premete P2 e verificate che venga riconosciuto correttamente e che i LED/cicalino segnalino il passaggio da un livello di potenza di trasmissione al successivo.

Se tutto è a posto, passate allo smartphone: per prima cosa andate nella pagina di configurazione WiFi Android, sezione Bluetooth ed avviate una scansione. Verificate che la scheda venga rilevata (nome di default: "Scheda Allarme") e se volete provate ad eseguire l'accoppiamento (PIN di default: 5555).

In seguito avviate il software *AntiTheftControl* che avrete scaricato dalla sezione download del nostro sito www.elettronica.in e installato sul vostro smartphone; dalla sezione di connessione eseguite un rilevamento dei dispositivi Bluetooth attivi e selezionate la vostra scheda. Attendete qualche secondo per la connessione e verificate che si attivi il pulsante "Attiva allarme", a conferma che il primo collegamento è andato a buon fine e che attualmente l'allarme sia disattivo.

Se volete, tramite il menu, potete visualizzare la pagina di configurazione e modificare le impostazioni secondo le vostre necessità. Unica raccomandazione: state attenti se modificate il PIN perché se sbagliate a digitarlo e/o lo dimenticate poi non potrete più usare la scheda (a meno di non disporre di un programmatore per PIC per riprogrammarlo; diversamente dovrete richiederne uno nuovo a Futura Elettronica). Tornati alla pagina principale,

attivate l'allarme e verificate che si accenda il LED rosso. Provate a muovervi nella stanza e nello spazio circostante e verificate che il link sia sempre attivo e non ci siano falsi allarmi. In seguito provate un allarme. Se non volete allontanarvi troppo fate così: chiudete il software Android e verificate che dopo alcuni attimi la scheda attivi il cicalino a segnalazione della condizione. Riavviate nuovamente il software, collegatevi nuovamente alla scheda e verificate che, quando la connessione è ristabilita, l'allarme venga fermato.

A questo punto spegnete la scheda e verificate che in questo caso scatti l'allarme sullo smartphone. Anche in questo caso provate a riaccendere la scheda e controllate che l'allarme sullo smartphone venga resettato.

Prima di concludere il test ricordatevi di disattivare l'allarme altrimenti al prossimo riavvio si attiverà in automatico e la scheda accenderà il buzzer.

Per concludere segnaliamo che, essendo il sistema funzionante a batteria, ha un certo periodo temporale di autonomia. Dalle nostre prove possiamo dire che questa è più che sufficiente per il normale utilizzo a cui il sistema è destinato; chiaramente abbiamo utilizzato una batteria nuova e correttamente ricaricata. Probabilmente con il passare del tempo le prestazioni della batteria peggioreranno; se utilizzate il sistema intensamente consigliamo ogni tanto di sostituire la batteria con una nuova.

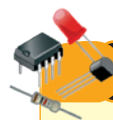
POSSIBILI MIGLIORAMENTI

Il progetto presentato in queste pagine permette di gestire un'unica periferica Bluetooth alla volta. In effetti, però, acquistando più kit, si potrebbe pensare di modificare il software in modo che da un unico smartphone sia possibile controllare diversi mo-

duli inseriti all'interno di diversi oggetti.

Il sistema di sviluppo Android permette di gestire più link Bluetooth verso diversi slave oppure - e questo a nostro parere è più semplice da realizzare - gestire una connessione alla volta e continuamente passare in sequenza da una scheda alla successiva. Scegliendo questa seconda strada, il firmware del PIC potrebbe non richiedere alcuna modifica (al massimo un minimo di aumento del timeout di rilevazione allarme per tenere conto che lo smartphone ha più periferiche da controllare e pertanto potrebbe passare qualche attimo in più prima che nuovamente porti a termine il ciclo); il software Android invece andrebbe modificato, ma non più di tanto.

Per il momento abbiamo deciso di lasciare questo compito a qualche lettore volenteroso; fateci comunque sapere se vi può interessare l'evoluzione, eventualmente torneremo sull'argomento. ■

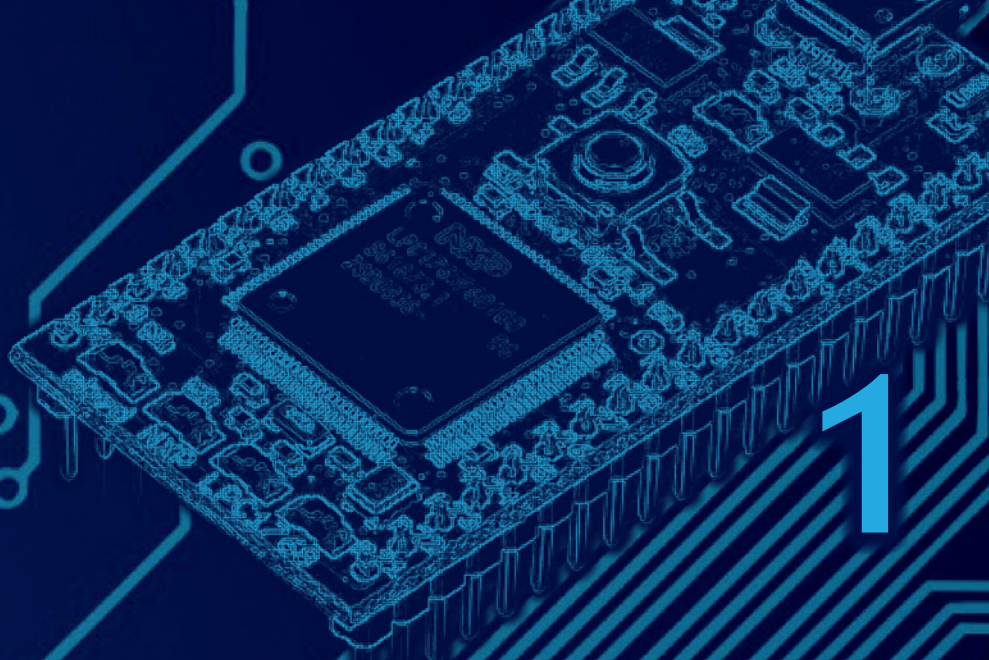


per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come il firmware utilizzato per programmare il microcontrollore PIC18F2520 ed il software per il cellulare Android. Il socket Bluetooth con RN-42 (cod. FT1018M) costa 29,00 Euro mentre il modulo di alimentazione FT864M costa 15,00 Euro. 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>

È un'alternativa più avanzata alla popolare scheda embedded ad 8 BIT (Arduino) e può contare sulla potenza di calcolo di MCU ad architettura ARM Cortex; più compatta e prestante, vi mostreremo passo passo come è possibile importare il codice scritto per Arduino e come usarne gli shield. Prima puntata.



Conoscere e usare

mbed

dell'ing. Vincenzo Mendola

Negli ultimi due anni si è fatto un gran parlare di Arduino, tanto che chi ha approcciato il mondo dell'elettronica con questa board o coloro che l'hanno conosciuta per hobby, probabilmente non si sono accorti che sulla scia di Arduino sono nate tante altre schede embedded; non parliamo degli innumerevoli cloni e varianti più o meno compatibili (Microchip, ad esempio), ma di un sistema di sviluppo e prototipazione che non tutti conoscono ma che è molto più potente e potenzialmente più versatile di Arduino, anche se magari meno immediato da utilizzare in maniera proficua da parte dei neofiti: mbed. Disponibile da alcuni anni e

sicuramente meno pubblicizzata, mbed è nata da un'idea (parliamo del 2005) di due dipendenti di ARM, anch'essi spinti dal desiderio, come nel caso del team di Arduino, di rendere disponibile a tutti gli appassionati di elettronica e autocostruzione dotati di conoscenze tecniche e di programmazione essenziali, un solido supporto hardware su cui sviluppare in maniera rapida ed efficace le proprie idee; da qui deriva la definizione da essi utilizzata di "tool for rapid prototyping with microcontrollers" (punto [1] dei *Riferimenti*).

Nel corso vi mostreremo come utilizzare gli shield di Arduino e come fare il porting del relativo codice su questa piattaforma.

mbed è attualmente disponibile in due versioni,

Fig. 1

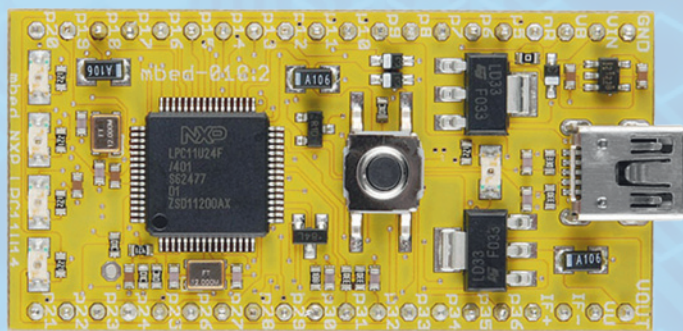
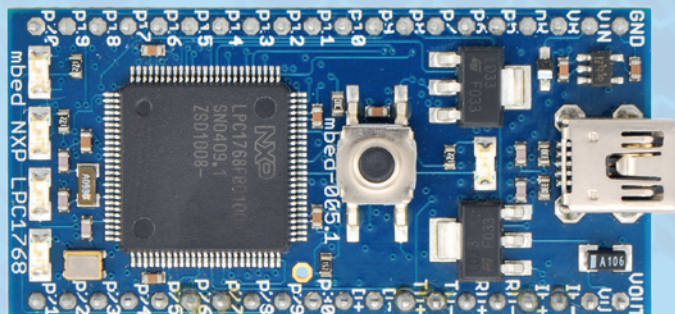


Fig. 2

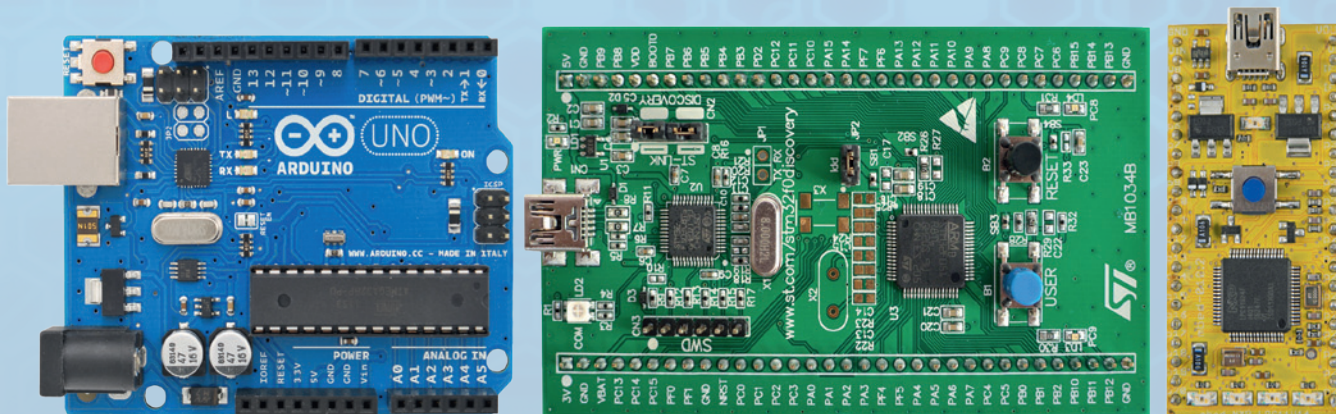


una con microprocessore Cortex M0 (<http://mbed.org/handbook/mbed-NXP-LPC1114>) visibile in Fig.1 e l'altra con Cortex M3 (<http://mbed.org/handbook/mbed-NXP-LPC1768>) visibile in Fig. 2.

A queste si è aggiunta recentemente una terza versione, il cui hardware non è stato sviluppato direttamente dal team mbed ma dalla Freescale: questa versione è commercializzata come "mbed ready", dato che per utilizzarla come una board mbed è sufficiente aggiornarne il firmware.

Questa scheda ha un form factor differente dalle altre due, dalle quali differisce per alcune peculiarità; per questa ragione abbiamo

Fig. 3



NXP LPC810: il primo Cortex M0+ in un package 8 DIP, al costo di una MCU a 8 BIT come l'ATTINY 85.



deciso di dedicarle un'intera puntata delle successive che formano questo corso.

PERCHÉ 32 BIT

La drastica diminuzione dei costi dei semiconduttori e dei microcontrollori in particolare, ha determinato la rapida diffusione di vari prodotti, rendendo disponibile la potenza dei micro a 32 bit anche agli appassionati, oltretutto ai professionisti.

Ma quali sono realmente i vantaggi dei microcontrollori a 32 bit? Ebbene, non esiste una risposta univoca, perché nel confrontare i sistemi a 8 e 16 bit con quelli a 32 andrebbero bilanciati i vantaggi con gli svantaggi delle differenti architetture, perciò volendo rispondere in maniera un po' semplicistica e grossolana, possiamo prendere spunto da quanto riportato dalla stessa ARM (punto [2] dei Riferimenti) nella pagina di presentazione dei propri microcontrollori Cortex-M (le ultime due considerazioni sono chiaramente dirette più alla tecnologia ARM che all'architettura a 32 bit).

In particolare, l'architettura a 32 bit offre:

- maggiore efficienza energetica; essendo più veloci, i microcontrollori possono eseguire le stesse operazioni richieste dai micro a 8 e 16 bit in meno tempo e quindi tornare più rapidamente e per tempi più lunghi in una condizione di funzionamento a basso consumo (sleep mode) ;
- codice più compatto, grazie all'alta densità del set di istruzioni, in grado di effettuare un maggior numero di operazioni per byte;

- facilità d'uso, in virtù della portabilità del codice, reso compatibile con una vasta gamma di prodotti di brand differenti (grazie alla standardizzazione dell'architettura assicurata dalla ARM, che ha permesso anche alle software-house che si occupano della realizzazione dei tool di sviluppo software l'unificazione di molti dispositivi in un unico prodotto);
- migliori prestazioni garantite dalla potenza dei Cortex-M, caratterizzati da una più elevata potenza di calcolo per MHz di clock rispetto ai micro a 8 e 16 bit.

A queste motivazioni va aggiunto il fatto che oggi possiamo contare su un'ampia disponibilità a costi relativamente bassi dei micro a 32 bit, il che ha reso conveniente usare microcontrollori del genere anche in ambiti fino a poco tempo fa tipicamente adatti ai micro ad 8 bit.

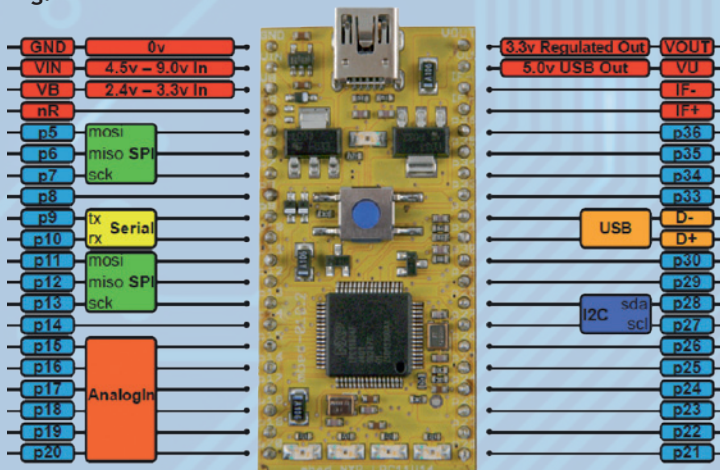
Come si vede in Fig. 3, la scheda mbed è grande meno di 1/3 della Discovery e dell'Arduino, grazie ad una miniaturizzazione spinta e ad un'accurata ingegnerizzazione. Questa caratteristica (non è di certo l'unica) rientra tra gli elementi che determinano la grande differenza di prezzo tra una Discovery F0 con microcontrollore ST STM32F051 e l'MBED LCP11U24, entrambe dotate di Cortex-M0 e quindi con prestazioni e caratteristiche comparabili.

Come molti di voi sanno, ARM non produce direttamente i propri processori, ma ne sviluppa la tecnologia che poi viene venduta sotto licenza ai produttori di semiconduttori, i quali la implementano nei propri prodotti: per questo, in alcuni microcontrollori leggiamo, tra le caratteristiche, "basato su tecnologia ARM" o "con tecnologia ARM".

Tra i produttori che usano ARM possiamo citare Texas Instruments, Toshiba, Analog Devices, NXP, ST, Atmel, Samsung, Freescale, Fujitsu, Renesas e Broadcom.

Per l'mbed sono state scelte due MCU della NXP, che in entrambi i casi danno il nome al dispositivo: si tratta delle LPC11U24 e LPC1768. La NXP LPC11U24 si basa "soltanto" su di un core ARM® Cortex™-M0 low-power a 48 MHz, con 8 kB di RAM e 32 kB di memoria Flash; le periferiche in dotazione (Fig. 4 e punto [3] dei Riferimenti) sono

Fig. 4

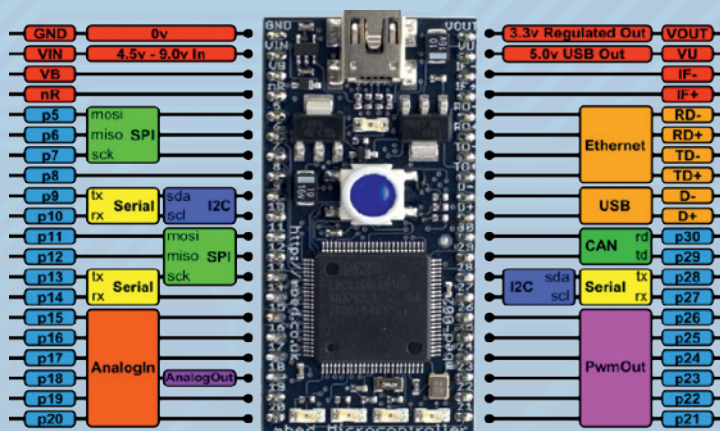


un USB di tipo esclusivamente "Device" (cioè può solo essere gestita da un host e non collegarsi ad altri apparati USB analoghi), 2 bus SPI, una I²C, un'UART, 6 ADC a 12 bit, più 30 pin che possono lavorare come GPIO (General Purpose Input/Output).

Il "cuore" dell'mbed NXP LPC1768 MCU è invece costituito da un ARM® Cortex™-M3, Core a 96 MHz, dotato di 32 kB di RAM e di 512 kB di Flash. Le periferiche disponibili in questo caso (Fig. 5) includono anche una porta Ethernet, USB funzionanti sia come Host che come Device, 2 bus SPI, 2 I²C, 3 UART, una porta CAN bus, 6 moduli PWM e altrettanti ADC, DAC, più 26 GPIO (le caratteristiche complete si possono consultare alla pagina web raggiungibile dal link al punto [4] dei Riferimenti). La MCU LPC1768 ha anche un Real Time Clock integrato.

Entrambe le MCU usate nell'mbed hanno un package DIP a 40 pin, con passo 0,1", che misura soltanto 54x26mm. Per ulteriori dettagli

Fig. 5



[Get an mbed!](#) |
 [Subscribe to the mailing list](#) |
 [Login or signup](#)

[Blog](#) |
 [Forum](#) |
 [Handbook](#) |
 [Cookbook](#) |
 [Code](#) |
 [Compiler](#)

Login

Username:

[I've forgotten my username](#)

Password:

[I've forgotten my password](#)

☐ Remember me

Signup

© mbed | [blog](#) | [get an mbed](#) | [about mbed](#) | [we're hiring!](#) | [support](#) | [service status](#) | [privacy policy](#) | [terms and conditions](#) | Language: en ja es de

mbed, the fastest way to prototype with ARM based microcontrollers.


[Blog](#) | [Forum](#) | [Handbook](#) | [Cookbook](#) | [Code](#) | [My Dashboard](#) | [My Code](#) | [Compiler](#)

[NXP](#) = mbed NXP LPC1768


[mbed NXP LPC1768 board connected!](#) [\(More info\)](#)

mbed NXP LPC1768

Getting Started

If you are just starting out, try downloading the "Hello World" program binary to your microcontroller to test things are working. Save it to your mbed Microcontroller Disk, and then hit the reset button to run it. For more things to try, see the [mbed Hardware](#) "Getting Started" guides.

Reference materials

 Further information about this microcontroller - Pinout guide, technical reference and essential facts

 [mbed Handbook](#) - The primary mbed reference



HelloWorld.bin
 Save to your mbed, and hit reset!



Getting Started
 Write your first mbed program

valutare bene le proprie necessità e considerare le possibili esigenze future, tenendo presente che la maggiore completezza e complessità circuitale dell'LPC1768, caratterizzata dalla disponibilità di molte funzionalità, è controbilanciata da maggiori consumi, fattore cruciale nelle applicazioni portatili in cui l'autonomia è un elemento determinante. La valutazione dei futuri sviluppi di un'applicazione è anch'essa un fattore chiave, perché magari scegliendo la versione basata su LPC1114 si spende inizialmente meno, ma se un'applicazione richiede l'aggiunta di una periferica non implementata nella MCU, che va quindi aggiunta a parte (ad esempio un'interfaccia Ethernet, una connessione USB di tipo Host invece che Device, oppure un RTC) è facile che i chip e il circuito da aggiungere facciano costare il tutto più della versione mbed più completa.

Le schede mbed sono entrambe plug and play: basta collegarle al PC mediante il cavo USB in dotazione e sono subito pronte a funzionare. Appena connesse, un LED blu posizionato vicino al connettore mini-USB inizierà a lampeggiare velocemente, per poi rimanere acceso una volta effettuato in maniera stabile il collegamento con il computer. Le mbed vengono riconosciute dal PC (lo abbiamo testato sia su Windows che su Linux) come memoria flash (USB Mass Storage Device) da 2 MB di capacità. Esplorandone il contenuto, vediamo che in questa memoria è presente il file *mbed.htm*, aprendo il quale il browser del sistema operativo darà accesso a una pagina web (**Fig. 6**) dalla quale potrete registrare gratuitamente il vostro dispositivo (<https://mbed.org/account/login/?next=/>); la procedura richiede, chiaramente, la connessione del PC ad Internet.

A questo riguardo va detto che, come segnalato anche dallo stesso team di sviluppo di mbed (punto [8] dei riferimenti) e personalmente sperimentato dall'autore, in alcuni casi il cavo USB fornito di serie nella confezione può creare problemi come il mancato ricono-

More info on your board

Board: Rev: mbed-005-1, Serial: 101000000000000000002FF0CDBB, Batch: 184, Firmware version: **21164**. Your firmware is up to date.

You will be automatically notified of any firmware upgrades.

Note that the board detection and firmware notification **only** works when you access the mbed site by opening MBED.HTM on an mbed board.

 HelloWorld.bin
Save to your mbed, and hit reset!

Fig. 10

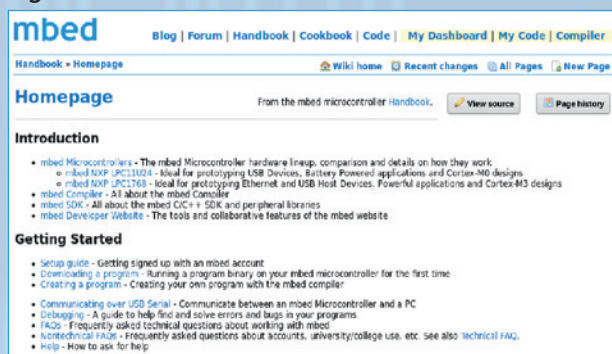


Fig. 11

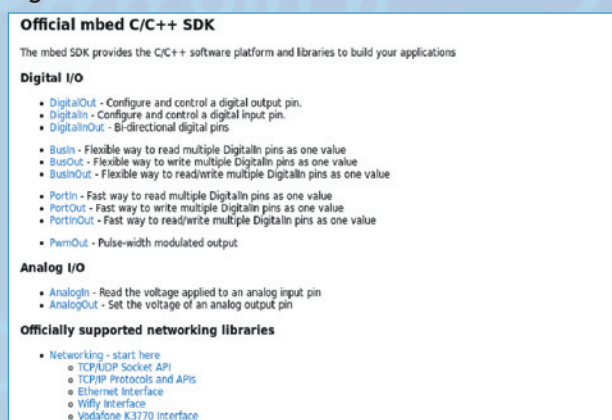
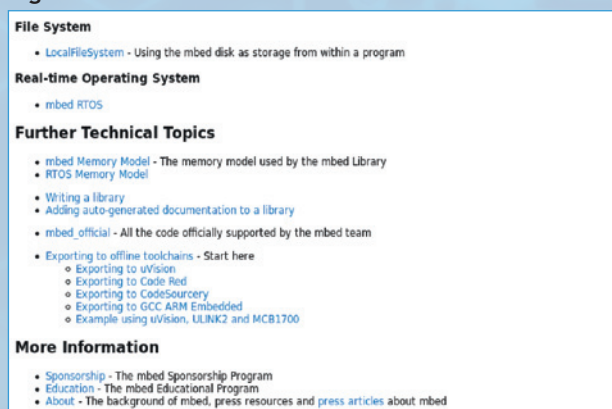


Fig. 12



Fig. 13



scimento della memoria Flash dell'mbed: ciò può dipendere dal chipset USB presente sulla scheda madre del computer, in quanto alcuni modelli sono particolarmente sensibili alla qualità del cavo. In questo caso è necessario cambiare cavo, optando per uno più corto e munito della ferrite di blocco della RF, ma soprattutto di qualità elevata (per qualità, si intende a bassa attenuazione e diafonia). Una volta effettuata la registrazione ed effettuato l'accesso, ci ritroveremo nella pagina mostrata in Fig. 7, nella quale in alto è indicato il modello di mbed che stiamo utilizzando (nel caso specifico, una LPC1768); inoltre ci viene segnalato che la scheda mbed è correttamente connessa al PC. Cliccando su *more info* ci verranno mostrate ulteriori informazioni sulla nostra scheda: revisione, numero di serie, batch e versione firmware, oltre all'indicazione sullo stato di aggiornamento di quest'ultimo (Fig. 8).

Torniamo ora alla pagina di Fig. 7: da qui possiamo raggiungere le principali sezioni del sito ed iniziare a lavorare con la nostra mbed. Per prima cosa proviamo a caricare il nostro primo programma tramite il link visibile in Fig. 9; una volta salvato il file "HelloWorld.bin" nella memoria del micro, resettiamo quest'ultimo premendo il pulsante presente sul lato superiore della scheda. Dopo pochi istanti, il primo LED in basso a sinistra inizierà a lampeggiare. La procedura di download qui descritta è dettagliata anche alla pagina web <https://mbed.org/handbook/Downloading-a-program>.

Seguendo il link "Getting Started" posizionato subito sotto quello appena utilizzato per caricare HelloWorld.bin, ci verranno fornite maggiori informazioni circa l'uso del nostro sistema (Fig. 10, 11, 12, 13).

Procediamo per gradi: l'IDE degli mbed è rilasciato solo in versione cloud (gratuita), una volta effettuato il login; tale scelta pare essere stata inizialmente dettata dalle difficoltà incontrate dal team di sviluppo nell'installare il software necessario per la programmazione e lo sviluppo del codice sui computer delle scuole e delle università, a causa delle forti restrizioni che spesso vengono implementate dai responsabili IT per garantire la sicurezza dei sistemi informatici, utilizzati da giovani curiosi e con tanta voglia di sperimentare,

che a volte possono causare inconsapevolmente gravi problemi. L'IDE è raggiungibile all'indirizzo <https://mbed.org/compiler/>, oppure più semplicemente, servendosi del link in alto a destra indicato come "Compiler".

Nella pagina "Creating a program" (<https://mbed.org/handbook/Creating-a-program>) sono riassunti i principali passaggi da seguire per creare e caricare in mbed il nostro primo programma ed è mostrato il codice di "HelloWorld" (Fig. 14).

In alto a destra c'è un link chiamato "Import this program" che ci permette di caricare il codice nell'IDE. Nella colonna di sinistra "Program workspace" troviamo il programma appena caricato, il cui codice è visibile nell'area centrale dell'IDE, selezionando *main.cpp* (Fig. 15).

Il file "mbed", visibile nello spazio di lavoro sotto a *main.cpp*, e identificato tramite un'icona a forma di ingranaggio, è la libreria che contiene tutte le istruzioni per controllare e far funzionare mbed.

Veniamo al codice che fa lampeggiare il LED1 (Fig. 16): la prima riga contiene l'istruzione "#include mbed.h" che, come abbiamo appena visto, dichiara al compilatore di includere la libreria delle funzioni base del nostro dispositivo; la riga successiva contiene il codice "DigitalOut myled(LED1);", che definisce il pin collegato al LED1 come uscita digitale, associandogli l'etichetta "myled". Le righe contenute all'interno di "int main()" tra le due parentesi graffe sono quelle che fanno lampeggiare il LED: "while(1)" definisce un loop infinito per le istruzioni seguenti

che sono "myled = 1;" che accende il LED1, mentre "myled = 0;" riporta l'uscita a questo associato a livello basso, spegnendolo. Questi due comandi sono intervallati da "wait(0.2);", che introduce un ritardo di 0,2 secondi. Terminata l'esecuzione dell'ultima istruzione, il codice si ripete indefinitamente fintantoché l'mbed è alimentato. Il codice appena visto è funzionalmente equivalente a quello utilizzato da Arduino per far lampeggiare il LED 13 presente sulla scheda (<http://arduino.cc/en/Tutorial/Blink>), come si vede nel Listato 1.

Notiamo subito che nel codice di "HelloWorld" manca il "void setup()" che in questo ambiente non viene utilizzato, mentre il "void loop()" è diventato "int main()" il cui contenuto viene eseguito in loop dall'istruzione C "while(1){}"; notiamo anche che "pinMode(led, OUTPUT);", diventa l'equivalente "DigitalOut myled(LED1);", e che "digitalWrite(led, HIGH);", corrisponde a "myled = 1;". Inoltre, "digitalWrite(led, LOW);", è diventato "myled = 0;". Infine, il comando "delay(1000);", è stato tradotto in "wait(1);". L'mbed utilizza la funzione "wait()" il cui argomento rappresenta il numero di secondi di ritardo da introdurre (con risoluzione al microsecondo), mentre "wait_ms" e "wait_us" hanno come argomento rispettivamente i millisecondi o i microsecondi che verranno lasciati trascorrere prima di eseguire l'istruzione successiva.

VISUALIZZARE I DATI SULLA COM VIRTUALE

Prima di passare ad un esempio più comples-

Fig. 14

```

#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}

```

Fig. 15

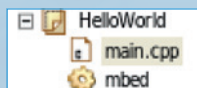


Fig. 16

```

1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6     while(1) {
7         myled = 1;
8         wait(0.2);
9         myled = 0;
10        wait(0.2);
11    }
12 }
13

```


Listato 1

```

void setup() {
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000);             // wait for a second
    digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
    delay(1000);             // wait for a second
}

```

so, che nel nostro caso prevede l'acquisizione del segnale di un trasduttore di temperatura analogico tramite l'ADC del micro, spieghiamo come si possono visualizzare i dati che viaggiano sulla seriale virtuale del nostro mbed, emulata dalla connessione USB. A riguardo, sul sito mbed esistono numerose pagine dedicate che spiegano nel dettaglio come utilizzare la seriale via USB: le trovate ai link descritti nei punti [9] e [10] dei *Riferimenti*. Questa operazione non può essere eseguita direttamente tramite l'IDE, come siamo abituati a fare con Arduino, ma richiede un ulteriore software emulatore di terminale da far girare sul PC; l'handbook consiglia di usare TeraTerm, ottimo e versatile, nonché il diffusissimo e conosciutissimo PuTTY, ovvero, anche Terminal by Bray (punto [11] dei *Riferimenti*). Sono tutti e tre adatti allo scopo e l'utilizzo di uno rispetto all'altro è da ricondursi esclusivamente a preferenze personali. In realtà la scelta non è limitata a questi tre software, perché potete usare qualunque altro programma in grado di gestire le connessioni e le comunicazioni sulle seriali (virtuali o fisiche) del PC.

Vediamo più nel dettaglio i primi due, i quali sono certamente quelli che offrono l'interfaccia utente migliore; quanto a Terminal di Bray, funziona in modo analogo ai precedenti: si seleziona la COM da monitorare e i dati seriali vengono mostrati in un'apposita area presente nella schermata principale del software, nella quale si impostano i vari parametri della connessione.

TERATERM

Accedendo al link indicato al punto [11] dei *Riferimenti*, scarichiamo l'ultima versione del software da sourceforge (<http://sourceforge.jp/projects/ttssh2/releases/>). Il programma è

disponibile in due versioni: quella eseguibile, da installare nel PC, e la versione compressa, che non necessita di installazione. Spieghiamo come servirsi della seconda, dato che la prima ha una procedura di installazione guidata e quindi si spiega da sé: una volta scompattato in una cartella di nostra scelta (mantenendo comunque la struttura originale delle sottocartelle), eseguiamo cliccando due volte sull'icona del file *termpro.exe*, allorché si aprirà la finestra riportata in Fig. 17, dove selezioniamo "Serial" e la corrispondente COM (parliamo della COM virtuale corrispondente all'USB) cui è assegnata la nostra mbed. In caso di dubbio è possibile verificare quale è la COM virtuale, accedendo alla pagina del sistema operativo che ci elenca le periferiche del computer (su Windows 7 è raggiungibile da *Pannello di controllo\Sistema e Sicurezza\Sistema* selezionando *Gestione dispositivi* e da qui *Porte COM e LPT*). La scheda mbed viene identificata come *mbed Virtual Serial Port (COMx)*, dove *x* è il numero assegnato alla COM virtuale e corrisponde alla USB presente ai terminali 31 e 32 degli mbed.

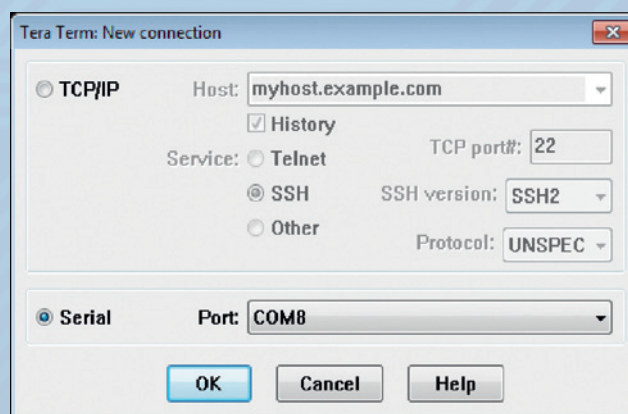


Fig. 17

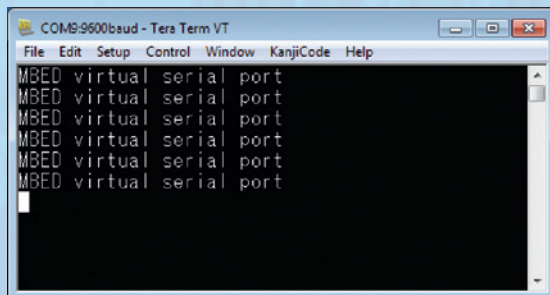


Fig. 18

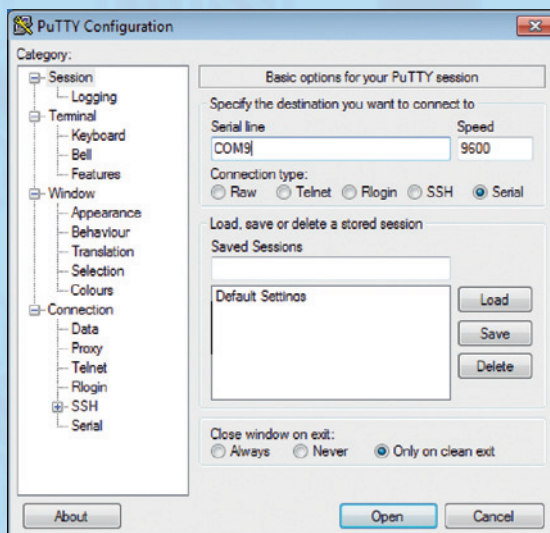


Fig. 19

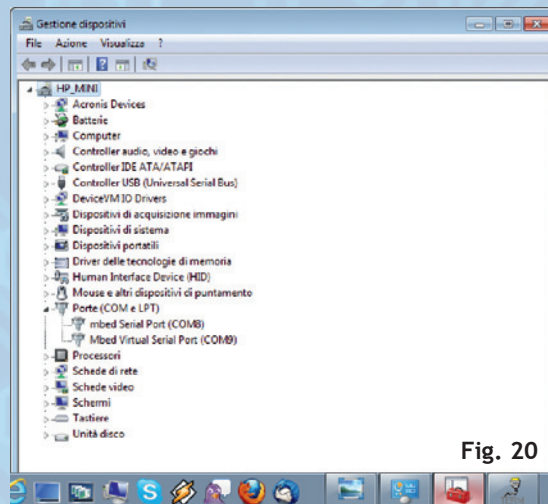


Fig. 20

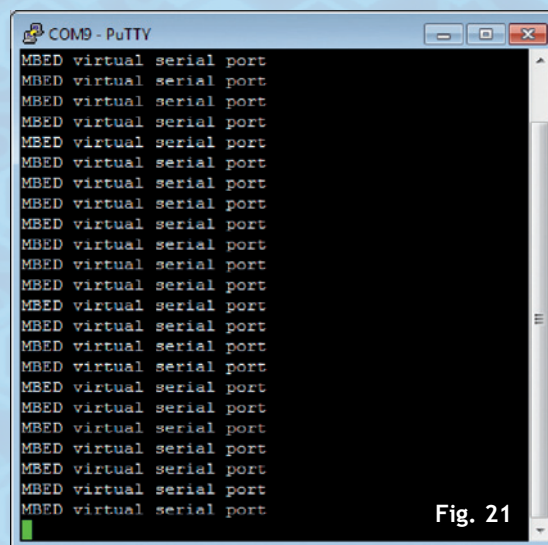


Fig. 21

Ora carichiamo sull'mbed il semplice codice mostrato nel **Listato 2**.

Sulla schermata del terminale avremo quanto riportato nella **Fig. 18**. La scritta si ripeterà finché non chiuderemo la connessione o non toglieremo alimentazione all'mbed.

Per iniziare una nuova connessione con TeraTerm si deve selezionare nel menu "File" "New connection", mentre per terminarla basta selezionare "Disconnect" (sempre da "File").

PuTTY

PuTTY è sicuramente il più conosciuto tra i client telnet e SSH del mondo open source; si scarica in FTP dal sito <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>, lo stesso nel quale si possono trovare anche i file sorgente. Lanciando "putty.exe" con un semplice doppio clic (non è necessaria alcuna installazione) si apre la finestra di dialogo, nella quale basta selezionare la COM in maniera analoga a quanto fatto precedentemente con Teraterm (la **Fig. 19** mostra le impostazioni del caso) e il programma è pronto per svolgere il proprio compito. La **Fig. 21** mostra una tipica schermata di ter-

Listato 2

```
#include "mbed.h"
#include "USBSerial.h"

//Virtual serial port over USB
USBSerial serial;

int main(void) {

    while(1)
    {
        serial.printf("MBED virtual serial port\r\n");
        wait(1);
    }
}
```

Listato 3

```
#include "mbed.h"

Serial pc(USBTX, USBRX); // tx, rx

int main(void) {while(1){
    pc.printf("MBED serial port\r\n");
    wait(1); }
}
```

Listato 4

```
#include "mbed.h"

Serial pc(USBTX, USBRX); // tx, rx
PwmOut led(LED1);

float brightness = 0.0;

int main() {
    pc.printf("Press 'u' to turn LED1 brightness up, 'd' to turn it down\n");

    while(1) {
        char c = pc.getc();
        if((c == 'u') && (brightness < 0.5)) {
            brightness += 0.01;
            led = brightness;
        }
        if((c == 'd') && (brightness > 0.0)) {
            brightness -= 0.01;
            led = brightness;
        }
    }
}
```

minale del software PuTTY, con il dettaglio dei dati scambiati sulla seriale. Se colleghiamo anche la mini-USB di cui ci serviamo per caricare il codice sul nostro mbed, vediamo che questa verrà riconosciuta come un'ulteriore porta COM (Fig. 20) virtuale, identificata in questo caso come *mbed Serial Port*.

Vediamo dunque come utilizzare anche questa seriale: possiamo farlo mediante il semplice codice d'esempio riportato nel **Listato 3**. Otteniamo così un risultato analogo a quello del caso precedente (**Listato 2**) ma questa volta non abbiamo fatto uso della libreria "USBSerial.h", di conseguenza il comando `serial.printf("")` è diventato `pc.printf("")` ("r" e "n" hanno la funzione rispettivamente di "ritorno carrello" e "nuova riga").

Prima di chiudere questa prima puntata del corso, proviamo ad eseguire un programma più articolato (**Listato 4**).

Questo codice, presentato nella pagina dedicata alla seriale del mbed (link di cui al punto [9] dei *Riferimenti*) ci mostra un esempio di comunicazione bidirezionale sulla COM del nostro mbed; come indicato sul terminale, premendo il tasto "u" si aumenta la luminosità (controllata in PWM) del LED1, mentre premendo "d" la luminosità stessa diminuisce. Bene, qui si conclude la prima parte del corso su mbed; in attesa di riprendere il discorso (lo faremo nel prossimo fascicolo di Elettronica In) speriamo di aver stimolato sufficientemente la vostra curiosità spingendovi a provare questa interessante scheda embedded.

Per ulteriori dettagli relativi alle istruzioni, alle librerie e alle funzionalità e su come implementarle, è possibile fare riferimento alla sezione "Handbook" (<https://mbed.org/handbook/Homepage>), al "Cookbook" (<https://mbed.org/cookbook/Homepage>) e al forum (<https://mbed.org/forum/>).

Nella prossima puntata illustreremo ulteriori dettagli del codice e presenteremo altri esempi pratici che mostreranno come sia possibile effettuare il porting (cioè la portabilità) del codice di Arduino in ambiente mbed e in che modo utilizzare gli shield di Arduino con i nostri dispositivi basati sui processori Cortex NXP.

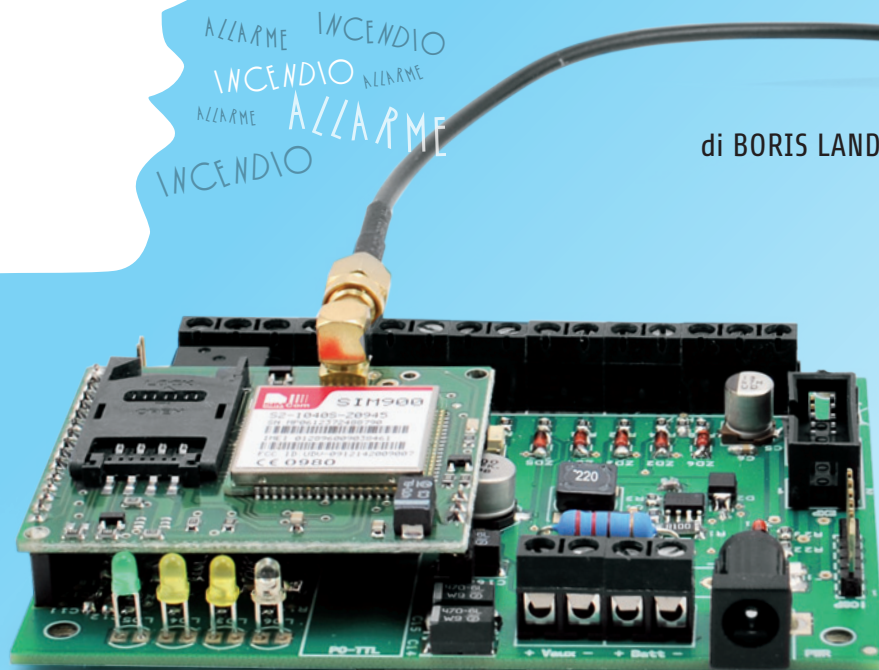
Riferimenti

- [1] <http://mbed.org/handbook/Founders-interview>
- [2] <http://www.arm.com/products/processors/cortex-m/index.php>
- [3] http://mbed.org/users/alexan_e/notebook/pinout-of-the-mbed-lpc11u24/
- [4] <http://mbed.org/users/fraserphillips/notebook/mbed-gpio-pin-table/>
- [5] <http://mbed.org/handbook/mbed-Microcontrollers>
- [6] <http://mbed.org/handbook/mbed-NXP-LPC11U24>
- [7] <http://mbed.org/handbook/mbed-NXP-LPC1768>,
- [8] <http://mbed.org/blog/entry/mbed-Windows-7-and-USB-cables/>
- [5] <http://mbed.org/handbook/USBSerial>
- [9] <https://mbed.org/handbook/SerialPC>
- [10] <https://mbed.org/handbook/Windows-serial-configuration>
- [11] <https://mbed.org/handbook/Terminals>

COMBINATORE GSM CON FUNZIONE TTS

In caso di attivazione di un ingresso, invia SMS o messaggi vocali a un massimo di otto numeri preventivamente memorizzati. Si programma mediante SMS o un'interfaccia vocale basata su TTS. Seconda puntata e ultima puntata.

di BORIS LANDONI



Ci siamo lasciati, nel fascicolo n° 178, dopo aver descritto caratteristiche e funzioni del combinatore telefonico dotato di TTS, promettendo di riprendere e concludere il discorso sulla gestione da remoto e sulle impostazioni del sistema. Come accennato, il combinato-

re può essere configurato sia al banco, tramite l'apposito software e l'interfaccia USB, sia da SMS e via telefono. La configurazione al banco è totale (cioè consente di impostare tutti i parametri di esercizio) ed è da prediligere quando il combinatore sia disponibile sul posto, mentre

chiaramente da remoto si può ricorrere a quella da telefono o via SMS. Di seguito descriveremo tutte le procedure. Spiegheremo, inoltre, in che modo si comanda il sistema affinché fornisca lo stato dei propri ingressi e imposti le uscite nella maniera da noi desiderata.

Partiamo dicendo che dopo la programmazione, nella Flash del microcontrollore sono già caricati i parametri di funzionamento predefiniti e le frasi di risposta ai comandi, nonché quelle del menu vocale. Diciamo altresì che da telefono o SMS, il

Tabella 1 - Comandi da SMS.

FUNZIONE	COMANDO SMS	VALORE PREDEFINITO	PWD NECESSARIA
Cambio password	PWDxxxx;12345	12345	SI
Memorizzare 1 numero (massimo 8 numeri, massimo 19 caratteri per numero); x è la posizione	NUMx+393355760937;12345	-	solo se la posizione x non è libera
Cancellare un numero	NUMx;12345	-	SI
Verificare i numeri memorizzati	NUM?;12345	-	SI
Reset completo	RES	-	SI
Impostare i numeri cui verranno inviati gli SMS	SMSxxxxxxx:ON	Tutti	SI
Impostare i numeri cui verrà fatta la chiamata vocale	VOCxxxxxxx:OFF	Tutti	SI
Livello logico di allarme alto (presenza tensione in ingresso); x può valere 1 o 2	LIVx:A	Attivo in presenza di tensione	-
Livello logico di allarme basso (assenza tensione in ingresso); x può valere 1 o 2	LIVx:B	Attivo in presenza di tensione	-
Livello logico di allarme variazione	LIVx:V	Attivo in presenza di tensione	-
Richiesta livello	LIV?	-	-
Tempo inibizione ingresso1 (da 00 a 59 minuti)	INI1:mm	5 minuti	-
Tempo inibizione ingresso2 (da 00 a 59 minuti)	INI2:mm	5 minuti	-
Richiesta del tempo di inibizione	INI?	-	-
Azzer tempo di inibizione se l'ingresso1 torna a riposo X=0: non azzerata x=1: azzerata	TIZ1x	0	-
Azzer tempo di inibizione se l'ingresso2 torna a riposo X=0: non azzerata x=1: azzerata	TIZ2x	0	-
Tempo di osservazione ingresso1 (00+59 s)	OSS1:s	1	-
Tempo di osservazione ingresso2 (00+59 s)	OSS2:ss	1	-
Richiesta tempo osservazione	OSS?	-	-
Attivazione uscita in modalità bistabile	OUTx:ON	-	-
Disattivazione uscita in modalità bistabile	OUTx:OFF	-	-
Cambio stato relé monostabile (tempo in secondi da 1+9)	OUTx:s	-	-
Ripristino relé: x vale 1 per avere il ripristino e 0 per non averlo	RIPx	1	-
Interrogazione ripristino	RIP?	-	-
Richiesta stato ingressi e uscite	STA?	-	-
Invio SMS all'accensione: x vale 1 per attivarlo e 0 per disattivarlo	AVVx	0	-
Testo allarme ingresso1 quando è presente tensione (max 100 caratteri)	TIN1A:xxxxxxxxxx	Allarme 1 alto	Multimessaggi non permessi
Testo allarme ingresso1 quando è assente tensione (max 100 caratteri)	TIN1B:xxxxxxxxxx	Allarme 1 basso	Multimessaggi non permessi
Testo allarme ingresso2 quando è presente tensione (max 100 caratteri)	TIN2A:xxxxxxxxxx	Allarme 2 alto	Multimessaggi non permessi
Testo allarme ingresso2 quando è assente tensione (max 100 caratteri)	TIN2B:xxxxxxxxxx	Allarme 2 basso	Multimessaggi non permessi
Testo messaggio accensione (max 100 caratteri; invio solo al primo numero in memoria)	TSU:xxxxxxxxxx	System startup	Multimessaggi non permessi
Richiesta pwd DTMF X=0: richiesta disabilitata x=1: richiesta abilitata	SICx	0	-
Filtro chiamate per gestione DTMF X=0: filtro disabilitato x=1: filtro bilitato	FILx	1	-
Disabilita la risposta per quel multimessaggio	RISP	-	-

sistema può essere configurato sia da numeri presenti nell'apposita lista di 8, sia da altri a patto di inserire una password, di cui parleremo tra breve. Siccome dopo la programmazione il modulo ha la lista vuota, se

non si può procedere al banco, la maniera più semplice per introdurre un primo numero, da cui poi impartire i comandi di memorizzazione degli altri, è l'Easy Setup, di cui abbiamo parlato nella prima puntata.

IL MENU VOCALE

Per semplificare le operazioni di comando e configurazione, questo combinatore implementa un menu vocale che viene presentato quando lo si chiama al numero di cellulare corrispondente alla

SIM inserita nella scheda GSM. Alla chiamata può rispondere sia se proviene da un numero di quelli in lista, sia da qualsiasi numero, ma in quest'ultimo caso richiede di digitare la password usando, allo scopo, la tastiera del telefono. Se non è stata impostata la richiesta della password, si presenta il seguente menu:

1. GESTIONE INGRESSI E USCITE
2. PROGRAMMAZIONE
3. GESTIONE NUMERI
4. VERIFICA SISTEMA

(questo menu viene proposto anche se è stata richiesta la password, una volta ottenuto l'accesso).

Componendo dal telefono il numero corrispondente, si accede al rispettivo sottomenu. Non è necessario attendere che la voce elenchi tutte le opzioni: potete impartire un comando prima, premendo il tasto corrispondente, allorché l'elenco viene interrotto e il comando eseguito.

Il menu e alcune frasi relative alla password non possono essere modificati; invece potete modificare a piacimento le risposte che il sistema fornisce sullo stato degli ingressi e delle uscite; le frasi possono essere impostate solo da SMS, in quanto non è possibile registrarle al telefono; vedremo più avanti, quando parleremo del comando da SMS, come si memorizzano.

Ora vediamo una ad una le voci del menu principale.

GESTIONE INGRESSI E USCITE

Vi si accede premendo il tasto 1 della tastiera del telefono e dà accesso a un sottomenu che raggruppa i comandi per conoscere al telefono lo stato degli ingressi e comandare le uscite; il sottomenu propone le seguenti opzioni:

1. stato ingressi;
2. stato uscite;
3. attiva uscita;
4. disattiva uscita.

Premendo il tasto 1 si ottiene lo stato degli ingressi (il formato è "stato ingressi: 1 alto, 2 basso ecc.") mentre con 2 si ottiene lo stato delle uscite (il formato è "stato uscite: 1 a riposo, 2 attivata, ecc.."). Con 3, la voce sintetizzata del sistema propone di digitare il tasto corrispondente al numero dell'uscita da attivare: premendo il tasto della tastiera del telefono si comanda la relativa uscita. Chiaramente sono disponibili i soli tasti 1, 2, 3, 4, 5, perché il sistema dispone di 5 uscite. Dopo aver esercitato la scelta 3, la voce sintetizzata chiede di definire, premendo un tasto da 1 a 9, il tempo di attivazione dell'uscita, che può essere compreso tra 1 a 9 secondi; se desiderate un tempo maggiore, dovete premere 0, così da attivare l'uscita in modo bistabile, quindi inviare il comando 4 (disattiva uscita) trascorso il tempo desiderato. Infine, con il tasto 4 il sistema propone, mediante la sua voce TTS, di premere il tasto corrispondente all'uscita da disattivare (sono disponibili i soliti tasti 1÷5). Resta inteso che impartendo un comando di attivazione di un'uscita già attiva, lo stesso è inefficace; analogo è il discorso se si va a disattivare un'uscita già a riposo.

PROGRAMMAZIONE

A questo sottomenu si accede premendo, nel menu principale, il tasto 2 della tastiera del telefono; esso propone le seguenti opzioni:

1. programma ingressi;
2. programma uscite;
3. numero tentativi di chiamata.

Premendo il tasto 1 si program-

ma l'attività degli ingressi, cioè si dice al sistema cosa debba considerare allarme (stato alto o basso); anche in questo caso si accede a un sottomenu, nel quale vengono proposte due opzioni:

1. memorizza stato ingressi;
2. tempo di osservazione e di inibizione.

La funzione corrispondente al tasto 1 imposta come normale l'attuale situazione degli input, cosicché il combinatore considererà allarme uno stato logico diverso da quello impostato; utilizzatela se intendete programmare con un solo comando lo stato normale degli input. Se invece volete configurare gli ingressi individualmente, avete a disposizione la modalità di comando da SMS. Dunque, dal sottomenu, premendo il tasto 1, la voce sintetizzata del sistema chiede di premere 1 se si intende impostare gli ingressi come sono attualmente: per esempio, se gli ingressi sono tutti a livello alto eccetto il primo, attivando questa funzione il combinatore rileverà come allarme il livello alto al primo input e quello basso ai restanti quattro ingressi.

Invece, premendo il tasto 2 la voce del sistema propone di scegliere l'ingresso di cui definire i tempi di osservazione e inibizione; la scelta si opera premendo sulla tastiera il tasto numerico corrispondente. Definito l'ingresso, il tempo di osservazione si imposta -in secondi- con un numero di due cifre (se il tempo è minore di dieci secondi bisogna digitare 0 seguito dal numero di secondi voluto). La voce sintetizzata del sistema confermerà la memorizzazione dell'impostazione mediante un messaggio del tipo: "tempo di osservazione di xx secondi memorizzato" (xx è il tempo impostato). Fatto ciò, il si-



stema propone di inserire il tempo di inibizione, cosa che si fa in maniera analogica, con due cifre; premuti i tasti corrispondenti, la voce sintetizzata confermerà l'avvenuta memorizzazione con un messaggio del tipo "tempo di inibizione di xx secondi memorizzato". La procedura si conclude e la voce sintetizzata del sistema ripropone il sottomenu dal quale è stata avviata.

Vediamo adesso l'opzione 2 di questo sottomenu, tramite la quale si può abbinare un'uscita all'attivazione di un determinato ingresso: accedendo ad essa, la voce sintetizzata propone di premere il tasto corrispondente al numero dell'uscita, quindi quello relativo al numero dell'ingresso la cui attivazione comanderà localmente l'uscita stessa. Passiamo all'opzione 3, che permette di definire quanti tentativi di chiamata il combinatore dovrà fare a seguito del verificarsi di una condizione di allarme ad uno degli ingressi; entrati nell'impostazione, basta premere il tasto (da 0 a 9) corrispondente al numero di tentativi che si desidera vengano fatti. La voce sintetizzata del sistema rispon-

derà con un messaggio del tipo: "verranno eseguiti x tentativi di chiamata" dove x è il numero impostato.

GESTIONE NUMERI

Con questo sottomenu si può gestire la rubrica degli 8 numeri cui il combinatore invia SMS ed effettua chiamate; vi si accede premendo, nel menu principale, il tasto 3 del telefono. Il sottomenu si compone di due opzioni:

1. inserisci o cancella numero;
2. funzioni numero.

Con la prima viene chiesto di premere il tasto (1÷8) corrispondente alla posizione della rubrica su cui operare, quindi la voce sintetizzata invita ad inserire il numero da memorizzare seguito da #, ovvero a premere asterisco per cancellare il numero presente in quella posizione di memoria. Il sistema risponderà con "numero xxxx memorizzato" nel primo caso e con "numero cancellato" nel secondo.

Con l'opzione 2 si abbina a ciascun numero una serie di azioni: scelta l'opzione (con il tasto 2 della tastiera del telefono) viene chiesto di premere il tasto (1÷8) corrispondente alla posizione della rubrica su cui operare, quindi la voce sintetizzata invita

a definire se in caso di allarme, al numero debba essere o meno inviato un SMS: 1 significa SI, 0 vuol dire NO. La voce sintetizzata del sistema risponderà confermando l'una o l'altra impostazione, dopodiché chiederà di scegliere se allo stesso numero, in caso di allarme, dovrà essere fatta (scegliere 1) o no (premere 0) una chiamata. Anche in questo caso il sistema confermerà con un apposito messaggio vocale l'impostazione fatta.

VERIFICA SISTEMA

Con questa voce del menu principale si può chiedere al sistema se è in allarme o a riposo; oltre a questa informazione, il combinatore comunica la potenza del segnale GSM, dato utile a comprendere se è regolarmente in collegamento o se possono esserci problemi di connessione.

COMANDI DA SMS

Visto come il combinatore si gestisce da telefono tramite il menu vocale, possiamo passare al comando da SMS, più articolato e completo.

I comandi da SMS sono di due tipi: quelli di configurazione e quelli operativi, che servono ad attivare e disattivare ognuna delle uscite presenti, in modo sia impulsivo (definendo, entro i limiti previsti, il tempo di attivazione) sia bistabile, nonché ad interrogare l'unità per conoscere la condizione delle uscite (o degli ingressi).

Quest'ultima funzione è indispensabile quando si sia dimenticato l'ultimo comando inviato, ovvero per accertarsi che un SMS di comando abbia sortito l'effetto desiderato; nel caso del comando tramite bitoni, la conferma arriva immediatamente tramite un messaggio vocale udibile nella cornetta del telefono da cui

si chiama. Vediamo brevemente i comandi di configurazione, meglio descritti nella **Tabella 1**: sono LIVx:A per impostare l'allarme quando l'ingresso corrispondente (x) riceve tensione oltre i 3 volt (se si tratta di IN5) o è aperto (se si tratta di IN1÷IN4), LIVx:B per determinare l'allarme in caso di assenza di tensione e LIVx:V per far sì che il circuito rilevi allarme quando all'ingresso indicato si verifica un cambiamento di stato.

Ogni impostazione determina la risposta del telecomando, sotto forma di SMS contenente il testo "Allarme!! Ingresso 1 alto", qualora sia presente una tensione sull'ingresso 1, oppure "Allarme!! Ingresso 1 basso", in assenza di tensione sull'ingresso 1. Lo stesso tipo di SMS, ovviamente, viene inviato in caso di variazione del livello agli altri ingressi.

Onde evitare continui invii di segnalazioni di allarme, abbiamo previsto un tempo di inibizione

della lettura dopo la segnalazione del primo allarme; tale tempo (l'impostazione predefinita è 5 minuti) si definisce mediante l'invio del comando INI1 per l'input 1 e INI2 per l'ingresso 2, ecc.; INI? consente di interrogare il combinatore per sapere se e per quali ingressi è stato impostato un tempo di inibizione e a quanto corrisponde. Scaduto l'intervallo di inibizione, un nuovo cambiamento dello stato logico determina l'invio del messaggio di allarme.

Quando viene rilevato l'allarme agli ingressi, il sistema può inviare messaggi di testo diversi a seconda di qual è il livello riscontrato; i messaggi del caso si impostano con i comandi da SMS TIN1A (per la presenza di tensione) e TIN1B (assenza di tensione) per l'input 1 e TIN2A, TIN2B per l'ingresso 2.

Se bisogna tenere sotto controllo sensori che forniscono segnali variabili frequentemente, per ricevere avvisi realistici circa gli

avvenimenti in corso può essere necessario ignorare il tempo di inibizione. Il combinatore prevede la possibilità di disattivare provvisoriamente, ingresso per ingresso, il tempo di inibizione impostato mediante i comandi di seguito specificati: il comando TIZ1x permette di azzerare il tempo d'inibizione se l'ingresso 1 torna a riposo; x vale 0 per azzerare e 1 per non azzerare. Analoghi sono i comandi per gli altri ingressi.

Vediamo adesso i messaggi di comando delle uscite, che sono OUTx:ON (x è il numero del relé) per attivare e OUTx:OFF per disattivare; impartendo OUTx:s l'uscita x funziona in modo impulsivo e cambia il proprio stato rispetto all'attuale per un tempo, espresso in secondi, pari ad s (il tempo è tra 1 e 9 secondi).

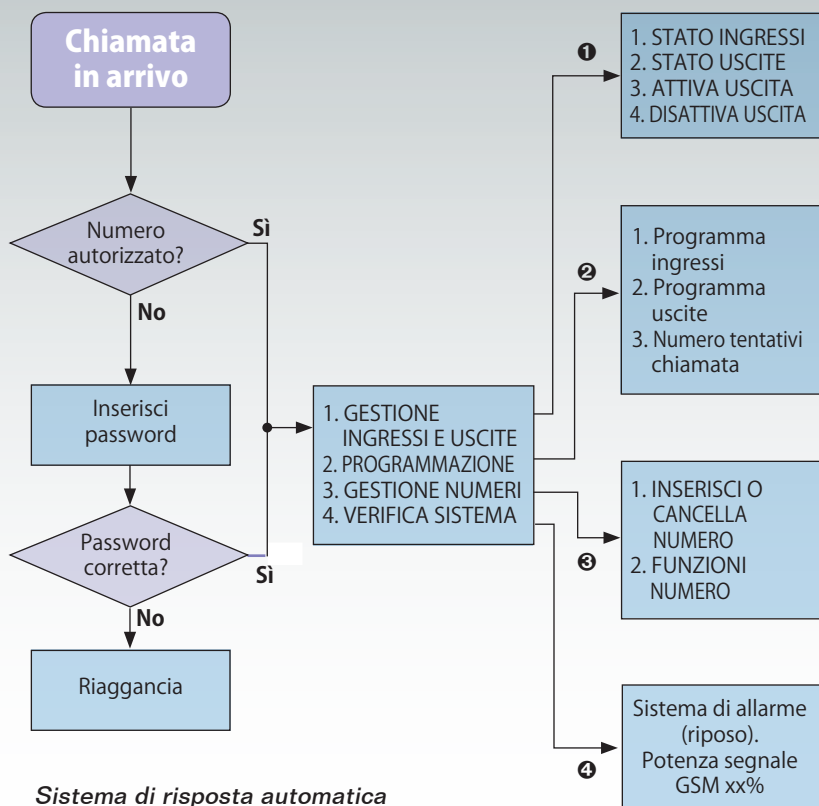
Con il comando STA? possiamo richiedere al combinatore

Anche l'iPhone ha il suo antifurto

Sistemi d'allarme e antifurto ne esistono per tutto: auto, casa, azienda, ma anche per oggetti di culto come l'iPhone, preda di ladri e ladroncoli metropolitani. Per ritrovare lo smartphone perduto esistono però App come Prey Anti-Theft, adattabile anche a tutti i dispositivi basati su iOS, Android (si scarica da Google Play) Mac Windows, Linux. Il funzionamento è semplice: dopo aver installato Prey Anti-Theft ed averlo collegato al proprio account, se si perde lo smartphone basta andare sul sito <http://preyproject.com> ed attivare

l'antifurto impostando il cursore su "missing". Appena lo smartphone è raggiungibile dalla rete 3G, riceve l'ordine di generare dei report periodici contenenti foto e posizione GPS, quindi di inviarli al sito, dal quale potranno poi essere consultati e da dove partiranno notifiche via e-mail all'indirizzo di posta elettronica del proprietario. In pratica, Prey Anti-Theft non fa altro che inviare fotografie georeferenziate scattate dalla fotocamera dello smartphone e inviarle al sito per tracciare il percorso e la posizione dell'apparecchio smarrito.





Sistema di risposta automatica e del menu vocale del combinatore GSM.

l'attuale condizione di ingressi ed uscite.

Il combinatore dispone della funzione di ripristino dello stato dell'uscita a relé dopo una perdita di tensione; essa serve ad evitare che, in caso di spegnimento accidentale, alla riaccensione il relé perda l'impostazione data dall'utente mediante i comandi remoti. L'attivazione del ripristino si effettua con RIP1 e la disattivazione con RIP0; in ogni momento è possibile sapere se la funzione di ripristino dello stato dei relé è attiva: basta impartire il comando RIP?.

Altra funzione è l'invio di un SMS quando il circuito riceve alimentazione; si inserisce con AVV1 e si disattiva con AVV0. Il comando che permette di definire il testo del messaggio è TSU:xxxxxx, dove al posto delle x va scritto il testo.

Ora è il caso di analizzare i comandi da SMS che impostano

il controllo da DTMF: il primo di questi attiva o disattiva il filtro sulle chiamate entranti; si impartisce con un messaggio contenente il testo FILx, dove al posto della x va 1 se si intende impedire l'accesso (la risposta del telecontrollo alle chiamate entranti) ai numeri che non siano quelli in lista, ovvero 0 se si desidera permettere che il telecontrollo risponda alle chiamate provenienti da qualsiasi numero telefonico.

Il secondo è quello che permette di impostare il sistema affinché, dopo aver risposto alle chiamate entranti, richieda la password di accesso al comando DTMF: si impartisce inviando un SMS con il testo SICx, dove al posto di x si scrive 0 se si desidera che l'accesso al comando sia libero ed 1 se, invece, si vuole che, una volta ottenuta la risposta dal telecontrollo, si esiga che il comando DTMF possa essere svolto solo

dalle persone in possesso della password.

Tutti i comandi di configurazione della modalità di telecontrollo (gestione a distanza delle uscite) e di teleallarme (combinatore) vengono accettati dal dispositivo a condizione che provengano da uno dei numeri telefonici memorizzati nell'apposita lista. Naturalmente, è prevista una soluzione per operare sul sistema quando lo si avvia per la prima volta, condizione nella quale, evidentemente, non avendo numeri memorizzati non riconoscerebbe alcun utente: i messaggi inerenti alla memorizzazione di indicativi telefonici nella lista vengono riconosciuti ed i corrispondenti comandi eseguiti, anche se provenienti da numeri di telefono sconosciuti, a condizione che contengano la password. Notate che alcuni comandi SMS devono essere completati dalla password anche se provenienti da numeri già memorizzati nella lista: si tratta di quelli riguardanti la richiesta dei numeri, la cancellazione di posizioni contenenti numeri e la sostituzione di un numero esistente (cioè quando nella posizione in cui si vuole scrivere è già presente un numero).

La password è un numero di cinque cifre; dopo la prima accensione o a seguito di un reset del sistema, viene caricata quella predefinita: 12345. È possibile cambiarla, in ogni momento, tramite un apposito comando via SMS, ovviamente corredato da password (che deve essere quella corrente); questo comando è PWDxxxxx;pwd e in esso xxxxx rappresenta la nuova password (numerica, di cinque cifre) e pwd indica la password in uso.

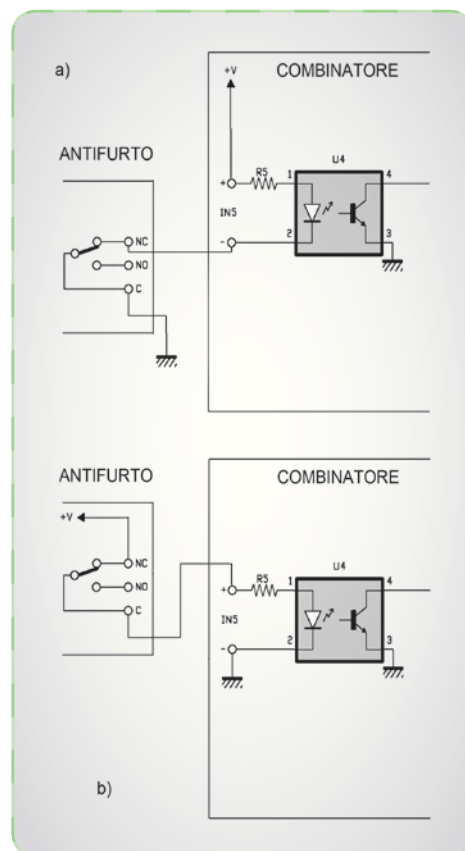
Completa il set di comandi il reset: inviato via SMS da uno dei telefoni in lista, riporta il tele-

controllo nelle condizioni iniziali, nel senso che cancella la memoria dei numeri e le modifiche alle impostazioni riguardanti la modalità di funzionamento degli ingressi e delle uscite. Essendo un comando di notevole importanza, viene considerato solo se giunge corredato della password. Ci sono poi i comandi per gestire la lista dei numeri telefonici: NUMx+39nnnnnnnnnn;pwd permette la memorizzazione nel dispositivo di un numero telefonico (massimo 8 numeri da 19 caratteri ciascuno); *x* rappresenta la posizione, nella lista, occupata dal numero, nnnnnnnnnn il numero telefonico preceduto da prefisso internazionale (+39 per l'Italia) e *pwd* la password in uso. Se il comando SMS di memorizzazione di un numero viene inviato da un cellulare già in lista, è necessario introdurre la password solamente quando si tenta di sovrascrivere una posizione già occupata; invece se proviene da un telefono estraneo, la password è sempre richiesta. Il comando NUMx;pwd permette di rimuovere un numero telefonico dalla lista; *x* rappresenta la posizione occupata dal numero nella lista e *pwd* la password in uso. Il comando NUM?;pwd permette di richiedere la lista dei numeri telefonici attualmente memorizzati nel dispositivo; *pwd* è la password in uso. Questo comando va sempre impartito con la password perché trasmette a chi l'ha inviato la lista di tutti i numeri abilitati ad operare con il combinatore, esponendo le persone che gli corrispondono a possibili rischi. È quindi importante che la lista non possa essere richiesta (anche per ragioni di riservatezza) da tutti gli utenti ma solo da chi gestisce il sistema. Il combinatore può inviare degli

SMS al telefono delle persone che devono essere avvertite di una variazione di stato degli ingressi. I comandi per l'abilitazione o la disabilitazione di tale funzione (valida solo per i numeri telefonici presenti nella lista) sono SMSx:ON, che permette al numero occupante la posizione *x* di ricevere gli SMS di notifica stato ingressi, ovvero SMSx:OFF, che esclude il numero occupante la posizione di rubrica specificata (*x*) dalla ricezione di SMS di notifica dello stato ingressi. Oltre ad inviare SMS, il dispositivo può effettuare chiamate verso il telefono delle persone (il cui numero è presente in lista) che devono essere avvertite di una variazione di stato degli ingressi; la chiamata vocale permette di richiamare l'attenzione, sul fatto che si è verificato un evento, più rapidamente di quanto non si otterrebbe dagli SMS, i quali possono giungere con un certo ritardo. Il comando VOCx:ON permette, al numero occupante la posizione specificata, di ricevere la chiamata di notifica stato ingressi; *x* rappresenta la posizione, della lista, occupata dal numero (o dai numeri). Il comando VOCx:OFF esclude il numero occupante la posizione *x* in rubrica dalla ricezione delle chiamate di notifica dello stato degli ingressi. Per impostazione predefinita (dopo il caricamento

del firmware nel microcontrollore o a seguito di un reset totale) è previsto che tutti i numeri telefonici registrati nelle prime otto posizioni della lista (naturalmente, a patto che siano memorizzati) ricevano la notifica di allarme ingressi, tramite SMS e breve chiamata vocale. Il combinatore permette di definire per quanto tempo deve permanere la condizione di allarme ad un determinato ingresso affinché il circuito invii le notifiche del caso; l'impostazione si effettua con il comando SMS OSSx:ss, nel quale *x* è il numero dell'ingresso ed *ss* rappresenta il tempo espresso in secondi. La durata del tempo di osservazione può essere scelta fra 1 e 59 secondi. Il valore predefinito in fabbrica e dopo ogni reset totale è pari a 1 secondo. Il comando OSS? permette di

Comando a caduta di positivo dell'ingresso optoisolato da parte di una centrale dotata di uscite a relé: in a) si alimenta il fotoaccoppiatore del combinatore mediante l'alimentazione principale e si chiude a massa il punto - mediante l'uscita della centrale, mentre in b) è quest'ultima a fornire alimentazione propria.



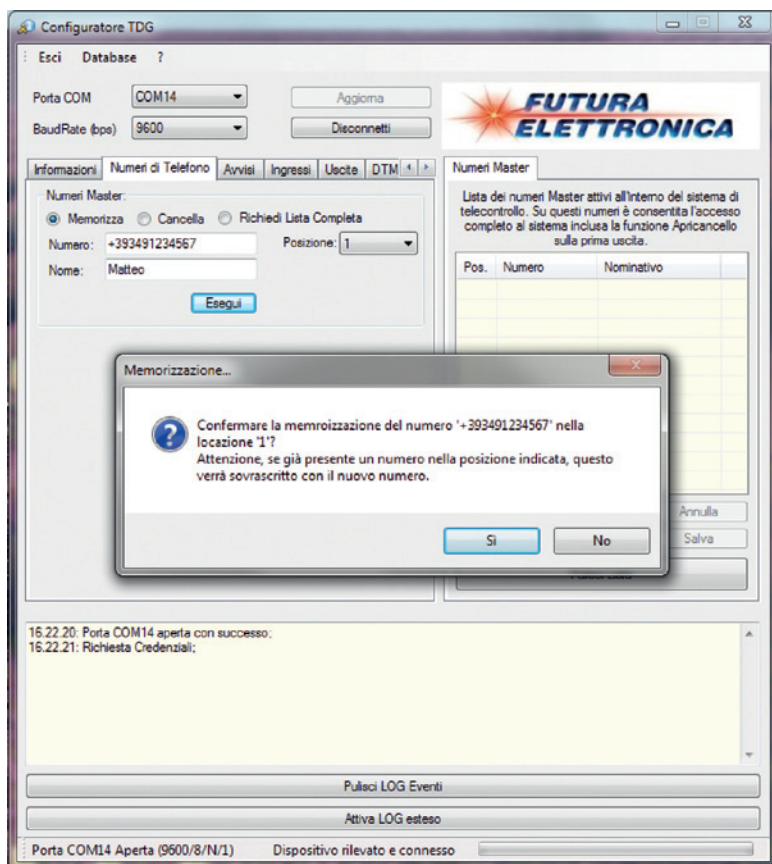


Fig. 1 - Finestra per la memorizzazione de numeri di telefono.

richiedere l'attuale impostazione del tempo di osservazione relativo agli ingressi.

Diamo ora uno sguardo ai comandi da SMS che ci consentono di definire a piacimento i messaggi vocali che il combinatore deve far udire agli utenti in lista quando si verifica un evento che prevede la notifica a voce: ad esempio un livello diverso da quello normale. I comandi sono due: TIN1B:xxxx permette di definire il messaggio che il dispositivo invia quando all'ingresso 1 viene applicato un livello logico basso e in esso xxxx rappresenta il testo del messaggio che si vuole scrivere (massimo 100 caratteri compresi gli spazi). Nel messaggio non può essere utilizzato il carattere "punto e virgola" (;) e le lettere devono necessariamente essere scritte tutte in maiuscolo. Il messaggio predefinito è: ALLARME 1 BASSO.

Per gli ingressi 2, 3, 4, vale un

discorso analogo e il comando è, rispettivamente, TIN2B:xxxx TIN3B:xxxx, TIN4B:xxxx, TIN5B:xxxx. Per l'ingresso 5, il livello basso è la mancanza di tensione o tensione insufficiente all'ingresso dell'optoisolatore. Il comando che determina la notifica al presentarsi del livello alto (per l'IN5, tensione sull'ingresso del fotoaccoppiatore) è TIN1A:xxxx, dove xxxx rappresenta il testo del messaggio (massimo 100 caratteri compresi gli spazi; nel messaggio non può essere utilizzato il carattere "punto e virgola" e le lettere devono necessariamente essere scritte tutte in maiuscolo). Il messaggio predefinito è ALLARME 1 ALTO. Per gli ingressi 2, 3, 4, vale un discorso analogo e il comando è, rispettivamente, TIN2A:xxxx TIN3A:xxxx, TIN4A:xxxx, TIN5A:xxxx. Resta inteso che il combinatore trasmette di volta in volta il

solo messaggio interessato: per esempio, se l'ingresso 1 deve normalmente essere a livello alto, in caso di allarme (livello basso) il combinatore telefonerà e riprodurrà il messaggio ALLARME 1 BASSO.

È consigliabile far riprodurre messaggi che corrispondano intuitivamente alla situazione dei dispositivi o sensori collegati agli input: per esempio, se all'ingresso 2 è collegato un P.I.R., il messaggio potrà essere: INTRUSIONE IN CORSO. Se sarà collegato un sensore di fumo, potrete impostare il testo: SVILUPPO DI INCENDIO.

Tra i comandi SMS, ne è previsto uno che permette di conoscere da SMS la qualità del segnale GSM in percentuale (da 1 a 100%) ricevuto dal SIM900: il comando è QUA? e sortisce un effetto analogo al 4 impartito via DTMF dal menu vocale.

Concludiamo il discorso ricordando che a seguito di un comando o di un'interrogazione, il combinatore risponde con un SMS di conferma esecuzione o contenente informazioni relative alle impostazioni. Il combinatore accetta SMS multipli, ossia contenenti più di un comando; in questo caso i singoli comandi devono essere separati ognuno da quello successivo mediante una virgola. Siccome un comando multiplo produrrà più SMS di risposta, per evitarli è necessario disattivare la risposta, cosa che si ottiene inserendo all'inizio dell'SMS pluricomando il testo RISP. Ogni comando deve essere scritto con lettere MAIUSCOLE e senza inserire spazi vuoti tra le voci che lo compongono.

LA CONFIGURAZIONE DA COMPUTER

È possibile effettuare la configurazione del combinatore da PC

mediante un programma che funziona in ambiente Windows XP/2000/Vista/7 - versione a 32 bit o 64 bit. Il file corrispondente si trova in formato compresso; fatta l'installazione, nel menu *Programmi* trovate il gruppo *Futura Elettronica*, all'interno del quale vedrete la voce di menu *Configuratore TDG*.

Il software è pensato, oltre che per il combinatore telefonico, per i nostri telecontrolli TDG133, TDG134, TDG139, TDG140 ed è in grado di riconoscere automaticamente il circuito che viene collegato all'USB del computer; in base al dispositivo connesso, il software adatta la propria interfaccia grafica, mostrando nella finestra di dialogo principale le sole schede attinenti.

Per effettuare la configurazione è necessario alimentare e collegare

il combinatore al PC quindi selezionare la porta di comunicazione "COM" (in alto a sinistra) creata. In caso di necessità, tramite il pulsante "Aggiorna", è possibile effettuare l'aggiornamento della lista delle porte COM disponibili nel PC in uso. Successivamente, facendo clic sul pulsante "Connetti" si avvia la connessione; da questo momento il combinatore risulta gestibile mediante il proprio PC (la modalità PC viene segnalata dall'accensione fissa del LED LD3). Se il dispositivo è stato collegato al PC prima che sia stata eseguita la memorizzazione di almeno un numero, nella finestra di log (in basso) apparirà il messaggio "Attesa Chiamata per memorizzazione numero Master". Per proseguire con la configurazione è necessario per-

tanto eseguire la chiamata per la memorizzazione del numero (procedura Easy Setup). L'esecuzione dell'operazione verrà confermata tramite messaggio che apparirà nella finestra di log. Nella scheda *Informazioni* troviamo riepilogate le caratteristiche del telecontrollo in fase di configurazione, ossia il modello (nel nostro caso TDG145), la versione del firmware e l'IMEI del modulo SIM900.

La sezione *Numeri Master* riepiloga i numeri della lista (8 al massimo) abilitati ad effettuare la configurazione ed impartire comandi via SMS.

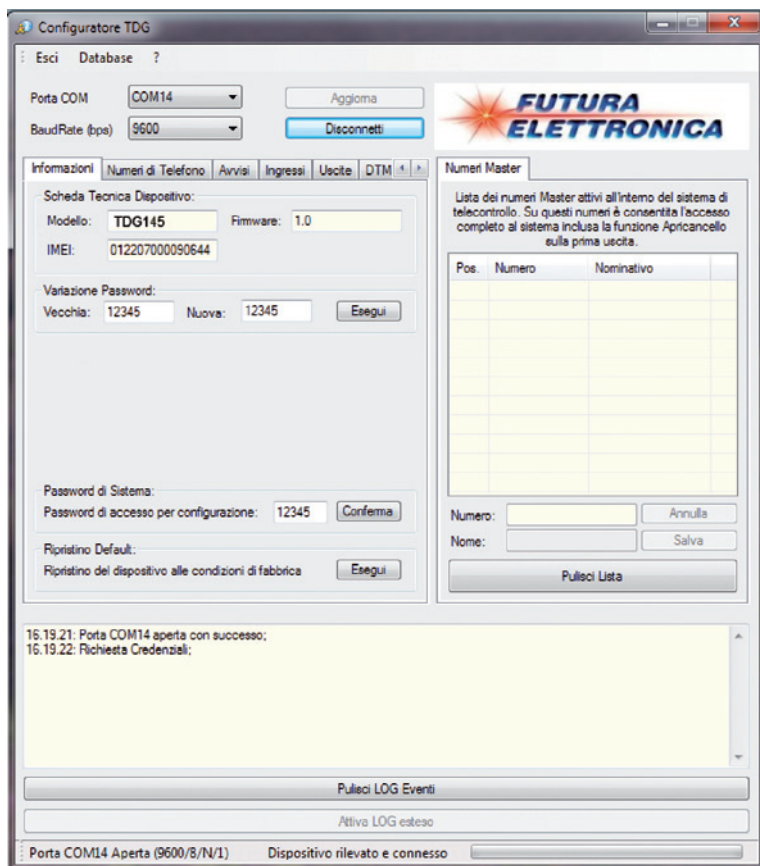
Le caselle *Porta COM* e *Baud Rate* riepilogano le informazioni sulla COM virtuale aperta per la connessione USB.

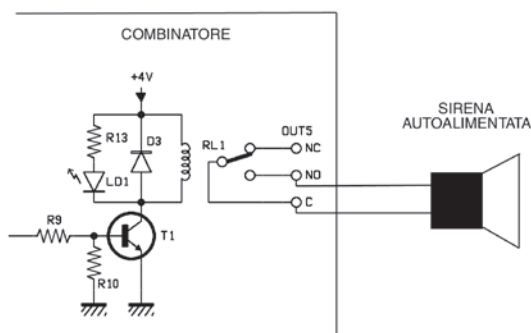
Nella barra dei menu troviamo due comandi: *Esci* permette di abbandonare il programma, mentre *Database* apre la finestra *Rubrica*, dove sono esposti i numeri di telefono memorizzati; in questa finestra i pulsanti *Pulisci Lista* e *Aggiorna Lista* consentono rispettivamente di cancellare la rubrica o inviare al circuito le modifiche alla rubrica.

La finestra principale (Fig. 1) mostra la memorizzazione di un numero di quelli abilitati al comando; notate che si può definire (mediante la casella *Posizione*) in quale posizione salvare il numero, oltre che il nome della persona che possiede il telefono di cui memorizziamo il numero. Con la scheda *Avvisi* è possibile specificare, per ciascun numero di telefono inserito nella lista dei numeri Master, se in caso di allarme deve ricevere l'avviso mediante SMS o con una telefonata.

La selezione si effettua apponendo il segno di spunta sulle caselle corrispondenti, fermo restando che il combinatore

Fig. 2
Informazioni relative al dispositivo connesso.





Collegamento all'uscita a relé di una sirena autoalimentata.

funziona anche senza attivare tale funzione. La scheda *Ingressi* permette di verificare o modificare la logica degli input (cioè ogni ingresso deve considerarsi in allarme quando è alimentato o privato dell'alimentazione, oppure solo in presenza di una variazione di stato) il tempo di inibizione tra un livello che determina allarme e il successivo, oltre al tempo di osservazione, ovvero l'intervallo temporale per cui la condizione di allarme deve permanere affinché venga considerata tale. La scheda *Uscite* consente invece l'impostazione del modo di funzionamento dei relé; dall'apposita sezione (Controllo Manuale) possiamo attivare manualmente i relé, in modo sia bistabile che impulsivo (in tal caso nel menu a tendina *Tempo* possiamo scegliere l'intervallo di attivazione) per verificare il corretto funzionamento loro e dell'intero sistema. Nella stessa scheda è presente la sezione *Ripristino* all'avvio, da cui possiamo decidere se il telecomando deve ripristinare lo stato delle uscite dopo un blackout, oppure lasciarle tutte a riposo. E veniamo ora alla scheda forse più importante: *DTMF*; da qui possiamo impostare gli intervalli di attivazione dei relé di uscita e decidere se il combinatore deve eseguire i comandi DTMF incondizionatamente o solamente se il chiamante digita prima la password di accesso. Una terza sezione della scheda

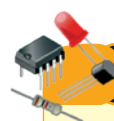
DTMF è quella che permette di attivare o disattivare il filtro sulle chiamate (*Filtro Chiamate DTMF*).

Bene, vediamo adesso la scheda che riguarda gli SMS inviati dal telecomando in risposta ai vari comandi o quelli trasmessi a seguito di allarme: si chiama *Testo SMS* e contiene due sezioni; dalla prima (*SMS Startup Modulo*) si può definire il testo del messaggio inviato dal sistema ogni volta che viene alimentato e attivare o disattivare la funzione corrispondente. Dalla sezione *Personalizza SMS di allarme* si può, invece, definire ingresso per ingresso sia il testo dei messaggi inviati, sia il livello o la condizione (stato basso, alto, cambio di stato logico) che ne determina l'invio.

INSTALLAZIONE DEL SISTEMA

Una volta configurato al banco, il combinatore vocale si applica a qualsiasi impianto di allarme dotato di un'uscita che si attiva (in caso d'allarme o anomalia) fornendo tensione, ovvero togliendola: occorre impostare l'ingresso IN5 con il comando *SMS LIVx:A* nel primo caso e con *LIVx:B* nel secondo. Disponendo di un'uscita a contatto, si possono usare gli IN1÷IN4 con l'impostazione *LIVx:B*. Vista la nutrita serie di ingressi e uscite, il sistema può operare come una centrale di allarme. Ogni ingresso può essere comandato con un livello di tensione

o attraverso un relé che fornisca o tolga alimentazione: i primi quattro input accettano livelli TTL (avendo il pull-up, sono ideali per il comando mediante contatti), mentre quello optoisolato si comanda con 4÷30 Vcc. Se collegato ad un antifurto, il combinatore può essere provvisto di antisabotaggio, collegando uno degli ingressi all'alimentazione delle sirene a caduta di positivo, ovvero alla linea di alimentazione principale proveniente dall'antifurto stesso: in questo caso, se vengono tagliati i cavi che dall'uscita dell'antifurto portano al combinatore, questo rileva allarme e può inviare gli SMS o fare le telefonate (la batteria gli permette di funzionare). L'uscita a relé può essere abbinata ad un ingresso di allarme tramite il comando da DTMF e con essa si può controllare una sirena locale. ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il combinatore è disponibile già montato e collaudato (cod. TDG145) al prezzo di 112,00 Euro; il dispositivo comprende le due board e l'antenna completa di cavo; non sono inclusi batteria, alimentatore, contenitore e SIM. La board GSM (codice TDGGSM_900TTS) è anche disponibile separatamente al prezzo di 56,00 Euro così come il modulo GSM codice SIM900TTS che costa 36,00 Euro. Il converter USB/seriale FT782M che consente di utilizzare un PC per la programmazione del combinatore costa 15,00 Euro. 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>

NIWEEK 2013, BENVENUTI IN UN MONDO PROGRAMMABILE

di ARSENIO SPADONI

Cyber-Physical Systems Across Industries and Applications



Quattro giorni con i protagonisti, la tecnologia e le applicazioni di National Instruments, per assaporare in anticipo un pezzetto di futuro.

Nonostante che da quest'anno l'intero evento venga trasmesso in streaming, partecipare di persona all'annuale Conference di National Instruments riserva delle emozioni e delle sensazioni che la diretta video non potrà mai offrire. Parlare con i progettisti dei sistemi elettronici più

avanzati e con gli scienziati e gli ingegneri che li utilizzano nelle loro ricerche, toccare con mano l'aerorazzo che tenterà di stabilire il record mondiale di velocità su terra o chiacchierare con i protagonisti delle keynote al termine dei loro interventi, è un'esperienza unica. Sembra

veramente di essere su un treno che viaggia verso il futuro, dove tutti parlano lo stesso linguaggio e hanno lo stesso obiettivo: utilizzare la tecnologia per migliorare la qualità della vita in tutti i campi, dalla salute all'ambiente, dall'istruzione ai trasporti. Tecnologia che National Instruments mette a disposizione di tecnici, ingegneri e scienzia-

ti per accelerare lo sviluppo e la messa a punto di nuove applicazioni.

NIWeek è l'occasione per conoscere questa tecnologia, ma anche per immaginare cosa ci riserverà il futuro.

Tra le persone in grado di immaginare il futuro c'è sicuramente il Dr. James Truchard, che fin dalla nascita di National Instruments ne ispira le strategie. E

Applications



Rapid
Control Prototyping

È alle porte, secondo Dr. T., una quarta rivoluzione industriale basata sull'interazione del mondo fisico con le tecnologie emergenti, dall'Internet delle cose, ai Big data. Il Cyber-Physical Systems (CPS), questa la definizione del nuovo paradigma, trasformerà interi settori industriali, inclusi energia, trasporti, sanità, produzione e agricoltura. Per affrontare questa nuova sfida c'è urgente bisogno di sviluppare metodi di progettazione in grado

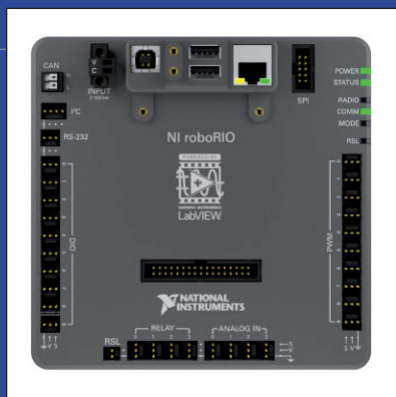
di offrire in tempo reale prestazioni adeguate alle nuove esigenze. Il Graphical Systems Design, la piattaforma hardware e software di National Instruments, è sicuramente lo strumento più adatto per realizzare il Cyber-Physical Systems, in modo particolare con le nuove soluzioni software-designed. La sfida per il futuro è riuscire col Graphical Systems Design a fare per il Cyber-Physical Systems ciò che i PLC hanno fatto per i con-

trolli industriali. L'importanza di una piattaforma comune, come del resto ha dimostrato iOS, è fondamentale in sistemi così complessi dove il volume dei dati può raggiungere livelli elevatissimi e dove è necessario integrare sensori, cloud e controlli real-time. Concetto che è stato ripreso da Eric Starkloff che ha evidenziato come il nostro mondo stia diventando un mondo programmabile, come lo ha definito il



anche quest'anno, nella keynote introduttiva di NIWeek 2013, il Dr. Truchard ha offerto non solo la sua personale visione del futuro dell'automazione industriale e dei sistemi di misura e controllo, ma un quadro più generale dell'impatto che l'evoluzione delle tecnologie elettroniche, informatiche e robotiche avrà sul nostro modo di operare e di produrre.





magazine Wired nel suo ultimo numero, un mondo con una combinazione di sensori di ogni tipo e piattaforme software in grado di aggregare e distribuire le informazioni.

Una tendenza che offre grandi opportunità per migliorare il nostro modo di vivere e lavorare e che rappresenta un'opportunità da cogliere per ingegneri e sviluppatori.

I nuovi prodotti presentati in anteprima da National Instruments a NIWeek si ispirano a questi concetti, offrendo la massima programmabilità. Dopo il VST presentato l'anno scorso per il settore RF (il primo Vector Signal Transceiver completamente riprogrammabile da software), quest'anno è toccato alla piattaforma CompactRIO seguirne l'esempio col modello 9068 che, proprio per questo motivo, possiamo definire "software-designed controller."

Completamente ridisegnata con l'impiego di un SoC Xilinx Zynq-7020 con processore dual-core ARM Cortex A9 e FPGA della serie 7, la nuova versione mantiene la piena compatibilità con la precedente piattaforma CompactRIO aumentando di 4 volte la potenza del processore e implementando per la prima volta un sistema operativo basato su Linux che garantisce una maggior flessibilità sia con l'utilizzo di LabVIEW Real-Time che con le applicazioni in C/C++ e che consente di accedere alla più vasta Community di sviluppatori al mondo. Ciò rende

questo prodotto il dispositivo più "aperto" mai realizzato da NI.

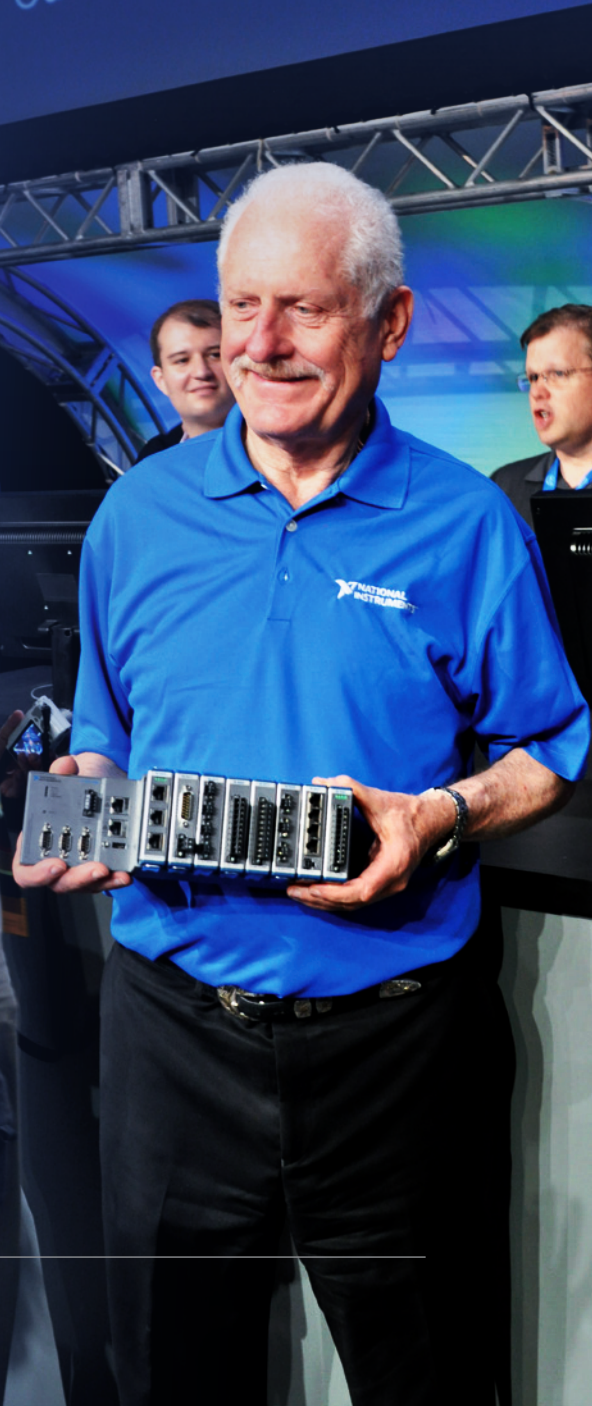
Utilizzando la stessa piattaforma hardware e lo stesso ecosistema, National Instruments ha presentato anche NI myRIO, una scheda didattica completa che abbraccia sia l'insegnamento, sia il lavoro di progettazione e che culminano entrambi nella capacità dello studente di "fare ingegneria" nel mondo reale. NI myRIO è costruito attorno alla tecnologia Reconfigurable I/O (RIO), ma con un focus educativo.

Tramite NI myRIO gli studenti progetteranno sistemi sofisticati già a scuola ed entreranno nel mondo dell'industria con il vantaggio di avere maturato un'esperienza con gli stessi strumenti che troveranno sul posto di lavoro.

L'attenzione al mondo scolastico è apparsa ancora più evidente il terzo giorno con la presentazione di roboRIO una scheda embedded basata sulla stessa architettura Zynq utilizzata nella nuova versione di CompactRIO e in myRIO, e che dovrebbe aiutare gli studenti nella realizzazione di nuovi e innovativi robot.

Uno strumento che ha anche lo scopo di aiutare gli studenti a superare quel divario tra istruzione meramente teorica ed esperienza pratica, oltretutto con una piattaforma che gli studenti troveranno nel mondo reale, una volta terminato il corso di studi. È forse questa la novità più im-

portante che è emersa da questa edizione di NIWeek, l'offerta di una tecnologia (e dell'ecosistema di sviluppo) molto simile sia per l'ambiente accademico che per il mondo industriale, una scelta sicuramente vincente. ■





20-Jun'13 15:05

Laboratorio



Realizziamo un valido strumento low-cost da banco in grado di misurare con precisione la frequenza di segnali analogici BF fino a 10 MHz, TTL e CMOS fino a 50 MHz, ma anche componenti a radiofrequenza fino a 1,1 GHz. Prima puntata.

FREQUENZIMETRO DIGITALE A MICROCONTROLLORE

..... del Prof. MICHELE MENNITI

Da sempre, il primo obiettivo di chi si avvicina al mondo dell'elettronica è l'acquisto di un multimetro digitale (una volta era il tester analogico); ma quando si comincia a metter mano a circuiti più complessi, il tester non basta più. Allora si comincia a guardare con attenzione al mondo della strumentazione professionale o semi-professionale, ma subito ci si imbatte nell'ostacolo più grande: il prezzo. Uno strumento di misura degno di tale nome può costare centinaia, ma anche migliaia o decine di migliaia di euro, cosicché si rinuncia o si è costretti a ripiegare su strumenti a basso costo, che subito mostrano i loro limiti in fatto di sensibilità, precisione, affidabilità. Il progetto che vi presentiamo scioglie il dilemma, perché è uno strumento di misura professionale ad un prezzo accessibile a qualsiasi hobbista.

Prima di vedere com'è fatto il circuito, coloro che volessero rispolverare alcuni concetti di base su frequenza, forme d'onda e loro caratteristiche, possono scaricare il pdf "Un po' di teoria" dalla sezione download del nostro sito www.elettronica.in.it.

STRUTTURA DELLO STRUMENTO

Il frequenzimetro consta di tre schede: *Sezione di alimentazione*, *Sezione ingressi* e *Sezione logica e display LCD*. La prima prende alimentazione dalla rete elettrica e fornisce 5 V per alimentare la logica. La sezione ingressi dispone di tre ingressi separati: BF (per segnali analogici da 1 Hz a 10 MHz), TTL (per segnali digitali da 1 Hz a 50 MHz con duty-cycle variabile dal 20% all'80%) e RF (per segnali generici e di radiofrequenza da 1 MHz a 1,1 GHz). Gli ingressi possono

essere usati uno alla volta; allo scopo vengono commutati da tre relé reed. Infine abbiamo la sezione logica e display LCD, basata su un microcontrollore ATmega328P (quello di Arduino, per intenderci) e sull'uso di una specifica libreria che provvede alla misura della frequenza in arrivo dalla sezione degli ingressi; il firmware analizza la misura e, nel caso degli ingressi BF e TTL, gestisce opportunamente il divisore x10 al fine di ottenere la massima risoluzione possibile. Mediante due pulsanti si selezionano l'ingresso cui applicare il segnale da misurare e la portata (Hz, kHz, MHz, GHz). La misura definitiva

viene visualizzata su un LCD, con un numero di cifre variabili, dipendenti dalla frequenza misurata e dalla portata selezionata.

Abbiamo previsto il montaggio in un contenitore plastico con mascherine in alluminio anodizzato e forature sui coperchi per una corretta ventilazione durante il funzionamento.

Diamo subito uno sguardo allo schema elettrico del frequenzimetro, analizzando una sezione alla volta.

SEZIONE DI ALIMENTAZIONE

Tralasciando il filtro anti-RF posto sulla 220Vac, la tensione 220 Vca viene applicata, tramite il connettore J1, direttamente al primario di TF1, il cui doppio secondario (2x6 V) viene collegato sul PCB in parallelo, in modo da avere a disposizione la massima corrente erogabile dal trasformatore; la tensione risultante viene raddrizzata da un ponte a diodi da 1 A, all'uscita del quale si trova una tensione di circa 9 V filtrata da C1 ed applicata all'ingresso dell'integrato stabilizzatore U1 (LM1117-5.0), dalla cui uscita possiamo prelevare i 5 V necessari al frequenzimetro. Il basso assorbimento dello strumento non fa scaldare particolarmente l'integrato, che quindi può raffreddarsi usando il rame del PCB. C4 filtra la tensione di uscita, mentre C2 e C3, montati molto vicini ai piedini di ingresso ed uscita dell'integrato, prevengono eventuali auto-oscillazioni.

Subito dopo abbiamo inserito un filtro RF per bassa tensione, di tipo CLC, costituito da C5 e C6 e dall'impedenza L1; questo filtro blocca eventuali disturbi RF che l'alimentatore potrebbe captare dall'esterno, ma anche quelli che potrebbero ritornare dalla Sezione ingressi. Sul connettore J2 avremo quindi 5 V ben stabilizzati e filtrati, per alimentare gli altri due circuiti.

Il montaggio di questo PCB è estremamente semplice: conviene iniziare dall'integrato, piegandone i tre pin con una pinza, in modo che una volta infilati nei pad del PCB, il foro presente sull'aletta metallica combaci alla perfezione con quello presente sul PCB; in tal modo sarà possibile far aderire alla perfezione la parte metallica dell'integrato con la superficie ramata del PCB e fissarla con una vite ed un dado. Non servono isolatori, in quanto sia il rame della faccia superiore che la piazzola di quella inferiore, destinati al fissaggio, sono isolati dal piano di massa. Serrata la vite di fissaggio, si procede a saldare i tre pin del regolatore e poi tutti gli altri componenti.

I condensatori e l'impedenza dovranno essere montati il più vicino possibile al PCB (per ragioni di immunità all'RF), mentre il ponte a diodi potrà essere posizionato a circa 0,5÷0,7 cm. L'ultimo componente da saldare è il trasformatore; badate di inserire completamente i pin nei loro fori, in modo da far aderire perfettamente il trasformatore al PCB.

Il collaudo di questo circuito consiste nel collegare il morsetto d'ingresso J1 alla rete 220 V ed un multimetro tra i morsetti d'uscita, in portata V DC (fondo scala 10÷20V); appena date corrente, leggerete circa 5 V sul multimetro. Una tolleranza di ±100mV è normale e non creerà alcun problema al funzionamento del frequenzimetro. Notate che il positivo del J2 è chiaramente indicato nella serigrafia.

In Fig. 1 vedete la foto del PCB montato nel contenitore e collegato alla Sezione 220 Vac, di cui parleremo più avanti.

SEZIONE LOGICA E DISPLAY LCD

Questa scheda si occupa di gestire gli ingressi per ricevere, analizzare, misurare e visualizzare la frequenza. Il cuore del circuito è il

CARATTERISTICHE TECNICHE

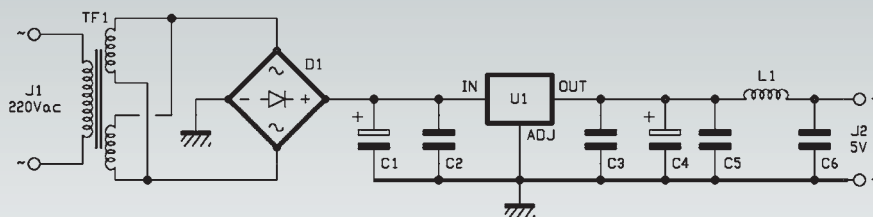
- Alimentazione esterna 220Vca con filtro anti-RF
- Alimentazione interna 5Vcc - consumo in corrente 140mA
- Visualizzazione Frequenza su display LCD 16 caratteri x 2 righe
- Retroilluminazione regolata automaticamente (diminuisce in assenza di misurazioni)
- Tre ingressi BNC commutabili elettronicamente: BF, TTL, RF
- Indicazione a LED dell'ingresso attivo
- Gamme frequenza:
 - Ingresso BF :1Hz ~ 10MHz
 - Ingresso TTL :1Hz ~ 50MHz
 - Ingresso RF : 1MHz ~ 1,1GHz
- Un ingresso RF con presa SMA, in parallelo all'ingresso RF BNC
- Quattro portate commutabili elettronicamente: Hz, kHz, MHz, GHz
- Indicazione della portata sul display LCD
- Sensibilità degli ingressi:

Ingresso TTL	
Frequenza	Ulivello logico
1Hz÷50MHz	3V÷12V
Duty Cycle	20%-80%

Ingresso BF	Ingresso RF	Sensibilità
Frequenza	Frequenza	dBm mVpp
1Hz	1MHz	-6,6 300
10Hz	5MHz	-19 71
100Hz	10MHz	-33 14
1kHz	50MHz	-39 7
10kHz	100MHz	-39 7
100kHz	200MHz	-33 14
1MHz	300MHz	-29,5 21
2MHz	400MHz	-28 25
3MHz	500MHz	-27 28
4MHz	600MHz	-24,1 40
5MHz	700MHz	-21 57
6MHz	800MHz	-17,4 85
7MHz	900MHz	-16,1 100
8MHz	1000MHz	-10,1 200
9MHz	1040MHz	-9,2 225
10MHz	1100MHz	-8,2 250

[schema **SEZIONE DI ALIMENTAZIONE**]

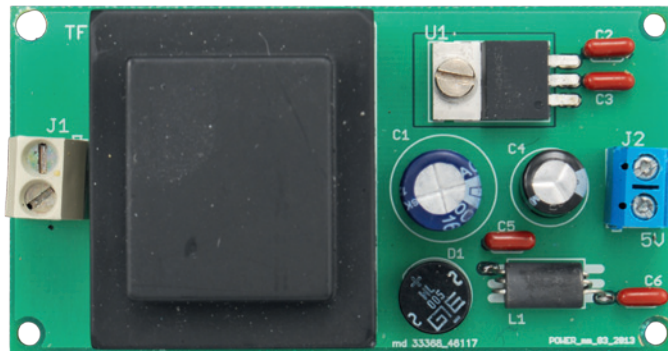
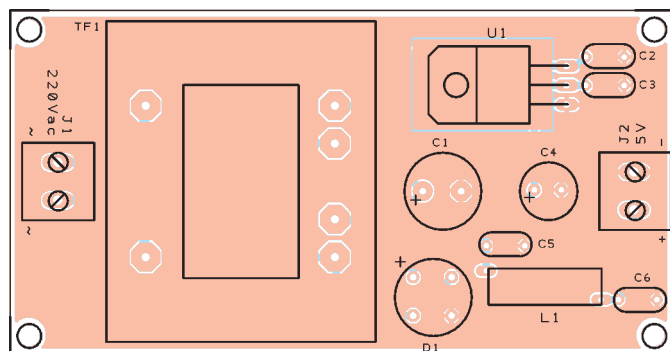
L'oscillatore Q1 necessita di 5 V applicati al pin 14 rispetto al 7 (GND) mentre la sua uscita (pin 8), sulla quale è presente un'onda quadra TTL di 16 MHz, è connessa al pin 9 del micro; il pin 10 dell'ATMega resta libero. Ma per garantire la massima precisione nella misura non basta usare l'oscillatore esterno: è necessaria anche una certa compensazione lato software; per facilitare l'operazione di taratura finale abbiamo pensato di implementare un



elemento hardware, che si avvale dell'uso dell'ADC del micro. Come spiegato dall'ATMEL, per usare al meglio l'ADC bisogna disaccoppiare l'alimentazione analogica AVCC (20) da quella generale del micro, VCC (7); noi l'abbiamo fatto interponendo tra questi due pin un induttore da 10 μ H (L1). Inoltre, tra il pin 20 e GND abbiamo collegato un condensatore da 100 nF (C4). Altro accorgimento è quello di adottare come riferimento per l'ADC una tensione molto stabile, invece di quella di alimentazione; nel nostro caso abbiamo ottenuto ciò attivando da firmware il riferimento interno a 1,1 V, abbastanza preciso e molto stabile. Tale tensione è disponibile tra il pin 21 (AREF) del micro e massa; filtrata dal condensatore C3 e applicata agli estremi di un trimmer multi-giri (R2), ci permette di applicare al pin 28 (ADC5, cioè ingresso 5

dell'ADC) il riferimento che servirà per tarare la precisione di lettura dello strumento. Regolando il trimmer R2, applicheremo al pin 28 una tensione variabile tra 0V e 1,1V. Oltre a leggere la frequenza, il micro gestisce un display LCD da 16 caratteri x 2 righe, mediante la tipica configurazione a 4 bit; il trimmer R6 serve per regolare *una tantum* il contrasto. Un piccolo approfondimento merita la gestione della retroilluminazione (backlight). In fase di "riposo" dello strumento, cioè quando esso non sta misurando, il pin 18 è a livello alto, il transistor PNP T1 è quindi interdetto, ma la resistenza R5 permette il passaggio di una corrente tale da far funzionare la retroilluminazione a bassa luminosità. Quando invece il micro sta effettuando misure, lo stesso pin 18 va a zero, T1 entra in conduzione e all'LCD arriva la massima corrente possibile. A tal

[montaggio **SEZIONE DI ALIMENTAZIONE**]



Elenco Componenti:

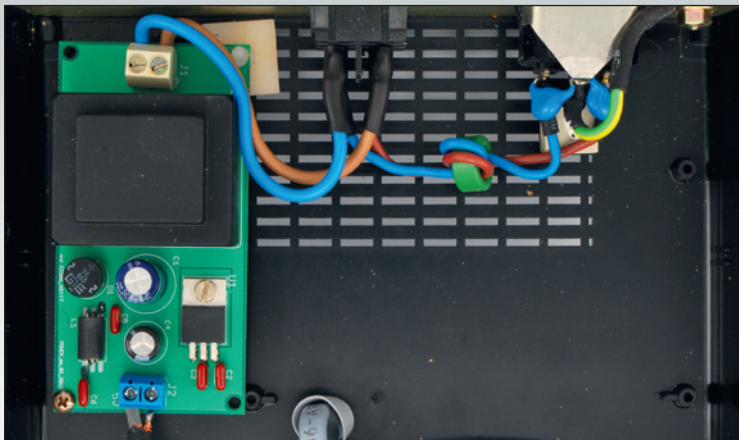
C1: 1000 μ F 25 VL elettrolitico
C2, C3, C5, C6: 100 nF 63 VL poliestere
C4: 470 μ F 16 VL elettrolitico
U1: LM1117-5.0

TF1: Trasformatore c.s. 2x6V 3VA
(cod. 6150-2060030M)
L1: Impedenza VK200
D1: Ponte diodi 1A

Varie:

- Morsetto 2 vie (2 pz.)
- Circuito stampato

Fig. 1 - Il PCB dell'alimentatore con tutti i componenti montati.



proposito è importante verificare se il display che usate ha o meno la resistenza interna di limitazione per la retroilluminazione; in caso affermativo la R3 può essere una 0 ohm o anche un banale ponticello, mentre in caso contrario R3 deve avere una resistenza tale da evitare che i LED di retroilluminazione si brucino.

I pulsanti P1 e P2, collegati ai pin 4 e 5 del micro, servono rispettivamente per selezionare l'ingresso (BF-TTL-RF) da usare per la misura e per cambiare la portata; nel primo caso il pulsante agisce sui relé collocati sulla Sezione ingressi, mentre nel secondo l'azione svolta è esclusivamente da software e il display mostrerà l'unità di misura selezionata (Hz, kHz, MHz, GHz) e la frequenza misurata espressa in base ad essa. Ad esempio, se stiamo leggendo una frequenza di 100 kHz, in base alla portata selezionata sul display vedremo: 100000 Hz, 100.000 kHz, 0.100000 MHz o 0.00100000 GHz.

Il connettore ad innesto obbligato J1 mette in diretta comunicazione questa scheda con il PCB della Sezione ingressi; conta sei pin (in configurazione 3x2) che collegano rispettivamente:

- 5 V, per portare l'alimentazione generale alla Sezione ingressi;
- F_IN, ossia il segnale proveniente dalla Sezione ingressi, squadrato e adattato alle pre-

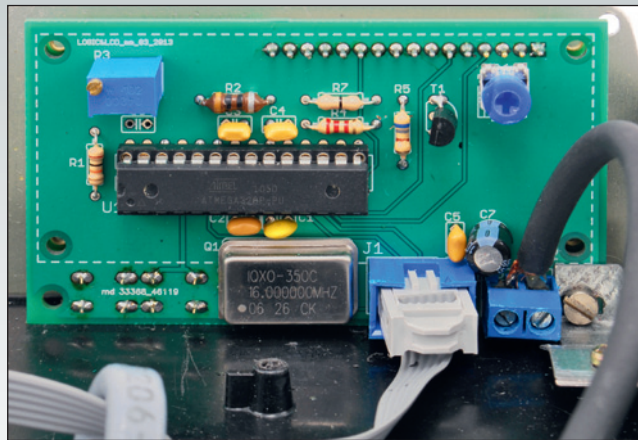
stazioni del micro (che lo riceve al pin 11);

- controllo del relé SWC, operato dal pin 2 del micro;
- controllo del relé SWB, operato tramite il pin 3 del micro;
- controllo del relé SWA, operato dal pin 6 del micro;
- GND, per collegare tra loro le masse dei due circuiti.

Vediamo ora il montaggio del PCB, tenendo per ultimi i due pulsanti e l'LCD, che devono essere montati dal lato saldature perché dovranno fuoriuscire dalla mascherina anteriore del contenitore.

Al solito, iniziate dai componenti più bassi, quindi resistenze e induttanza, proseguendo con i condensatori ceramici e in poliestere, lo zoccolo dell'integrato e quindi tutti gli altri. Attenzione al posizionamento dei due connettori J1 e J2: il primo deve avere la tacca di riferimento rivolta verso l'interno del PCB (cosa peraltro visibile sulla serigrafia) e il secondo deve avere i fori per gli ingressi dei fili rivolti anch'essi verso l'interno del PCB. Non rispettando questa indicazione, nel caso di J1 si invertono i collegamenti (quindi l'alimentazione) mentre per J2 non è possibile collegare i due fili, in quanto la sua parete esterna si troverà a contatto con la base del contenitore. Per quanto riguarda la polarità, nessun problema per J1, in quanto ha l'innesto obbligato, mentre

Fig. 2 - Il PCB dello stadio della Logica lato componenti.



per J2 bisogna fare attenzione alla serigrafia: il polo positivo (come per l'alimentatore) è indicato dalla dicitura "5V" e dal simbolo "+".

Inizialmente avevamo previsto connettori ad innesto, ma poi abbiamo visto che erano più sensibili alle RF ed abbiamo optato per il contatto diretto, mediante sistema a vite. Per collegare l'alimentatore a questo stadio è sufficiente usare uno spezzone di filo schermato per bassa frequenza, con il centrale collegato a +5V e la calza a GND. In Fig. 2 vedete il PCB dello stadio della logica. Solo dopo aver completato le saldature dei componenti montati sul lato superiore del PCB, sarà possibile completare il montaggio lato saldature, con i due tasti P1 e P2 e il display LCD 16x2. Quest'ultimo va collegato al PCB mediante una fila di 16 pin saldati sia lato PCB che lato display; nel caso dei display provati da noi, il solo spessore di plastica dei pin (pochi mm) è sufficiente ad evitare contatti con il PCB della logica. Tuttavia il sistema di chiusura della mascherina copri-display, con delle linguette metalliche piegate sul fondo del PCB, potrebbe in qualche modo rappresentare un problema. Quindi, prima di procedere con la saldatura occorre inserire entrambi i PCB sulla fila di pin e controllare visivamente se c'è contatto; in caso affermativo, bisogna ricorrere a una coppia di distanziatori plastici spessi 4 mm per garantire una corretta

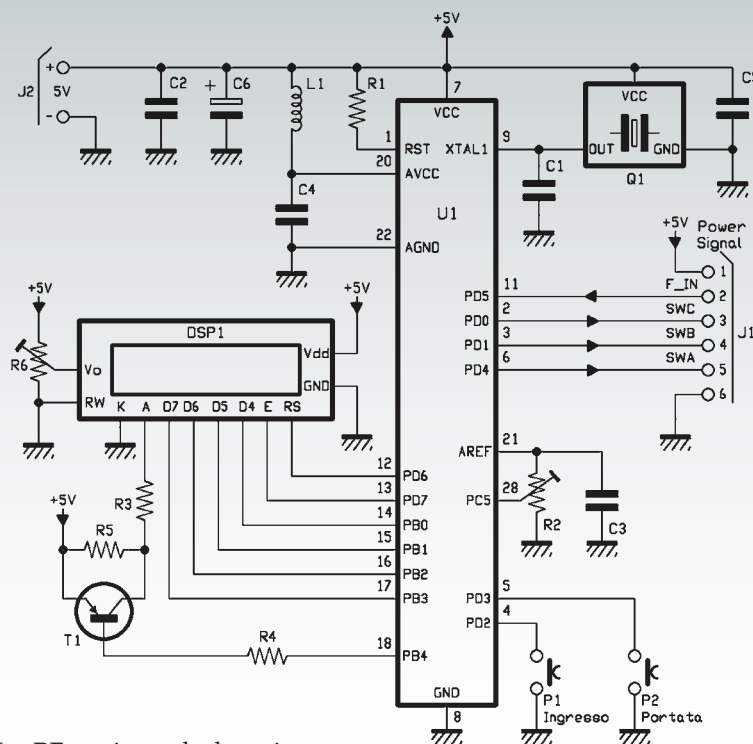
distanza tra i due PCB. A riguardo notate che, per fuoriuscire dalla mascherina anteriore, i tasti devono avere un attuatore da 13 mm; se si dovesse distanziare ulteriormente il display, probabilmente sarà necessario ricorrere a dei pulsanti specifici per montaggio a pannello, collegandoli via filo al PCB. In Fig. 3 è visibile il collegamento saldato tra il display ed il PCB della logica e i due tasti con attuatore da 13 mm (fate attenzione al momento dell'acquisto, perché spesso la lunghezza dell'attuatore viene indicata sommata a quella del corpo, quindi il 13 mm viene venduto come 17 mm). Da notare anche il piccolo spessore plastico che serve ad evitare il contatto dei pin saldati con l'alluminio della parte interna della mascherina frontale; permette al display di fuoriuscire quanto basta ma impedisce il cortocircuito.

SEZIONE INGRESSI: BF

Questo stadio è un PCB che riceve e adatta i segnali dei tre ingressi BF,

TTL e RF; ospita anche la sezione delle commutazioni, che si occupa della selezione dell'ingresso da utilizzare e, nel caso dei due ingressi BF e TTL, anche di abilitare o meno il divisore x10 posto a valle. Iniziamo dall'ingresso BF, il cui cuore è

costituito dai due doppi amplificatori operazionali della Microchip, siglati MCP662, con caratteristiche di tutto rispetto: 60 MHz di banda passante, uscita Rail-to-Rail, rumore



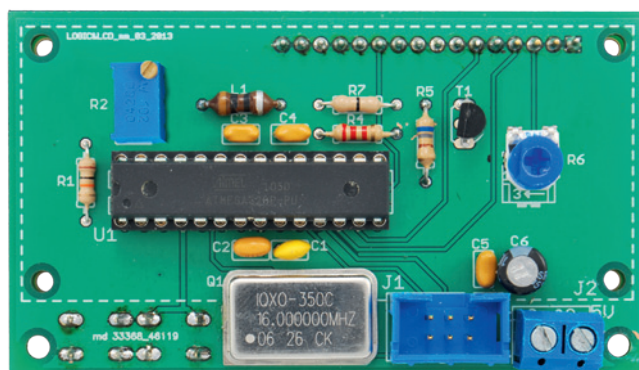
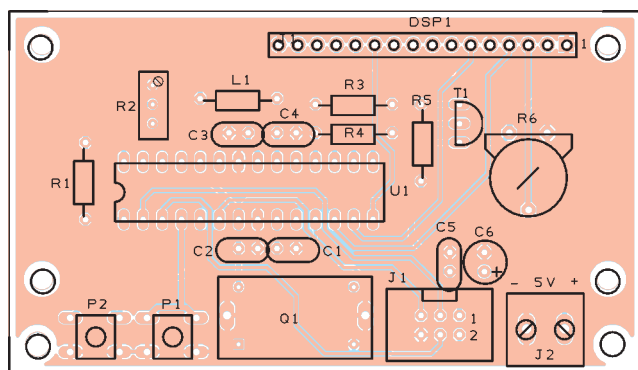
[montaggio LOGICA E LCD]

Elenco Componenti:

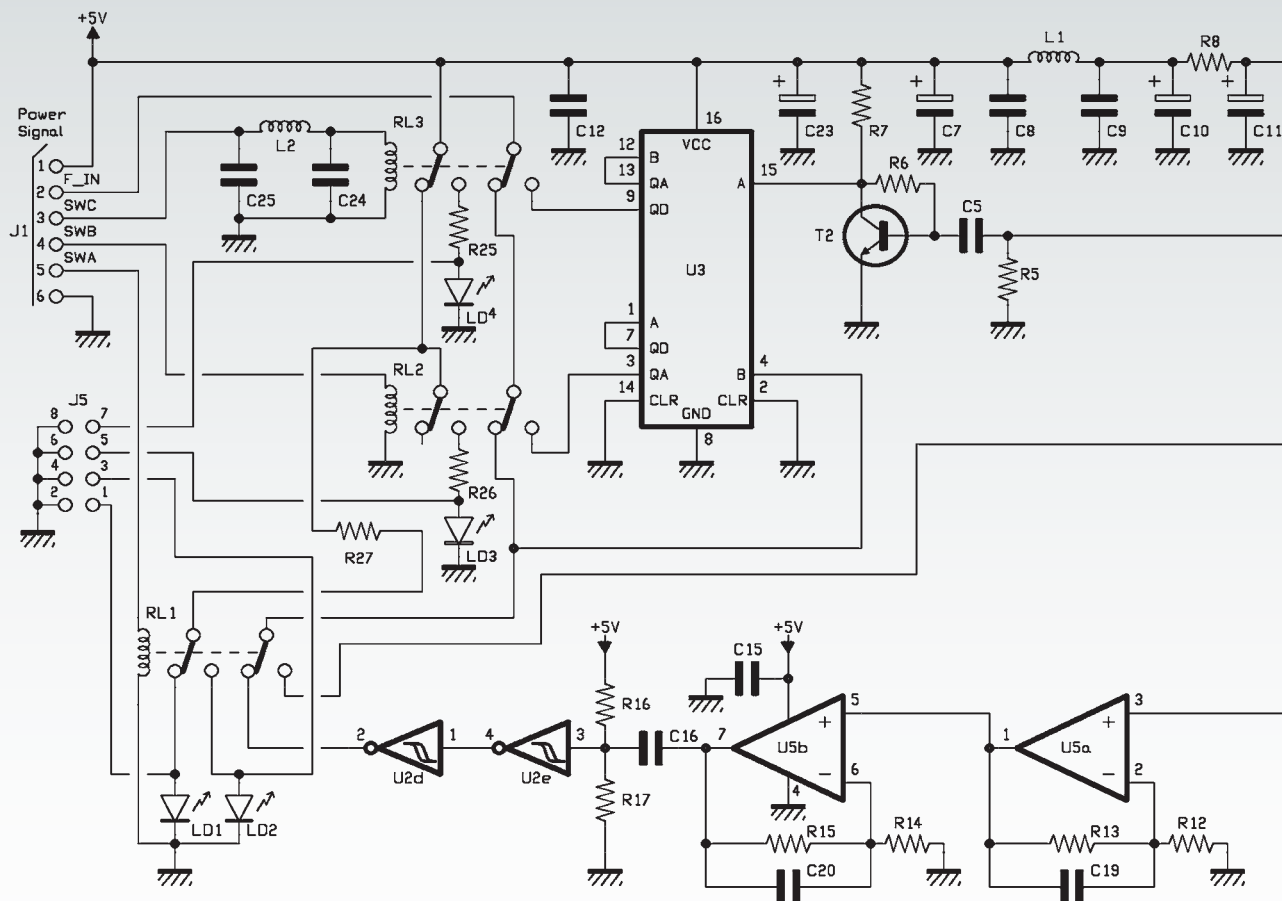
R1: 10 kohm	C6: 100 μ F 16 VL elettrolitico
R2: Trimmer multigiri 1 kohm	U1: ATMEGA328P-PU
R3: 0 ohm (vedi articolo)	T1: BC547
R4: 2,2 kohm	Q1: Oscillatore quarzato 16MHz (DIL14S)
R5: 680 ohm	P1, P2: Microswitch
R6: Trimmer 10 kohm MO	L1: Induttanza 10 μ H
C1: 15 pF ceramico	DSP1: Display LCD 16x2
C2÷C5: 100 nF multistrato	

Varie:

- Morsetto 2 vie
- Connettore IDC 6 vie maschio
- Strip maschio 16 vie
- Strip femmina 16 vie
- Zoccolo 14+14
- Circuito stampato



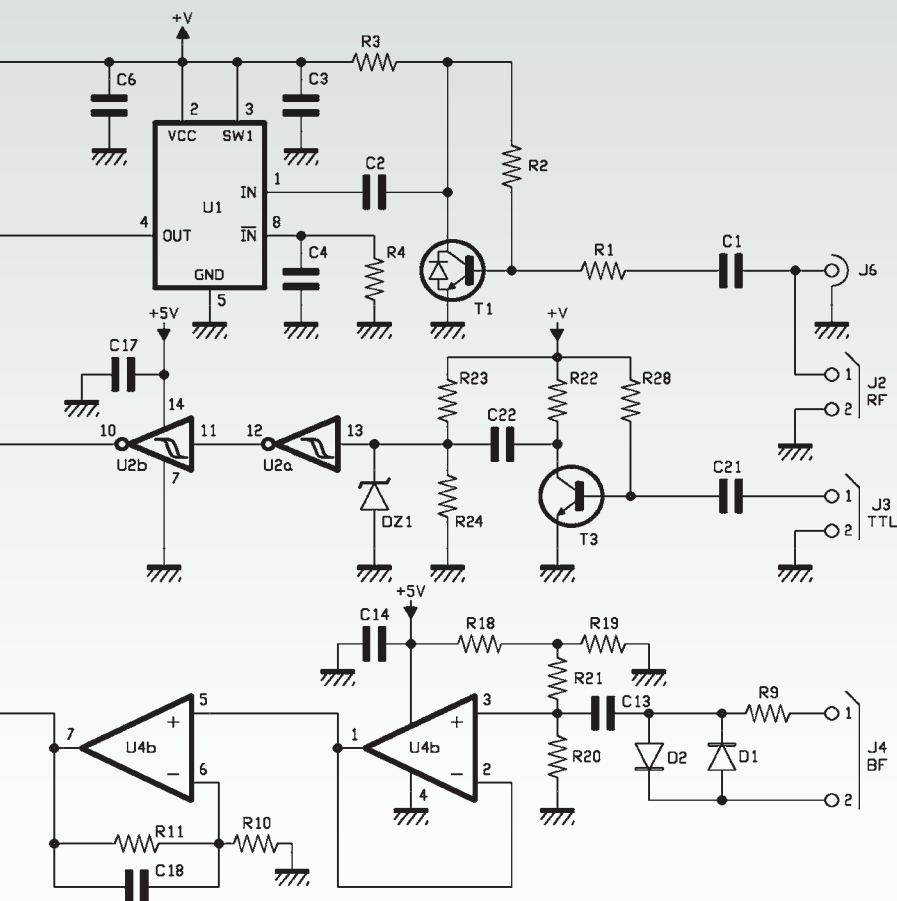
[schema **SEZIONE INGRESSI**]



bassissimo (tipicamente $6,8 \text{ nV}/\sqrt{\text{Hz}}$ ad 1 MHz), consumo di 6 mA . Il segnale in ingresso viene limitato ad un'ampiezza massima di $0,4 V_{pp}$ dal gruppo di protezione costituito da $R9$ e dai diodi Schottky $D1$ e $D2$, collegati in antiparallelo. Il successivo condensatore non polarizzato ($C13$) del valore di $10 \mu\text{F}$, ha il compito di bloccare l'eventuale componente in corrente continua associata al segnale, prima che esso giunga all'ingresso non invertente di $U4A$, il primo dei due op-amp dell'integrato, usato in configurazione "voltage follower" (inseguitore di tensione). Questo stadio ci permette di ottenere un ingresso ad alta impedenza e contemporaneamente di applicare un offset CC fisso, in modo da uniformare la "posizione" di partenza del segnale applicato in ingresso. In pratica con $C13$ blocchiamo la componente continua (di

cui non conosciamo l'ampiezza a priori) associata al segnale originale, e così evitiamo la possibile saturazione degli stadi successivi, mentre con il circuito costituito dalle resistenze $R18$ - $R19$ - $R20$ - $R21$ applichiamo un offset CC costante di 50 mV . La coppia $R18$ - $R19$ è un partitore con rapporto $1:50$, quindi nel suo punto centrale presenta una tensione di 100 mV ; la coppia $R20$ - $R21$ è un partitore $2:1$, che quindi riduce la tensione ulteriormente a 50 mV . Questo "trucco" del doppio partitore ci ha permesso di avere l'offset richiesto sull'ingresso non invertente di $U4A$, ma senza caricarlo con una resistenza di basso valore ohmico, che avrebbe vanificato la funzione di questo stadio. Lo stadio voltage follower ha anche la caratteristica di tagliare la componente negativa del segnale reale; nel caso in cui esso sia simmetrico rispetto allo 0, sulla sua uscita

presenta un segnale positivo con ampiezza pari a $V_{pp}/2$, in cui V_{pp} è l'ampiezza del segnale espressa in Volt picco-picco. L'uscita del primo stadio viene applicata ad una catena amplificatrice costituita dal secondo operazionale di $U4$ e dai due op-amp di $U5$, tutti in configurazione non invertente. Tale configurazione ha la caratteristica di amplificare il segnale in ingresso secondo la formula: $(R_a/R_b)+1$, in cui R_a è la resistenza posta tra ingresso invertente ed uscita (nel nostro caso, per i tre stadi, sono $R11$, $R13$ e $R15$), mentre R_b è la resistenza posta tra ingresso invertente e GND (nel nostro caso $R10$, $R12$ e $R14$). Di conseguenza il primo stadio ha un fattore di amplificazione di $4,3x$, mentre i due stadi successivi guadagnano $3,2$ volte. Poiché le amplificazioni vengono applicate in cascata, otteniamo un fattore globale



di 44x, che espone l'intero circuito al rischio dell'auto-oscillazione, come abbiamo avuto modo di verificare in fase sperimentale. Per stabilizzare il sistema è stato necessario aggiungere in parallelo ad ogni resistenza di retroazione un condensatore da 1pF (C18, C19 e C20). L'uscita dell'intero stadio di amplificazione presenta un segnale analogico che va trasformato in modo che sia compatibile con l'ingresso del microcontrollore. La squadratura viene effettuata da due porte NOT Schmitt-trigger, in serie tra loro, appartenenti a U2 (74HCT14); C16 si occupa di ripulire il segnale dalla componente di tensione continua e il partitore R16-R17 adatta i livelli in uscita dallo stadio di amplificazione con quelli TTL necessari allo squadratore. Infatti, è noto che una porta logica TTL riconosce come HIGH la porzione di segnale con ampiezza minima di

2 V, mentre identifica come LOW la porzione di segnale con ampiezza massima di 0,8 V. Sull'uscita dello stadio amplificatore analogico potremmo trovare livelli non compatibili con questi, col risultato di falsare le misure o addirittura di non riuscire a misurare alcunché. Ad esempio, se il segnale si venisse a trovare nella zona indeterminata della porta TTL, a motivo del comportamento della porta NOT Schmitt-trigger, all'uscita dello stadio squadratore (pin 2 di U5) troveremmo un segnale LOW fisso, e quindi il micro non leggerebbe alcuna frequenza. Il partitore R16-R17 aggiunge al segnale una componente continua

fissa di circa 1,3-1,4V, che sommata all'ampiezza del segnale amplificato, permette di raggiungere un'ampiezza compatibile con i livelli TTL. A questo punto sull'uscita dello stadio squadratore troveremo un segnale perfettamente squadrato con ampiezza di circa 5 Vpp.

Per farvi un'idea del funzionamento dello stadio ingressi, guardate la Fig. 4: l'immagine "A" mostra il segnale misurato direttamente sul BNC d'ingresso, ossia quello che stiamo per misurare, costituito da una componente positiva e da una negativa rispetto allo 0 (linea orizzontale indicata con "1►"). In "B" vediamo il segnale come appare all'ingresso dello stadio voltage follower: in pratica il segnale è stato attenuato (da 50 mVpp a 20 mVpp) dai componenti presenti sull'ingresso, e contemporaneamente è stato portato verso l'alto di 50 mV a causa dell'offset CC applicato al segnale stesso, per cui ora il segnale è tutto positivo mentre l'offset CC è suddiviso in due parti: circa 27 mV sono sopra lo 0 e circa 23 mV restano sotto lo 0. In "C" il segnale in uscita dal primo op-amp è praticamente immutato, in quanto non ha parti negative da tagliare. In "D" vediamo il segnale dopo la prima amplificazione: da notare che nella banda "MISURA" sono riportati sia il valore Vmax (ampiezza totale del segnale compresa la componente CC positiva) che il valore Vpp (ampiezza del solo segnale).

Fig. 3 - Il montaggio del display e dei due tasti con attuatore lungo.

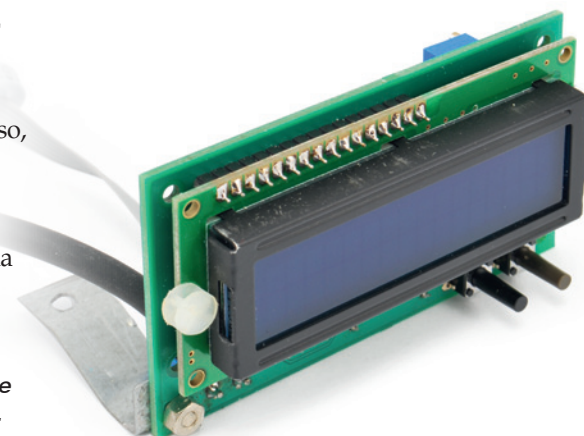
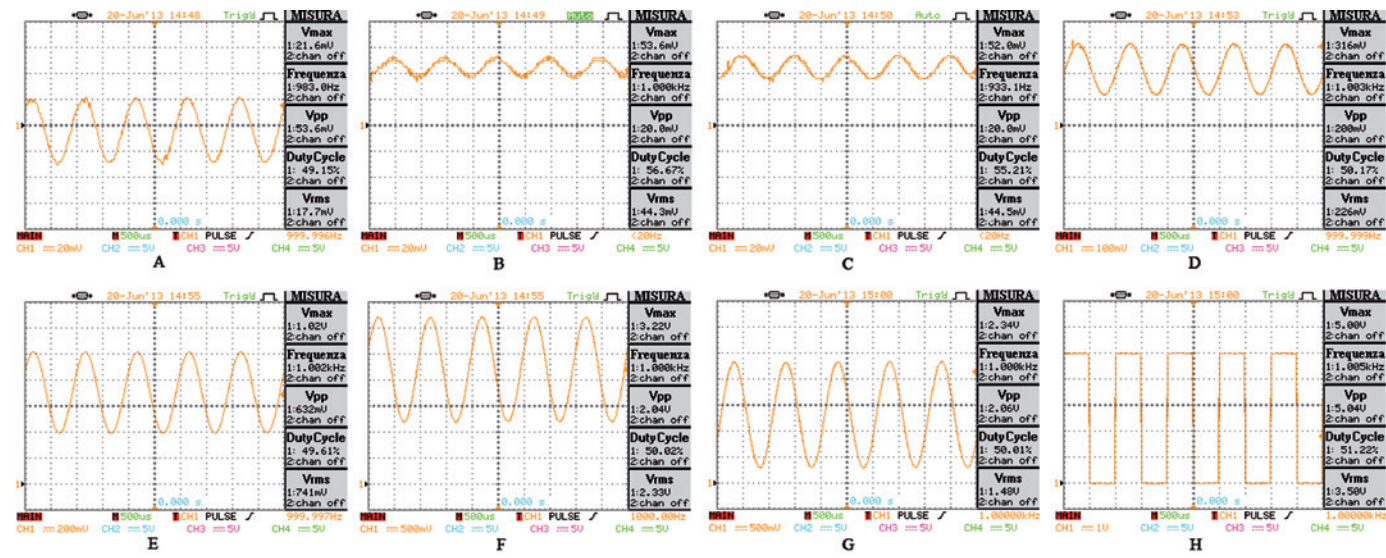


Fig. 4 - Amplificazione e squadratura di un segnale BF a 1 kHz/50 mVpp: le schermate sono riprese dall'oscilloscopio digitale DSO.

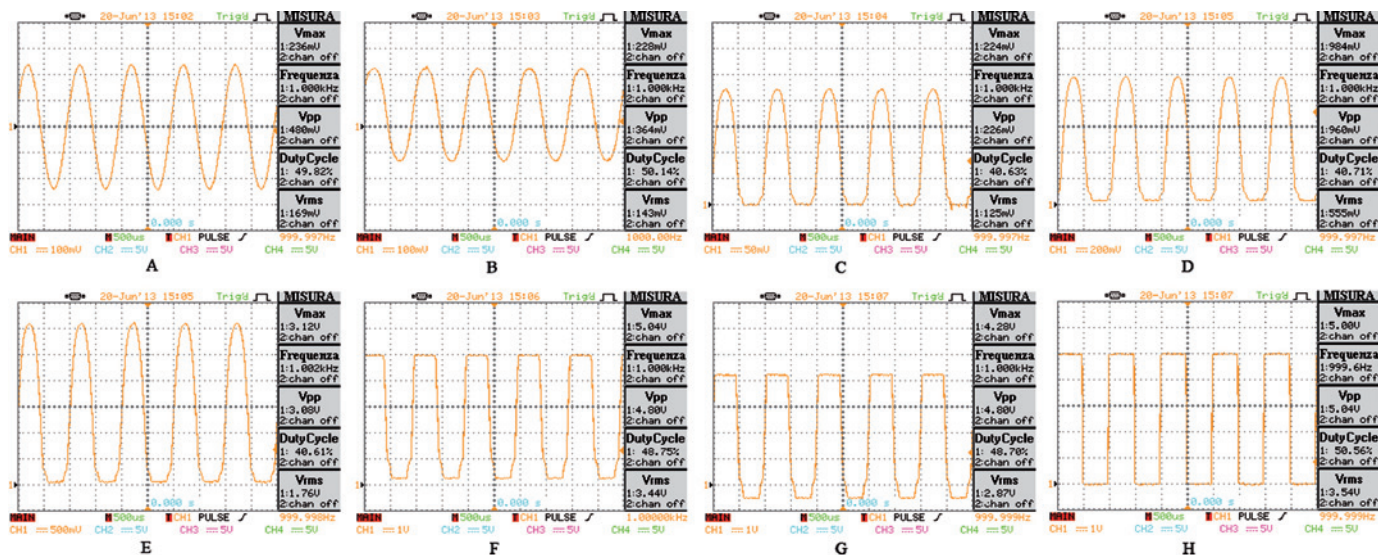


L'amplificazione sembrerebbe di circa 6x per il segnale complessivo e di circa 10x per il segnale reale, invece del previsto 4,3x, ma i valori in gioco sono così bassi che nella misura precedente anche la sonda del DSO potrebbe aver attenuato la misura, e comunque il segnale è stato amplificato almeno quanto ci si aspettava, se non di più. In "E" vediamo l'uscita del secondo amplificatore: qui l'amplificazione prevista di 3,2x è perfettamente rispettata, perché i 316 mV del segnale complessivo sono diventati 1,02 V, mentre i 200 mV del segnale reale sono diventati 632 mV. In "F" invece abbiamo l'uscita del terzo ed ultimo amplificatore (3,2x); anche qui è tutto perfetto: 1,02V diventano 3,22V e 632 mV diventano 2,04V. Ora, se applicassimo questo segnale nella sua interezza, compreso l'offset CC, direttamente all'ingresso dello squadratore, ci troveremmo il picco inferiore del segnale reale posizionato a circa 1,25 V; infatti lo "0" relativo, in "F", è posto sul primo quadratino in basso (1►) e il V/div è di 500 mV, quindi ogni quadratino rappresenta 0,5V. Il picco negativo del segnale è posizionato a circa 2,5 quadratini rispetto allo "0", quindi abbiamo $0,5 + 0,5 + 0,25 = 1,25V$; ciò significa che il picco inferiore si trova al di sopra della soglia massima di riconosci-

mento di un livello LOW da parte di una porta TTL. In queste condizioni la prima porta NOT Schmitt trigger in uscita avrebbe un livello H stabile, la seconda un L stabile, quindi nessuna misura della frequenza. In "G" vediamo allora l'effetto dei componenti C16, R16 e R17: il condensatore elimina l'offset CC ed il partitore aggiunge un nuovo offset di circa 1,3 V; il risultato è che ora il segnale parte da circa 0,3 V ed arriva a circa 2,3 V, quindi è perfettamente compatibile con i livelli logici TTL. Infatti in "H" troviamo un'onda quadra perfetta di 5Vpp, che parte esattamente da 0 V. Ma perché applichiamo un offset DC di 1,3V e ci troviamo solo 0,3V alla base del segnale? La risposta la troviamo seguendo il percorso di quei 50 mV di offset CC iniziali; ricordiamo che 27 mV si trovano sopra lo "0" e, all'uscita del terzo amplificatore saranno diventati circa: $0,027 \times 4,3 \times 3,2 \times 3,2 = 1,2V$; anche i 23 mV negativi sono stati amplificati e sono diventati circa: $0,023 \times 4,3 \times 3,2 \times 3,2 = 1V$. Nel momento in cui il segnale attraversa C16, tutto l'offset CC (sia positivo che negativo) viene eliminato, per cui il segnale reale si trova a partire da circa 1 V al di sotto dello 0; andando poi ad aggiungere 1,3 V col partitore, il segnale sale verso il

positivo e si trova quindi a partire 0,3 V sopra lo 0 volt. Ed ora vediamo cosa succede applicando in ingresso un segnale da circa 500 mVpp (Fig. 5). L'immagine "A" mostra il segnale originale, applicato al BNC d'ingresso: 1 kHz di frequenza ed un'ampiezza di circa 480 mVpp. In "B" (ingresso voltage follower) il segnale si presenta ridotto a circa 360 mVpp in funzione del taglio effettuato dai diodi di protezione D1 e D2, ma non è più equamente diviso rispetto allo 0, perché ad esso è stato ora sommato l'offset DC di 50 mV; la parte positiva è ampia, circa 230 mV. Come già spiegato, lo stadio voltage follower ha la caratteristica di tagliare la componente negativa del segnale ad esso applicato, infatti in "C" ora ci ritroviamo solo la parte positiva, sempre di circa 230 mV, mentre il "taglio" è evidente alla base del segnale, che appare piatto e coincide con lo 0. Tutto l'offset CC si trova al di sotto dello 0, infatti i valori Vmax e Vpp coincidono. In "D" vediamo l'effetto dell'amplificazione 4,3x (primo stadio): 226 mVpp diventano 960 mVpp; è evidente l'aumento del segnale tutto nella fascia positiva (la base è sempre piatta). In "E" abbiamo $960mV \times 3,2 = 3,08V$ circa. In "F" notiamo qualcosa di

Fig. 5 - Amplificazione e squadratura di un segnale BF a 1kHz 500mVpp.



nuovo: in base all'amplificazione 3,2x di quest'ultimo stadio, avremo dovuto ottenere un segnale ampio circa 10 Vpp, ma questo è impossibile in quanto l'integrato è alimentato a 5 V, ed allora si ha un effetto di saturazione, per cui ora il picco superiore (Vmax) coincide appunto con i 5V e il segnale viene tagliato a questo livello, mentre nel picco inferiore notiamo un offset CC di circa 0,2 V; il segnale assume una forma molto simile all'onda quadra. Infatti con un segnale del genere, con ampiezza di 5 V (4,8 V di segnale e 0,2 V di offset CC), potremmo tranquillamente entrare nella porta TTL, però il circuito è unico quindi esso subirà lo stesso le ultime due trasformazioni.

In "G" vediamo il solito effetto di C16, R16 e R17: l'offset CC (2 V circa, di cui 1,8 V sotto lo 0) viene eliminato da C16; a questo punto il segnale partirebbe da -1,8 V e finirebbe a 3,2 V, invece il partitore R16-R17 aggiunge nuovamente 1,3 V e quindi il segnale parte da -0,5 V e termina a 4,2 V.

In "H" vedete il segnale definitivo a 5 Vpp, riferito a 0V, regolarmente squadrato: a prescindere dall'ampiezza del segnale in ingresso (nei limiti della sensibilità dichiarata), in uscita c'è sempre un'onda perfettamente quadra a livelli TTL. Tale

segnale sarà applicato, tramite la sezione di commutazione, direttamente all'ingresso del micro; se la sua frequenza è di valore non superiore a 5,5 MHz il micro provvederà direttamente alla sua misurazione, mentre in caso contrario esso comanderà il passaggio del segnale attraverso un divisore x10 e ne misurerà il risultato.

SEZIONE INGRESSI: TTL

L'ingresso BF permetterebbe di misurare segnali fino a 10 MHz con ampiezze molto basse; laddove serva misurare segnali digitali, tipo TTL o CMOS, anche se aventi frequenze compatibili con lo stadio BF, è preferibile applicarle all'ingresso TTL. Tale ingresso accetta ampiezze da 3V a 12 Vpp e frequenze da 1 Hz a 50 MHz. Ciò non sarebbe possibile applicando tali segnali direttamente alla solita coppia di porte NOT Schmitt trigger, che accetta fino a 5 V, quindi abbiamo previsto uno stadio costituito da T3, C21, C22, R22, R23, R24, R28 e DZ1, che ripulisce il segnale in ingresso dalla componente continua e ne porta l'ampiezza a un livello tollerabile dalla porta NOT, applicando un offset CC fisso. DZ1 taglia i segnali con ampiezza maggiore di 5 V. Il transistor entra in conduzione nella porzione HIGH del segnale e si interdice nella porzione

LOW, trasformandolo in un segnale ad ampiezza fissa che, ripulito da C22 e adattato da R23-R24 arriva alla prima porta NOT. Lo stadio squadratore si occuperà di ripulire ulteriormente il segnale, trasformandolo in un perfetto segnale ad onda quadra, con ampiezza di circa 5 Vpp. Anche in questo caso la frequenza sarà letta direttamente dal micro per verificare se essa sia maggiore di 5,5 MHz e quindi necessiti di una divisione x10.

Da notare che l'alimentazione dello stadio iniziale è stata presa dal circuito filtrato dell'ingresso RF (ne parleremo più avanti), infatti è indicata come "V+", questo ha consentito di bloccare disturbi che avrebbero creato non pochi problemi dai 15 MHz in su.

La Fig. 6 aiuta a comprendere il funzionamento dell'ingresso TTL in presenza di segnali con ampiezza di 3-5-12 Vpp, con frequenza di 1 MHz. Iniziamo dall'immagine "A", che mostra il segnale applicato all'ingresso TTL, 1 MHz e 3 Vpp, equamente diviso rispetto allo 0. In "B" c'è il segnale rilevato sul collettore di T1, che ora è leggermente attenuato (2,5Vpp) e tutto negativo (ciò a causa del comportamento "invertente" del T1). In "C" vedete il segnale come arriva all'ingresso della porta NOT U2A, dopo essere stato traslato al di

Fino a 50 MHz

Per compiere misure fino a 50 MHz bisogna adottare un integrato 74HCT14 in grado di operare ad una frequenza anche maggiore (55-60 MHz). Molti costruttori producono tale integrato, ma non tutti garantiscono tale larghezza di banda, quindi dovete dare uno sguardo ai data-sheet e cercare il parametro Propagation Delay (ritardo di propagazione del segnale) che determina la massima frequenza di lavoro e viene espresso in ns. Leggete il valore "typical" e dividete 1.000 per esso: otterrete la frequenza in MHz. L'integrato da noi usato nelle prove è l'MM74HCT14MX prodotto dalla Fairchild Semiconductor, che ha Propagation Delay pari a 13 ns, quindi permette di operare fino a 76,923 MHz. In verità andrebbe considerato anche il Transition Time (tempo di transizione tra H e L o viceversa, anche detto Maximum output rise and fall time) che, nel calcolo, si somma al Propagation Delay; sempre nel caso del MM74HCT14MX, tale valore è 9 ns, quindi la frequenza massima diventa 45,454 MHz, ma le prove hanno ampiamente dimostrato che con questo integrato è possibile leggere tranquillamente i 50 MHz indicati nelle specifiche dello strumento.

sopra dello 0, ad opera del partitore. In "D" abbiamo il segnale in uscita dalla seconda porta NOT U2B, perfettamente squadrato e con ampiezza di 5 Vpp. Nell'immagine "E" si vede il nuovo segnale in ingresso, 1 MHz e 5 Vpp, sempre simmetrico rispetto allo 0.

In "F" c'è il segnale attenuato (3 Vpp) e tutto negativo, presente sul collettore di TR1. In "G" lo stesso segnale traslato sulla zona positiva, all'ingresso di U2A. In "H" vediamo il segnale in uscita da U2B, perfettamente squadrato e con ampiezza di 5 Vpp. Nell'immagine "I" appare il segnale sotto misura, che è ora di 1 MHz e 12 Vpp, ancora simmetrico rispetto allo 0.

In "L" c'è il segnale attenuato (circa 6 Vpp) e tutto negativo, localizzato sul collettore di T1. In "M" vedete lo stesso segnale traslato sulla zona positiva, all'ingresso di U2A. Da notare che l'ampiezza massima è di circa 5 V, causa l'intervento dello zener DZ1, che taglia tutta la tensione in eccesso. In "N" c'è il segnale in uscita da U2B, perfettamente squadrato e con ampiezza di 5 Vpp. Nelle immagini riprese abbiamo usato segnali simmetrici rispetto allo 0, ma il frequenzimetro si comporta alla stessa maniera anche applicando segnali solo positivi.

SEZIONE INGRESSI: RF

Lo stadio dedicato alle misure RF permette di leggere frequenze fino a 1,1 GHz e segnali di ampiezza minima da 7 a 250 mVpp. Il cuore di

questa parte di schema è il prescaler della Microchip MC12080D (U1) che è un integrato in grado di accettare frequenze fino a 1,1 GHz e dividerle x10, x20, x40 o x80, in base alla configurazione dei pin SW1-SW2-SW3. Per migliorare la già buona sensibilità di tale componente, abbiamo fatto ricorso a T1, un transistor bipolare ad emettitore comune, denominato BFP420, progettato per sistemi di amplificazione ad alto guadagno ($H_{fe}=100\div150$) e basso rumore, in grado di operare fino a ben 25 GHz di frequenza!

Come si vede nello schema elettrico il segnale di ingresso può essere applicato al BNC J2 o alla presa SMA J6, in quanto sono in parallelo tra loro; la scelta del doppio ingresso è legata alla sempre maggiore diffusione dello standard SMA, soprattutto come sistema di collegamento delle antenne di trasmissione e ricezione, ed anche di eccellenti sonde RF. Il segnale viene applicato, tramite C1-R1, direttamente alla base di T1 e viene opportunamente amplificato, quindi entra, tramite C2, nel prescaler U1 (pin 1) con il valore di frequenza originale. Nel nostro caso abbiamo adottato, per SW1-SW2-SW3, la configurazione H-L-L (il data-sheet specifica che per avere lo stato H bisogna collegare il pin al positivo di alimentazione, mentre per quello L basta lasciare sconnesso il pin). Tale configurazione corrisponde ad una divisione x40 della frequenza in ingresso, per cui, considerando il range di lettura da 1 MHz a 1,1

GHz, sul pin 4 di U1 troveremo una frequenza tra 25 kHz e 27,5 MHz. Considerando che la maggior parte dei valori del range di questo ingresso sarebbero superiori ai 5,5 MHz tollerati dal micro, abbiamo pensato, contrariamente a quanto fatto con gli altri due ingressi, di applicare direttamente un divisore x10 (U3B), in modo da avere sull'ingresso del micro frequenze comprese tra 2.500 Hz e 2,75 MHz.

Il 74HC390 è un doppio divisore a due stadi (x2 e x5); anche in questo caso vale lo stesso discorso fatto in precedenza per il 74HCT14 riguardo alla frequenza di lavoro; la questione non interessa tanto la metà U3B, usata per lo stadio RF, perché i 27,5 MHz sono una frequenza operativa raggiungibile da qualsiasi integrato della famiglia 74HC, quanto la metà U3A, usata come divisore per gli stadi BF e TTL, che può arrivare a leggere segnali fino a 50 MHz. Il metodo di calcolo della massima frequenza operativa è identico a quello descritto in precedenza.

Il segnale in uscita dal prescaler è di $0,8\div1,2$ V massimi con una corrente di 1,1 mA, quindi assolutamente insufficiente per pilotare il divisore HC, che richiede un'ampiezza minima di 2,4 V (possibilmente 3,15 V). Per tale ragione abbiamo aggiunto lo stadio costituito da C5, T2, R6 e R7, la cui funzione è amplificare il segnale in tensione in modo da poter essere letto correttamente dall'ingresso di U3B (pin 15); l'uscita del primo stadio, divisa per 2 (pin 13) viene applicata all'ingresso del secondo stadio (pin 12) e divisa per 5 (pin 9). Quindi sul pin 9 di U3B ci troveremo la frequenza originaria applicata all'ingresso J2 o J6, divisa x400, perfettamente ripulita e a livello logico TTL; tramite il sistema delle commutazioni essa sarà applicata poi direttamente all'ingresso del micro che provvederà a misurarla e a visualizzarla correttamente (dopo l'opportuna riconversione

via software). Una parte circuitale estremamente importante in questo stadio è l'alimentazione; osservando lo schema elettrico, infatti, si nota chiaramente come lo stadio preamplificatore e prescaler non siano alimentati direttamente dai 5 V soliti, bensì da una tensione chiamata "V+". Essa è frutto di un efficace filtraggio operato sui 5 V, mediante il circuito composto da C7, C8, L1, C9, C10, R8 e C11; possiamo garantire che senza di esso sarebbe stato impossibile avere dall'ingresso RF le prestazioni attuali. La tensione V+ è perfettamente stabile e pulita, anche durante le misurazioni RF più gravose, e ciò permette ovviamente la misurazione di segnali di piccola ampiezza, che altrimenti sarebbero sommersi dai disturbi ed autooscillazioni RF e quindi non potrebbero essere rilevati correttamente. Infine spendiamo qualche parola sul

circuito costituito da R4-C4: si tratta di un filtro R-C con il compito di eliminare tutti i segnali spuri che entrano nell'ingresso del prescaler, per migliorare le condizioni di lettura. Come tutti i filtri, però, esso incide, e non poco, sulla sensibilità d'ingresso. Noi abbiamo adottato un C da 1nF ed una R da 200 kohm, ottenendo i risultati che sapete. Aumentando il valore di R4 (C4 va bene così) migliora la sensibilità in ingresso ma aumentano anche i disturbi, per cui il frequenzimetro inizia a mostrare numeri casuali sul display, frutto di tutto ciò che la preamplificazione riesce a captare ed amplificare. Consigliamo di adottare la nostra combinazione: in assenza di segnale da misurare il display mostra rigorosamente 0 (lo stesso collegando una sonda di buona qualità); invece se si collega una sonda non idonea per le RF, il display (ma solo in assenza di

segnale) mostrerà valori a caso che scompariranno appena sarà rilevato un segnale stabile. Quindi questo può essere anche un buon metodo per verificare la qualità della sonda che desiderate usare: se appena la collegate alla presa BNC o SMA, in assenza di segnale da misurare, il display mostra numeri a caso, allora la sonda non ha caratteristiche valide per le misure RF.

In Fig. 7 descriviamo i vari passaggi per la misurazione di un segnale RF da 100 MHz con ampiezza di 100 mVpp. Anche in questo caso l'immagine "A" mostra il segnale originale, rilevato direttamente sulla presa BNC, mentre la "B" evidenzia il segnale, amplificato da T1, ora di circa 370 mVpp con un offset CC di circa 0,9 V ($V_{max} = 1,3$ V). In "C" il segnale è praticamente immutato, ma è aumentata la componente CC (ora la V_{max} è circa 2,9V); in queste

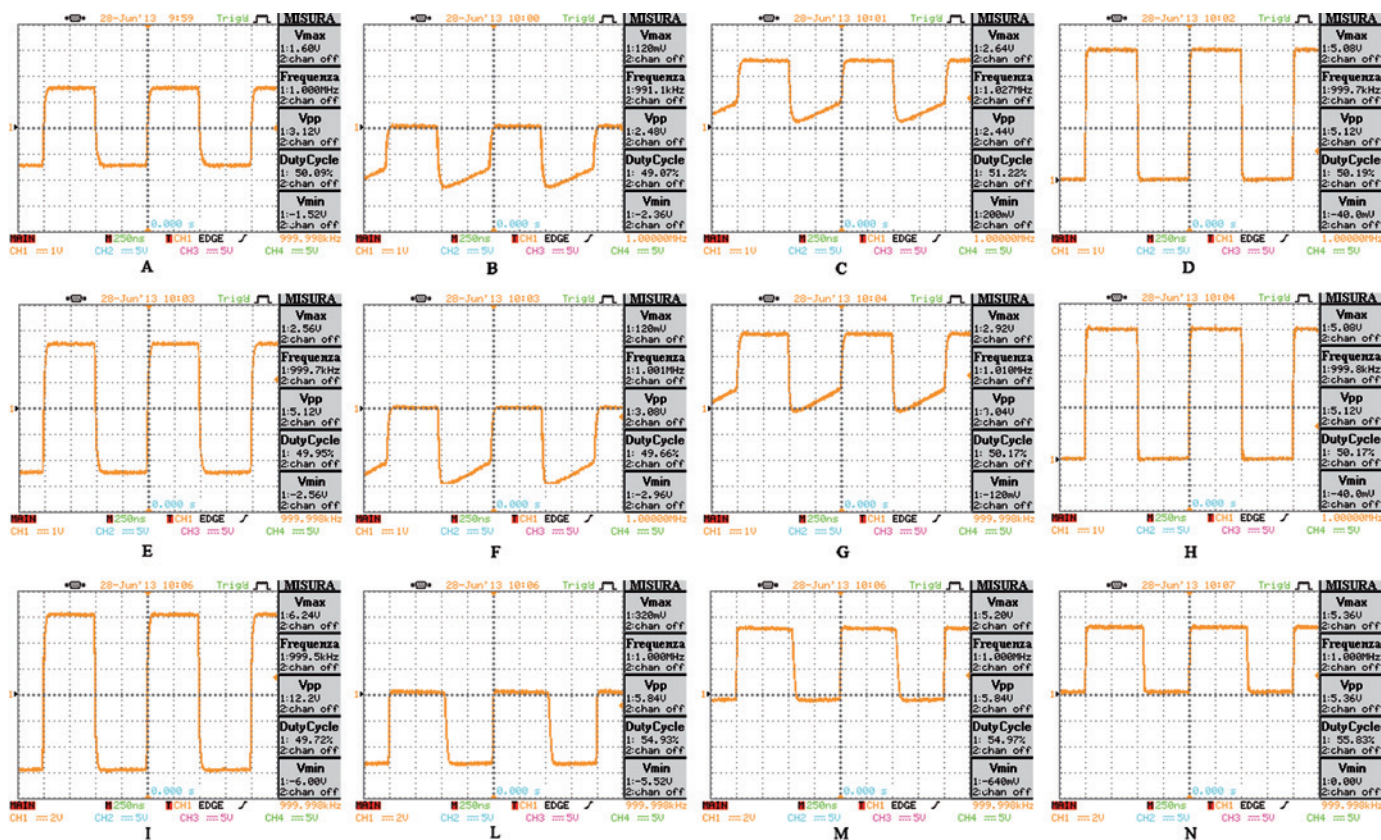


Fig. 6 - Lettura di segnali TTL ad ampiezza variabile.

condizioni viene applicato direttamente all'ingresso IN del prescaler U1. In "D" c'è il segnale sul pin 4 di U1, diviso x40 in frequenza (ora è 2,5 MHz) che ha un'ampiezza di circa 1,1 Vpp ed un'ampiezza totale Vmax di circa 4 V. In "E" il segnale viene nuovamente amplificato (C5, T2, R6, R7) e raggiunge un'ampiezza di circa 5 Vpp, con una leggera saturazione sui picchi, cosa che lo rende perfettamente compatibile con l'ingresso TTL di U3B (74HC390), che effettua un'ulteriore divisione x10 della frequenza. In "F" c'è il segnale, all'uscita di U3B, destinato al micro, con ampiezza 5 Vpp e frequenza iniziale divisa x400 (100 MHz->250 kHz). Il divisore U3, un 74HC390, è in realtà costituito da due coppie (U3A e U3B) di divisori x2 e x5: la prima l'abbiamo riservata agli stadi BF e TTL, come vedremo più avanti, effettuando prima la divisione x5 e poi quella x2, mentre la seconda coppia l'abbiamo usata per questo stadio, ma questa volta

abbiamo effettuato prima la divisione x2 e poi quella x5. Poi vedremo per quale motivo.

STADIO DEI RELÉ E DEL DIVISORE PER BF/TTL

Descriviamo ora l'ultima parte dello schema elettrico, sempre fisicamente collocata sul PCB della Sezione ingressi e riguardante la sezione delle commutazioni degli ingressi stessi e l'opzionale divisione x10 del segnale BF o TTL sotto misura. I relé sono di tipo reed SMD, molto veloci nella commutazione e silenziosissimi; consumano pochissimo, al punto da poter essere pilotati senza problemi direttamente dai pin del micro. La condizione iniziale all'accensione è quella di SWA-SWB-SWC allo stato LOW e quindi i tre relé RL1-RL2-RL3 sono in condizione di riposo; questa condizione, che è rappresentata nello schema elettrico, provoca l'accensione del LED verde LD1 (ingresso BF) e porta il

segnale, proveniente dall'ingresso BF, e denominato F_OUT_BF direttamente sul pin 2 (F_OUT) del connettore a 6 poli (3x2) destinato al collegamento col PCB della logica, come abbiamo descritto in precedenza. Il segnale arriverà quindi direttamente al micro per la sua misurazione. Da notare che lo stesso segnale viene comunque immesso nel divisore x10 U3A (pin 14), per uscire, appunto diviso x10, dal pin 3 dello stesso U3A, però nella condizione iniziale si ferma lì.

Premendo una prima volta il pulsante "ingresso" (PCB Logica), SWA e SWB vengono portati entrambi allo stato logico HIGH e si eccitano i relé RL1 e RL2; il primo scambio fa sì che il segnale prelevato sia ora quello in uscita dallo stadio TTL (F_OUT_TTL) e fa accendere il LED rosso LD2 (ingresso TTL); ovviamente si spegne quello verde (ingresso BF). Il secondo scambio, invece,

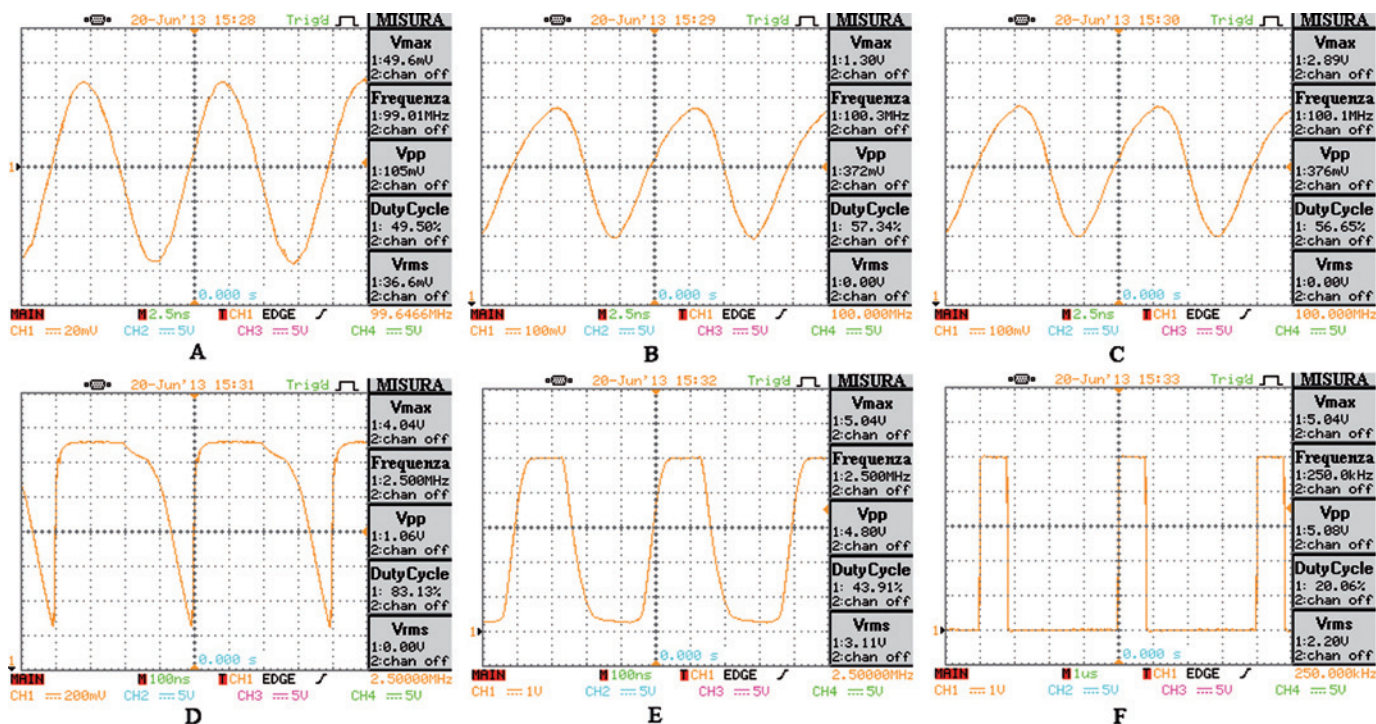


Fig. 7 - Il funzionamento dello stadio RF – misura di un segnale di 100 MHz 100 mVpp.

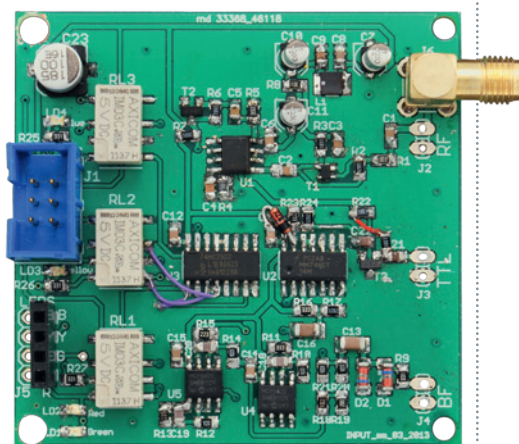
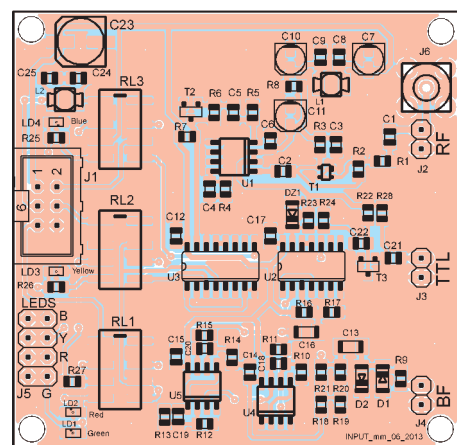
Elenco Componenti:

R1: 33 ohm (0805)
 R2: 4,7 kohm (0805)
 R3: 1,8 kohm (0805)
 R4: 200 kohm (0805)
 R5: 3,3 kohm (0805)
 R6: 20 kohm (0805)
 R7: 100 ohm (0805)
 R8: 10 ohm (0805)
 R9, R25, R26, R27: 330 ohm (0805)
 R10, R12, R14, R20, R21: 10 kohm (0805)
 R11, R16: 33 kohm (0805)
 R13, R15: 22 kohm (0805)
 R17, R24: 12 kohm (0805)
 R18: 1 kohm (0805)
 R19: 20 ohm (0805)
 R22, R28: 100 kohm (0805)
 R23: 27 kohm (0805)
 C1, C2, C3, C6, C8, C9, C12, C14, C15, C17: 100 nF ceramico (0805)
 C4: 1 nF ceramico (0805)
 C5, C24, C25: 220 nF ceramico (0805)
 C7, C10, C11: 1 μ F 16 VL elettrolitico (EUUD-4x5,8)
 C13, C16: 10 μ F ceramico (0805)
 C18÷C20: 1 pF ceramico (0805)

C21, C22: 1 μ F ceramico (0805)
 C23: 100 μ F 16 VL elettrolitico (EU153CLV-0605)
 U1: MC12080D
 U2: 74HCT14D
 U3: 74HC390D
 U4, U5: MCP662
 T1: BFP420 (SOT343)
 T2, T3: BC817 (SOT23)
 L1, L2: Induttanza 33 μ H (MS50)
 D1, D2: BAT48 (MINIMELF)
 DZ1: Zener 5,1V (SOD80C)
 LD1: LED verde (0805)
 LD2: LED rosso (0805)
 LD3: LED giallo (0805)
 LD4: LED blu (0805)
 RL1÷RL3: Relé reed G6J-2FL-Y

Varie:

- Connettore IDC 6 vie maschio
- Strip maschio 2 vie (3 pz.)
- Strip maschio 4 vie (2 pz.)
- Presa SMA da CS
- Zoccolo 14+14
- Circuito stampato



porta su F_OUT non più il segnale diretto ma quello diviso x10 ad opera di U3A; quindi appena si attiva l'ingresso TTL, automaticamente parte anche la divisione x10 (il motivo sarà approfondito al momento opportuno) e contemporaneamente si accende il LED giallo LD3 (divisore x10 BF/TTL). Premendo una seconda volta il pulsante "ingresso", SWA e SWB tornano allo stato LOW, mentre SWC va su HIGH; in queste condizioni viene commutata su F_OUT l'uscita dello stadio RF (segnale F_OUT_RF) e si accende il LED blu LD4. La commutazione toglie i 5V destinati agli altri tre LED1-2-3, che si spengono. Il funzionamento del pulsante portata è ciclico, quindi un'ulteriore pressione riporta lo stato dei relé e dei LED alla condizione iniziale (ingresso BF attivo). Abbiamo visto che il relé RL2, che attiva la divisione x10 del

segnale, viene attivato automaticamente ogni volta che si seleziona l'ingresso TTL, ma non è l'unica situazione in cui ciò accade; infatti, come già detto, il divisore x10 serve anche per il segnale BF. È il software che si occupa di gestire automaticamente questo relé (ne parleremo nella prossima puntata, quando affronteremo il firmware). Nello schema si possono notare i tre componenti C24-L2-C25: essi rappresentano un indispensabile filtro anti-RF sull'alimentazione del relé RL3. Questi relé, infatti, si sono rivelati leggermente sensibili a segnali RF di una certa intensità, per cui è la stessa bobina che capta il segnale che passa nei contatti e lo immette sul pin che lo controlla, andando a disturbare notevolmente il funzionamento del micro. Naturalmente il problema riguarda solo RL3, in quanto si trova posizionato molto vicino al tran-

sistor T3, sul cui collettore, come abbiamo visto, possono trovarsi frequenze fino a oltre 27 MHz con un'ampiezza di 4-5 Vpp. Il filtro funziona egregiamente, bloccando ogni residuo RF. Un'ultima nota riguarda l'integrato U3: abbiamo accennato al fatto che esso è composto da due blocchi separati, ognuno costituito da due divisori, uno x2 e l'altro x5, che si usano in cascata per ottenere la divisione x10. In genere si usano proprio nella sequenza x2-x5, ma questo genera in uscita un segnale con un duty-cycle del 20%, come è evidente osservando la tabella di verità dell'integrato. L'uscita Q3 (stadio x5) mostra come il rapporto tra H e L sia appunto di 2:10 (20 %). Questo, nel caso dell'ingresso TTL, rappresenta un problema perché il microcontrollore ATmega328P è in grado di leggere correttamente frequenze

COUNT	OUTPUTS			
	Q ₀	Q ₁	Q ₂	Q ₃
0	L	L	L	L
1	L	H	L	L
2	L	L	H	L
3	L	H	H	L
4	L	L	L	H
5	H	L	L	L
6	H	H	L	L
7	H	L	H	L
8	H	H	H	L
9	H	L	L	H

Fig. 8
Tabella di
verità del
74HC390.

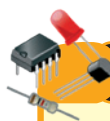
pari alla metà del clock, a patto che il segnale abbia un duty-cycle del 50%. In caso contrario la massima frequenza diventa esattamente uguale ad una percentuale, rispetto al clock, pari al duty-cycle del segnale. Quindi, nel nostro caso, al micro avremmo potuto applicare un segnale massimo di $16\text{MHz} \times 0,2 = 3,2\text{MHz}$. Questo valore non rappresenta un problema per l'ingresso RF, che, dividendo qualsiasi segnale $\times 400$ ed avendo come massimo valore garantito 1,1 GHz, porta al micro una frequenza massima di $1,1\text{GHz} / 400 = 2,75\text{MHz}$; per tale ragione l'U3B è stato usato nella tipica configurazione $\times 2 \times 5$. Invece tale duty-cycle avrebbe creato problemi all'ingresso TTL, il quale, avendo a valle solo un divisore $\times 10$, non ci avrebbe permesso di superare la soglia di 32 MHz. Osservando la Fig. 8 si evince come, semplicemente invertendo i due stadi tra loro, sia possibile ottenere un duty-cycle del 50% (infatti l'uscita Q0 dello stadio $\times 2$ ha un rapporto H-L di 1:1) e ciò ha permesso di aggirare il problema e raggiungere i 50 MHz (il limite, a questo punto diventa la frequenza operativa dei due integrati TTL usati in questo ingresso, tra cui lo stesso 74HC390). Detto ciò, passiamo al montaggio del circuito ingressi: per il cablaggio di questo PCB fate rigorosamente riferimento al piano di montaggio e non alla foto del PCB che, come si può notare, ritrae il prototipo finale, per il quale è stato realizzato un nuovo e definitivo PCB. Il piano di montaggio

fa appunto riferimento al PCB definitivo, che poi è quello che potrete scaricare dalla sezione download del sito. Al solito consigliamo di iniziare dai componenti passivi di piccolo spessore: resistenze, condensatori ceramici, impedenze; quindi i tre transistor e i cinque circuiti integrati. Vista la vicinanza dei componenti, per evitare che entrino in contatto tra loro è consigliabile applicare del flussante alle piazzole prima di saldarli. Bisogna fare molta attenzione alla saldatura dei tre transistor, che hanno dei pin davvero piccoli; quindi conviene aiutarci con una lente di ingrandimento. Per quanto riguarda gli integrati, la tecnica più semplice consiste nel posizionare correttamente il chip sulle piazzole e poi saldare uno qualsiasi dei pin di angolo (quelli più esterni e vicini agli spigoli del chip stesso); a quel punto sarà abbastanza comodo saldare tutti gli altri pin, senza timore di spostare il chip. Arrivati a questo punto si potrà procedere con i componenti più alti: condensatori elettrolitici, connettori, la presa SMA, i relé. Una nota a proposito dei LED: abbiamo previsto il montaggio di LED smd sul PCB ed il connettore J5 per il montaggio dei LED da 3 mm sul pannello frontale del contenitore. Poiché il collegamento dei LED da 3 mm è previsto solo alla fine del montaggio nel contenitore, in fase di test i quattro LED SMD potrebbero tornare utili; in particolare, riguardo il LED giallo LD3 non abbiamo previsto il suo montaggio sul pannello, visto che la sua gestione è assolutamente automatica, quindi converrebbe usare almeno quello smd per verificare il corretto funzionamento del divisore $\times 10$ BF/TTL. Non ci sono problemi a lasciare contemporaneamente collegati i LED SMD e quelli esterni da 3



Fig. 9 - Cavo flat
Ingressi-Logica.

mm; per chi volesse, abbiamo previsto l'uscita del LED giallo sul connettore J5, per il montaggio sul pannello frontale. Vista la notevole sensibilità di questo stadio e le elevate frequenze che lo interesseranno, completate tutte le saldature conviene ripulire il PCB dalle tracce del flussante, se lo avete usato, e del disossidante presente nello stagno. Allo scopo usate alcool isopropilico oppure solvente spray per flussante. Per realizzare il cavetto flat di connessione tra il PCB dello stadio degli ingressi e quello della logica, servono due connettori femmina tipo hatching 3×2 ad innesto obbligato, e circa 15 cm di cavo flat a 6 poli: il collegamento deve essere pinto-pin. Per filtrare la RF conviene far passare il cavetto all'interno di un idoneo anello di ferrite, compiendo un giro completo (Fig. 9) prima di crimpare i connettori. Bene, a questo punto avete pronti i tre PCB; nella seconda ed ultima puntata vedremo come montarli nel contenitore, spiegheremo il firmware ed il funzionamento del frequenzimetro. ■

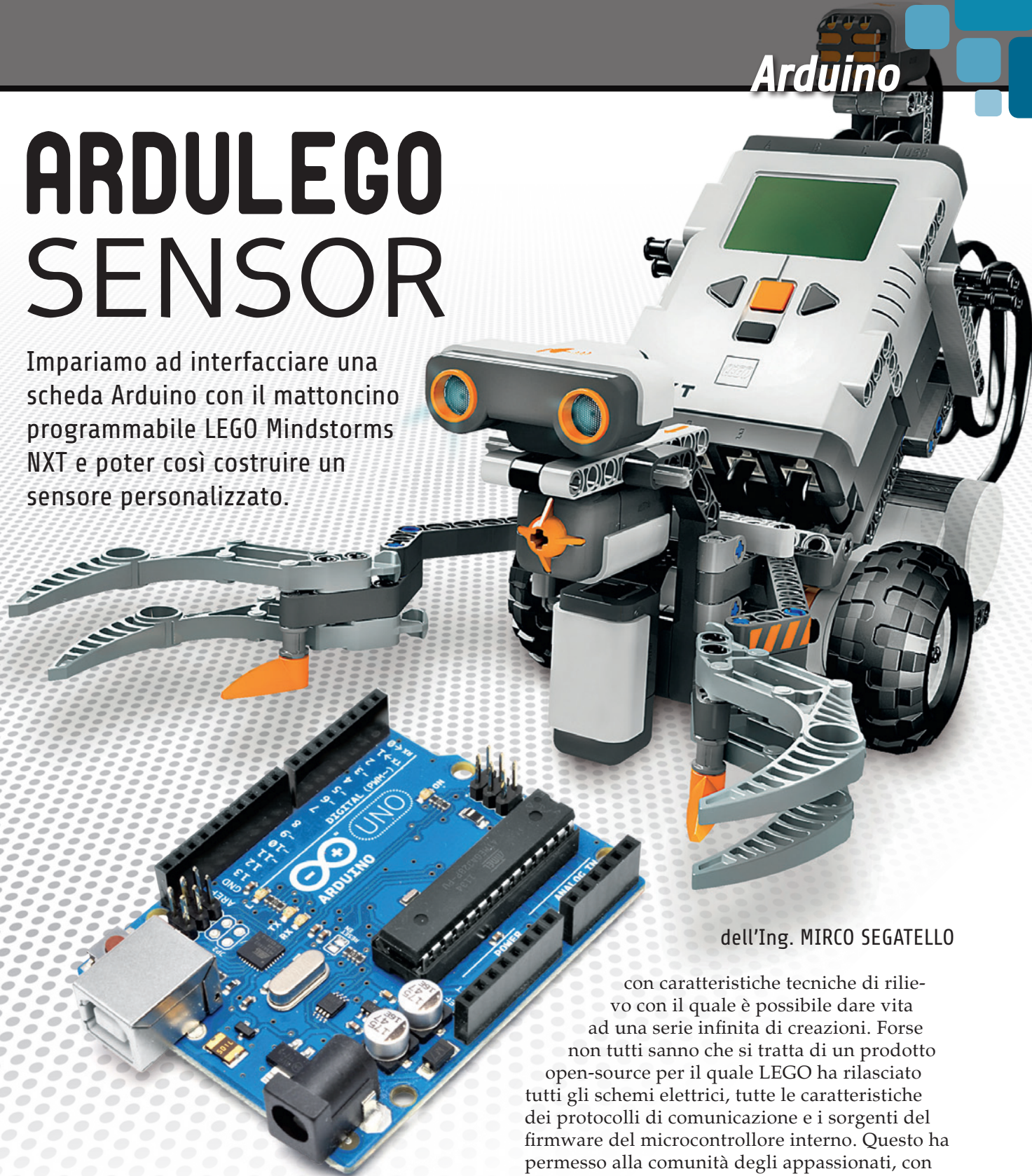


per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. I master dei circuiti stampati possono essere scaricati dal sito della rivista così come il firmware utilizzato per programmare il microcontrollore Atmel. Collegati subito a www.elettronicaonline.it !

ARDULEGO SENSOR

Impariamo ad interfacciare una scheda Arduino con il mattoncino programmabile LEGO Mindstorms NXT e poter così costruire un sensore personalizzato.



dell'Ing. MIRCO SEGATELLO

Il successo del mattoncino programmabile LEGO Mindstorms NXT è ormai di portata mondiale con impiego in ambito scolastico, per fini ludici e nelle gare di robotica di qualsiasi livello, dalle elementari all'università. Si tratta senza ombra di dubbio di un prodotto di qualità

con caratteristiche tecniche di rilievo con il quale è possibile dare vita ad una serie infinita di creazioni. Forse non tutti sanno che si tratta di un prodotto open-source per il quale LEGO ha rilasciato tutti gli schemi elettrici, tutte le caratteristiche dei protocolli di comunicazione e i sorgenti del firmware del microcontrollore interno. Questo ha permesso alla comunità degli appassionati, con il supporto della Rete, di ampliare le funzioni di base del mattoncino aggiungendo caratteristiche più avanzate o particolari non implementate nella versione commerciale. Soluzioni firmware sono state sviluppate per permettere di programmare il dispositivo con i più disparati sistemi di sviluppo, ma anche per l'hardware sono state

Fig. 1
LEGO
Mindstorm
NXT2.0.



sviluppare applicazioni specifiche, volte essenzialmente ad ampliare la gamma dei sensori. Infatti, per la versione NXT2.0, sono disponibili solo alcuni tipi di sensori suddivisi in due tipi di categorie: sensori base di tipo analogico e sensori avanzati che comunicano con il mattoncino digitalmente. In questo articolo descriviamo come sia possibile autocostruire un sensore "personale" da interfacciare con il mattoncino NXT e come possa essere gestito con i principali software di sviluppo per Mindstorms come il software base NXT-G e il compilatore C open source NXC.

L'unica difficoltà per chi vuole creare delle interfacce per gli NXT riguarda i connettori del mattoncino: si tratta di un particolare modello di connettore RJ12, con la linguetta di ancoraggio spostata su di un lato e non al centro come nelle normali spine telefoniche (Fig. 3). In commercio è possibile trovare, in negozi specializzati, il connettore femmina da circuito stampato compatibile NXT (Fig. 4), anche se non proprio a buon mercato. Per aggirare il problema ci sono essenzialmente due soluzioni: la prima consiste nel

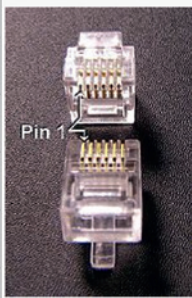
modificare il connettore dei cavetti dell'NXT togliendo la linguetta di ancoraggio, in modo che possa entrare all'interno di un connettore RJ12 standard. Il connettore RJ12 andrà saldato su di una piccola basetta millefori (oppure fatta ad hoc) assieme al classico strip maschio adatto all'inserimento in una breadboard (Fig. 5).

Per i collegamenti elettrici si faccia riferimento alla Fig. 2 che riporta la piedinatura del cavo con i colori.

La seconda soluzione prevede di tagliare il cavetto originale e saldare direttamente sui fili uno strip maschio; all'occorrenza i singoli contatti dello strip possono essere separati.

Adesso che abbiamo risolto il problema di adattamento dell'hardware, prendiamo in considerazione le caratteristiche elettriche del mattoncino NXT, in particolare l'alimentazione e gli ingressi. Analizzando lo schema elettrico riferito a un connettore di input (quelli indicati sull'NXT con le sigle 1,2,3,4) è possibile dedurne dettagliatamente il funzionamento. Ciascun connettore (Fig. 7) dispone di sei contatti elettrici facilmente identificabili dai colori dei fili del cavetto (Fig. 2), ciascuno con una specifica funzione. Partiamo dal contatto 1 del

Fig. 2 - Piedinatura dei connettori per NXT.

Pin	Name	Function	Color	Pin Numbering
1	ANALOG	Analog interface, +9V Supply	white	
2	GND	Ground	black	
3	GND	Ground	red	
4	IPOWERA	+4.3V Supply	green	
5	DIGIAI0	I ² C Clock (SCL), RS-485 B	yellow	
6	DIGIAI1	I ² C Data (SDA), RS-485 A	blue	

connettore RJ12 (filo bianco del cavo): è la linea di ingresso analogico con scala da 0 a 5 volt. Dall'analisi dello schema elettrico (Fig. 8) si nota che su questa linea è presente una resistenza di pull-up del valore di 10 kohm connessa ai 5V, seguita da una rete elettrica di protezione formata dalle resistenze R53 e R56 e dal diodo D12, dopodiché il segnale finisce all'ingresso analogico ADC_A0 del microcontrollore.

A causa della resistenza di pull-up è sconsigliabile inviare direttamente una tensione analogica a questa linea, mentre risulta più agevole connettere un resistore tra questo pin e GND, creando di fatto un partitore di tensione, utile per la misura del valore di un potenziometro. Il potenziometro potrebbe essere usato per conoscere la posizione assunta da un albero in rotazione così come avviene nei servomotori da modellismo, se invece il potenziometro fosse del tipo a slitta potremmo sapere la posizione di un organo in movimento longitudinale.

A questo pin fa capo anche la rete formata da tre transistor che ha lo scopo di fornire la tensione di alimentazione a 9V della batteria di alimentazione, decurtata dalla caduta di tensione dovuta a un diodo di protezione collegato in serie; questo valore dipende ovviamente dal livello di carica e dal tipo di

Fig. 3 - Differenza tra connettore RJ12 e NXT.



Fig. 4 - Connettore femmina per NXT.

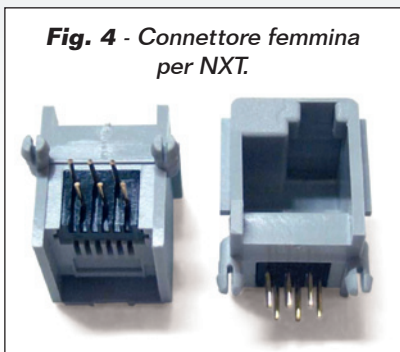


Fig. 5
Connettore RJ12 con adattatore autocostruito.

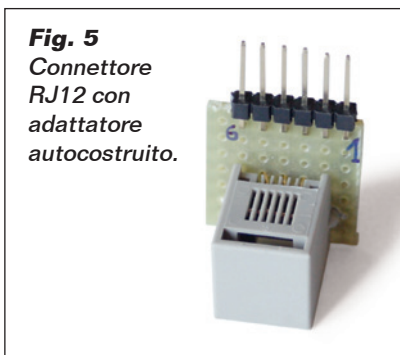


Fig. 6 - Cavetto NXT modificato con strip maschio per breadbord.

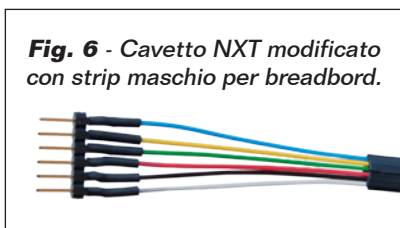


Fig. 7 - Il connettore d'ingresso dell'NXT.

batterie utilizzato. Notare come il transistor Q5A e la resistenza R33 formino un limitatore in corrente a protezione dello stadio nel caso di cortocircuito verso massa di questa linea. Nel caso della versione Education, l'alimentazione giunge da due celle LiPo che forniscono una tensione compresa tra 6V e 8,4V, mentre alimentandolo con 6 celle Ni-Mh la tensione sarà compresa tra 6V e 9V; alimentandolo invece con batterie alcaline a secco la tensione varierà tra 7V e 9V. L'abilitazione di questa tensione avviene da parte del micro agendo sulla linea I_ONA (TP21), ovviamente questa linea potrà essere usata o come ingresso analogico o come linea di alimentazione, ma non entrambe le funzioni contemporaneamente. La corrente è limitata a circa 18 mA, insufficienti per alimentare, ad esempio, una scheda Arduino. I pin 2 e 3 del connettore corrispondono ai fili di colore nero e rosso e sono entrambi il riferimento di massa GND. La linea 4 denominata IPOWER-A (colore verde) è la linea che fornisce alimentazione al circuito esterno; essa giunge da un apposito circuito studiato per

Fig. 8 - Schema elettrico relativo alla linea 1 del connettore RJ12.

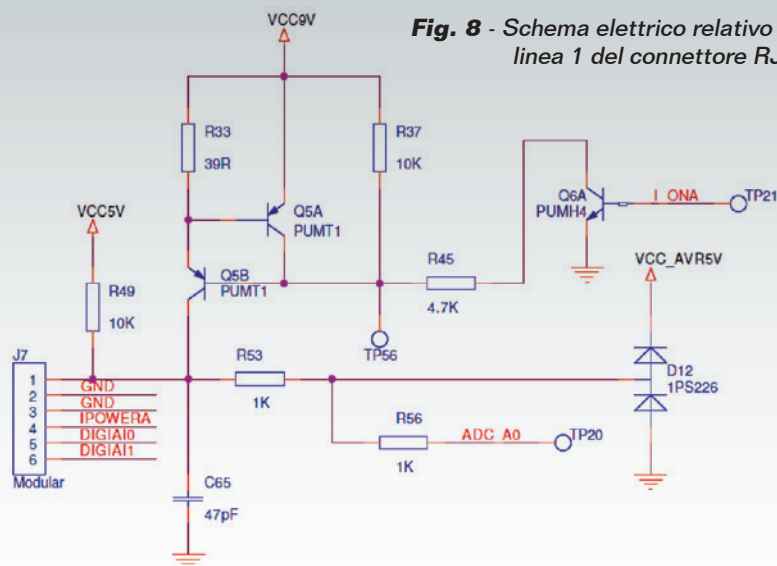
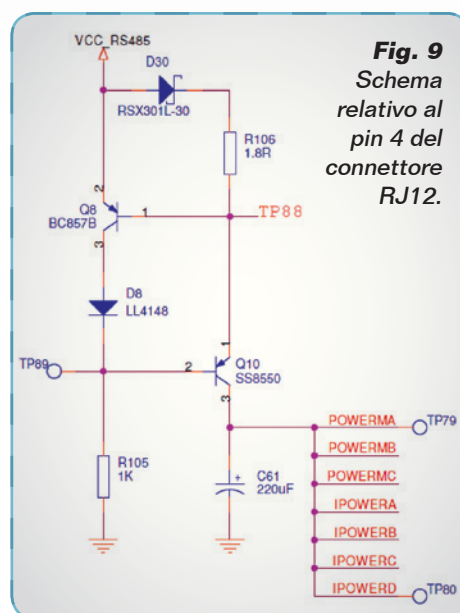


Fig. 9
Schema relativo al pin 4 del connettore RJ12.



alimentare i sensori (Fig. 9). Dall'analisi dello schema si nota che la rete è essenzialmente un limitatore in corrente a protezione dai cortocircuiti. Questo circuito viene alimentato con la tensione VCC_RS485 (Fig. 10) la quale giunge tramite un interruttore a stato solido dalla tensione VCC_SW (Fig. 11), fornita da un alimentatore switching. A causa delle varie protezioni, questa tensione è però leggermente inferiore ai 5V canonici e si attesta attorno ai 4,8V a vuoto, per scendere sino a 4,3V

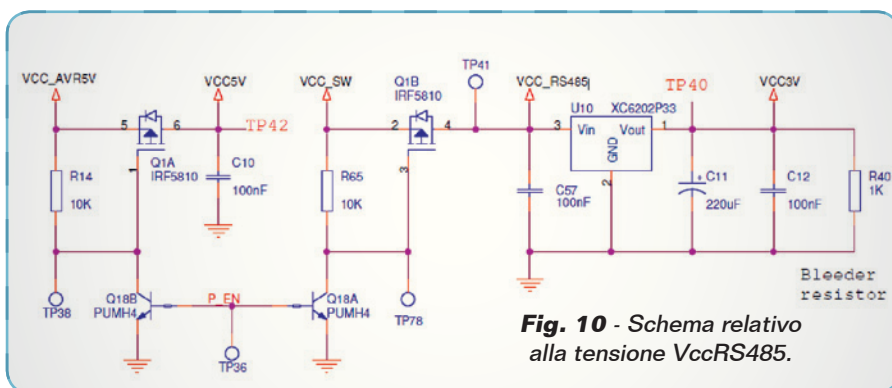


Fig. 10 - Schema relativo alla tensione VccRS485.

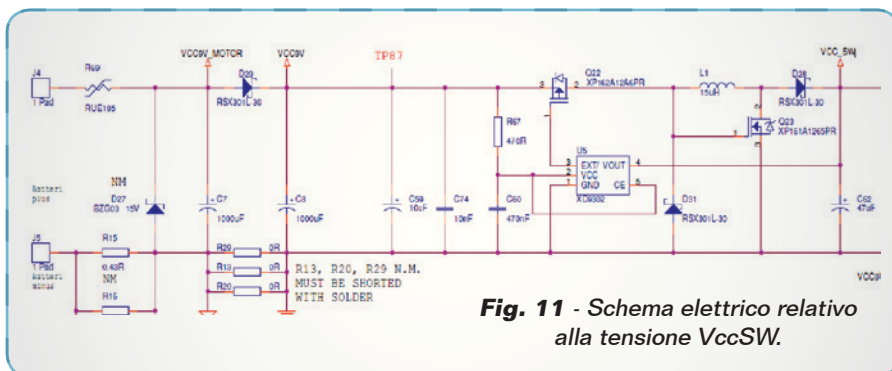


Fig. 11 - Schema elettrico relativo alla tensione VccSW.

nel caso si connettano diversi sensori all'NXT, motivo per cui in determinate situazioni e con batteria particolarmente scarica alcuni sensori con logica integrata potrebbero non funzionare correttamente.

Dobbiamo ammettere che i tecnici LEGO hanno fatto un lavoro molto approfondito nel progettare la parte elettronica di questo mattoncino programmabile, prevedendo anche un gran numero di protezioni che rendono di fatto l'NXT elettricamente molto robusto e affidabile.

Possiamo fare una semplice prova utilizzando una breadboard per connettere un trimmer all'NXT; per i collegamenti fate riferimento alla Fig. 12.

Il trimmer del valore di 1 kohm è usato come partitore di tensione rispetto la tensione di alimentazione fornita dal pin 4; il basso valore consente di ridurre l'influenza della resistenza di pull-up sulla linearità della misura. Sarebbe possibile usare solo due pin del trimmer (Fig.13) sfruttando la resistenza

di pull-up interna, in questo caso un valore di 10 kohm sarebbe da preferire; la lettura fornita non sarà proporzionale alla posizione del trimmer. A livello di programmazione con l'ambiente di sviluppo NXT-G è sufficiente utilizzare il blocco relativo al sensore di tocco, utilizzando la funzione di lettura RAW, che riporta esattamente il valore letto dal convertitore ADC con risoluzione a 10 bit (valori da 0 a 1023). Il primo blocco è relativo al sensore di tocco del quale viene sfruttata la lettura RAW, segue un blocco per la conversione del formato numerico in una stringa e quindi il blocco per la visualizzazione sul display. Ricordatevi di indicare la corretta porta di ingresso utilizzata: nel nostro caso la 4. Sul display vedrete semplicemente il valore numerico relativo alla lettura del convertitore ADC (Fig. 15). Per usare il compilatore NXC è sufficiente impostare correttamente la modalità di utilizzo della porta e quindi usare la funzione per leggere il valore

dell'ADC (Listato 1).

Questa modalità di funzionamento è la stessa del sensore di luce fornita assieme al kit NXT, dove il valore fornito è semplicemente la quantità di luce letta dal sensore riportata su una scala in percentuale.

Questa però non è l'unica modalità di funzionamento, soprattutto quando si utilizzano sensori più complessi che necessitano dell'invio di più informazioni al mattoncino programmabile; LEGO ha pensato a tutto e il connettore prevede anche i segnali per il BUS I²C standard, offrendo possibilità di sviluppo davvero interessanti. Le due linee 5 e 6 (filì giallo e blu) sono di tipo digitale e tramite un apposito circuito di protezione e traslazione vengono trasferite sugli ingressi digitali del micro (Fig. 16). Queste linee sono rispettivamente i segnali SCL e SCA del BUS I²C e le linee digitali del BUS di comunicazione RS485 (solo per la porta 4). Per utilizzare questa modalità di comunicazione è necessario disporre di un'opportuna logica che sia in grado di gestire il

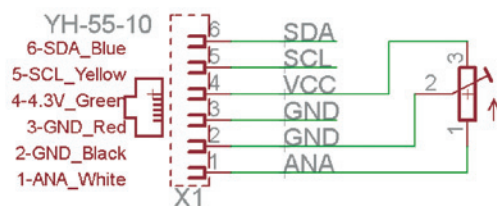


Fig. 12 - Collegamento di un trimmer alla porta di ingresso dell'NXT.

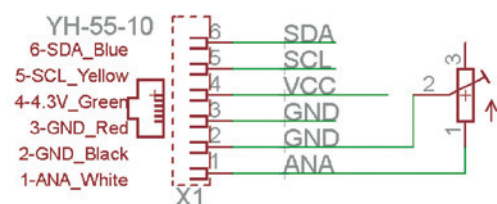


Fig. 13 - Collegamento alternativo di un trimmer alla porta di ingresso dell'NXT.

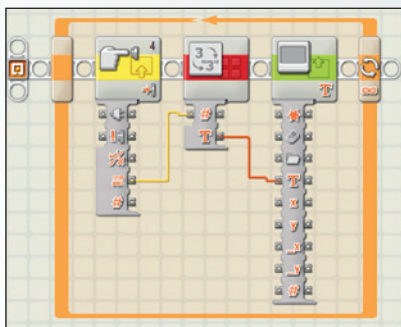


Fig. 15 - Display relativo alla lettura del trimmer.

BUS I²C sia dal punto di vista elettrico, sia per il protocollo di comunicazione. Per prima cosa è opportuno leggersi la documentazione rilasciata da LEGO che descrive in modo preciso il tipo di protocollo usato e fornisce le specifiche per l'interfaccia hardware; il file si chiama "LEGO MINDSTORMS NXT Hardware Developer Kit". Tra le indicazioni viene riportata la velocità del bus che è 9600 bit/s e la necessità di utilizzare esternamente delle resistenze di pull-up del valore di 82 kohm.

Fig. 14 - Programma per leggere il valore del trimmer.

Giusto per completezza possiamo dire che Arduino gestisce comunicazioni I²C fino a 400 kHz, caso in cui le resistenze di pull-up sono impostate al valore di 1,8 kohm. Il numero massimo di byte letti o scritti durante ogni singola operazione è pari a 16; viene inoltre specificato che nel caso siano connesse più periferiche in cascata sulla stessa porta, la resistenza totale di pull-up deve rimanere di 82 kohm. Per la versione NXT l'unico sensore che utilizza questa tecnologia è il sensore a ultrasuoni, ma molte aziende hanno provveduto a introdurre sul mercato sensori compatibili come giroscopi e accelerometri. Per poter sfruttare questo sistema di comunicazione la via più semplice è quella di utilizzare una piattaforma di sviluppo come Arduino che offre già nativamente la porta di comunicazione I²C e le librerie per poterla gestire. Interfaciare Arduino a una porta di ingresso dell'NXT è quindi molto semplice e richiede solo due resistenze aggiuntive; lo schema elettrico è riportato in Fig.17. Le linee 2 e 3 (filo rosso e nero, o anche solo una di esse) devono essere portate al GND di Ardui-

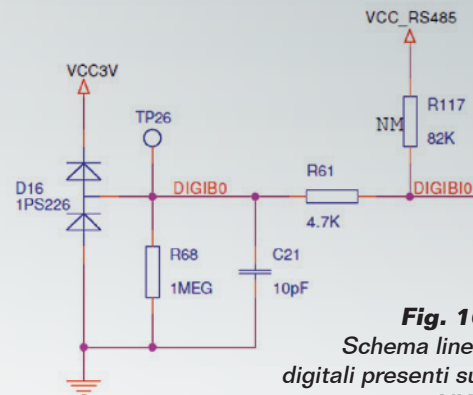


Fig. 16
Schema linee digitali presenti sul connettore NXT.

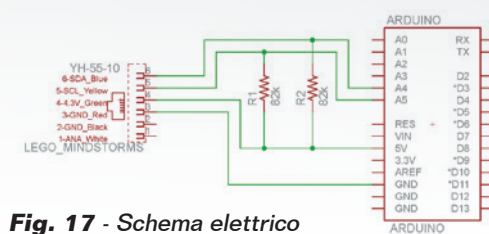


Fig. 17 - Schema elettrico di collegamento Arduino con NXT.

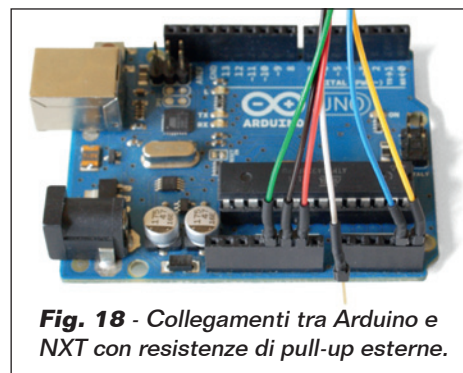


Fig. 18 - Collegamenti tra Arduino e NXT con resistenze di pull-up esterne.

no. La linea 4, filo verde, funge da alimentazione per Arduino e va portata al pin 5V di Arduino. Di fatto Arduino sarà alimentato da una tensione inferiore ai 5V perché la tensione sul pin 4 (filo verde) è di circa 4,6V. Potete alimentare Arduino anche tramite il pin VIN ma a causa della caduta di tensione introdotta dallo stabilizzatore di tensione il microcontrollore di Arduino sarà alimentato a soli 3,7V. Non è possibile utilizzare la linea 1 di alimentazione a 9 V perché non fornisce sufficiente corrente per alimentare un'intera scheda Arduino. La linea 5 (filo blu) va portata

Listato 1 - Lettura sensore analogico con NXE.

```
// Lettura ADC V1.0

task main()
{
    // imposta il tipo di sensore (contatto)
    SetSensorType (IN_4, IN_TYPE_SWITCH);
    // imposta il tipo di lettura (valore ADC)
    SetSensorMode (IN_4, IN_MODE_RAW);
    while(TRUE)
    {
        TextOut(0,LCD_LINE5, "Lettura tensione");
        ClearLine(LCD_LINE6);
        // la funzione SENSOR_4 ritorna il valore dell'ADC
        NumOut(0, LCD_LINE6, SENSOR_4);
        Wait (250);
    }
}
```


Ovviamente è interessante che Arduino acquisisca dei dati esterni da inviare poi all'NXT; per questo motivo inseriamo nello schema anche un trimmer in grado di fornire una tensione variabile all'ingresso analogico A0 di Arduino. Ovviamente questo è solo un semplice esempio, ma potete far leggere ad

Ogni dispositivo connesso al bus ha un indirizzo univoco a 7 bit ed al massimo possono essere gestiti 127 slave; l'ottavo bit specifica se è attiva una lettura o una scrittura verso il dispositivo. È il master che gestisce

Analizziamo ora attentamente il listato in modo che chiunque possa modificarlo secondo le proprie esigenze. Nella proce-

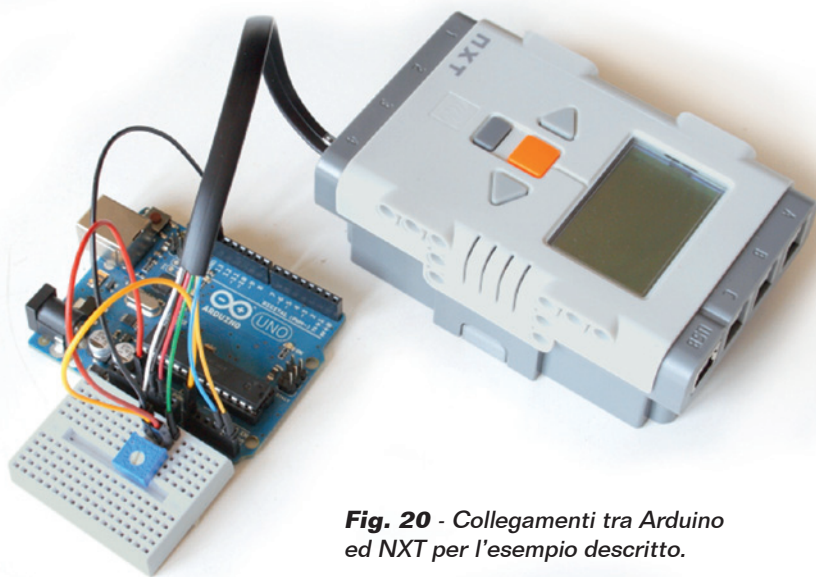


Fig. 20 - Collegamenti tra Arduino ed NXT per l'esempio descritto.

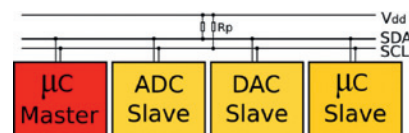


Fig. 21 - BUS I²C e periferiche.

dura di *Setup* viene impostata la comunicazione seriale standard a 9600 baud, utile come diagnostica ma non necessaria al funzionamento con NXT. Lasciando Arduino connesso alla porta USB del PC, tramite la funzione *serial-monitor* di Arduino sarà possibile vedere i byte effettivamente ricevuti. Segue la configurazione della porta di comunicazione I²C specificando l'indirizzo della periferica (0x01) e le due funzioni che saranno richiamate in automatico qualora si verifichino determinati eventi; ovviamente è possibile utilizzare qualsiasi indirizzo I²C a piacere. Sulla stessa porta possono quindi coesistere più sensori purché ciascuno con indirizzo univoco; l'unico limite è la massima corrente prelevabile dalla porta per alimentare i sensori.

La funzione *receiveEvent* viene richiamata quando il master (NXT) apre una comunicazione con la periferica e invia dei dati; ovviamente l'indirizzo di periferica inviato da NXT coinciderà con l'indirizzo di Arduino. Useremo questa funzione per leggere i byte in arrivo e memorizzare l'ultimo byte che conterrà la locazione di memoria alla quale NXT vuole accedere (variabile *read_data*).

Successivamente viene richiamata la procedura *requestEvent* che implementa la risposta da parte di Arduino, essa corrisponderà all'invio di uno o più byte a seconda del valore della variabile *read_data* in accordo con la mappatura degli indirizzi fornita da LEGO.

Appena richiamata questa procedura viene letto il valore della tensione all'ingresso analogico A0.

Ecco quindi che ricevuto il valore 0x00, Arduino risponde

Tabella 1 - Byte trasmessi da NXT alla periferica.

Comando	byte0	byte1	byte2
Lettura versione	Indirizzo periferica	0x00	R+0x03
Lettura ID produttore	Indirizzo periferica	0x08	R+0x03
Lettura tipo sensore	Indirizzo periferica	0x10	R+0x03
Lettura zero	Indirizzo periferica	0x18	R+0x03
Lettura fattore di scala	Indirizzo periferica	0x19	R+0x03
Lettura divisione di scala	Indirizzo periferica	0x1A	R+0x03
Lettura unità di misura	Indirizzo periferica	0x1B	R+0x03
Lettura variabile1	Indirizzo periferica	0x40	R+0x03
Lettura variabile2	Indirizzo periferica	0x41	
---	---		
Comando 1	Indirizzo periferica	0x80	0xFF
Comando 2	Indirizzo periferica	0x81	0xFF

Listato 2 - Sketch per Arduino *customsensornxt.ino*

```
// CustomSensorNXT V1.0
// Sensore per NXT via I2C, lettura tensione AN0.
// su NXT si usi customsensornxt.nxc

#include <Wire.h>

// Indirizzo Arduino, per NXT questo valore va traslato
// (Arduino) 00000001 -> (NXT) 00000010 (0x02)

#define ADDRESS 0x01

byte read_data;

void setup() {

    Serial.begin(9600);

    // Configura I2C
    // Le resistenze di pull-up vengono abilitate dalla libreria wire
    Wire.begin(ADDRESS); // Imposta indirizzo Arduino
    Wire.onReceive(receiveEvent); // Ricezione dati da NXT
    Wire.onRequest(requestEvent); // Inoltro dati verso NXT

    // Disabilita R di pull-up
    digitalWrite(SDA, 0);
    digitalWrite(SCL, 0);
}

void loop()
{
}

// Funzione richiamata quando sono ricevuti dati da NXT
void receiveEvent(int byteIn)
{
    read_data = byteIn;
    Serial.println(read_data, BIN);
    if (Wire.available() > 0 )
    {
        read_data= Wire.read(); // memorizza l'ultimo byte ricevuto
        Serial.println(read_data, BIN); // Visualizza dati ricevuti
    }
}

// Procedura richiamata quando Arduino risponde ad NXT
void requestEvent()
{
    // Legge il valore dell'ADC su A0
    int ADCValue = analogRead(A0);
    byte upper_eight_bit;
    byte low_two_bit;

    switch (read_data) {
```

(Continua)

Listato 2 - Segue

```

case 0x00:
    Wire.write("V1.0"); // Versione del sensore
    break;
case 0x08:
    Wire.write("ElettronicaIN"); // Nome del produttore
    break;
case 0x10:
    Wire.write("Custom"); // Tipo di sensore Acc, Gyro ...
    break;
case 0x41:
    Wire.write("Hello"); // Esempio di generico testo
    break;
case 0x42:
    // Ritorna gli 8 bit più significativi (ADC in 8 bit)
    upper_eight_bit = ADCValue>>2;
    Wire.write(upper_eight_bit);
    break;
case 0x43:
    // Ritorna i 2 bit meno significativi
    low_two_bit = ADCValue & B00000011;
    Wire.write(low_two_bit);
    break;
case 0x44:
    // invia il valore dell'ADC su 2 byte (10bit)
    byte sendbuffer[] = {0, 0};
    sendbuffer[0] = ADCValue & B00000011;
    sendbuffer[1] = ADCValue>>2;
    Wire.write(sendbuffer, 2);
}
}

```

con quattro byte contenenti la versione del progetto ovvero "V1.0"; con il valore 0x08 invieremo i byte contenenti il nome del produttore "ElettronicaIN" e con il valore 0x10 invieremo il tipo di sensore "Custom". Non si tratta di dati fondamentali perché il sensore lo abbiamo fatto noi e quindi li conosciamo perfettamente. Con la ricezione del valore 0x40 si risponde semplicemente con la stringa "Hello".

Il nostro scopo è quello di leggere il valore analogico

all'ingresso A0 di Arduino e di renderlo disponibile all'NXT, per questo i successivi indirizzi contengono i dati effettivamente utili all'applicazione.

Se viene ricevuto il valore 0x42, Arduino risponderà con il valore dell'ADC scalato però a otto bit ed inviato come valore di tipo byte; vedremo che questo formato è più semplice da gestire con il software NXT-G. L'indirizzo successivo contiene i rimanenti due bit in modo che con due letture separate si possano leggere tutti i dieci bit del convertitore

ADC. In **Tabella 2** è riportato un esempio nel quale il convertitore ADC fornisce il dato numerico a 10 bit (1100100101) con valore 805, il quale viene scomposto in un primo byte contenente i primi otto bit (11001001) corrispondenti al valore numerico 201, ed un secondo byte contenente i primi due bit (01).

All'indirizzo 0x44 sarà invece disponibile l'intero dato a 10 bit dell'ADC, leggibile con una sola operazione di lettura; il dato scomposto in due byte sarà poi ricomposto dal software lato NXT.

In definitiva la mappatura degli indirizzi per il nostro sensore è quella riportata in **Tabella 3**.

A questo punto dobbiamo occuparci di come gestire la nuova periferica lato Mindstorms e nel fare questo abbiamo preso in considerazione i due linguaggi di programmazione più diffusi: il compilatore open source NXC e l'ambiente di sviluppo fornito di serie chiamato NXT-G. Per poter sfruttare appieno tutte le possibilità offerte dalla comunicazione I²C cominciamo dal compilatore NXC che mette a disposizione tutta una serie di comandi dedicati (vedasi il **Listato 3**). In NXC la riga chiave per definire un sensore I²C è: `SetSensorLowSpeed(ARDUINO_PORT);`

Il termine *lowSpeed* serve proprio per specificare una periferica I²C a bassa velocità (9600 bit/s) il cui parametro `ARDUINO_PORT` è la porta dell'NXT alla quale è connesso Arduino. Abbiamo implementato alcune procedure, ciascuna specifica per la lettura di un diverso tipo di dato; ad esempio `ShowSensorInfo` permette di visualizzare sul display dell'NXT il produttore, il tipo di sensore e la versione così come specificati nello sketch di Arduino.

Tabella 2 - Suddivisione in due byte del valore dell'ADC.

8 bit più significativi								2 bit meno significativi	
1	1	0	0	1	0	0	1	0	1

Tabella 3 - Mappatura indirizzi Arduino.

Contenuto	Tipo	Indirizzo
versione	Stringa (4 byte)	0x00
ID produttore	Stringa (13 byte)	0x08
tipo sensore	Stringa (6 byte)	0x10
Testo	Stringa (5 byte)	0x41
ADC 8 bit più significativi	Byte	0x42
ADC 2 bit meno significativi	Byte	0x43
ADC 10 bit	2xByte	0x44

Listato 3 - Programma per NXT customsensornxt.nxc

```
// Custom sensor per ARDUINO NXT

#define ARDUINO_PORT S4 // Porta di Arduino
#define ARDUINO_ADDRESS 0x01 // Indirizzo di Arduino
#define ADDRESS_SEND (ARDUINO_ADDRESS << 1) // shift indirizzo per compatibilità con NXT
#define ARDUINO_TXT_ADR 0x41 // Arduino registro per TEXT
#define ARDUINO_BYTE_ADR 0x42 // Arduino registro per BYTE type
#define ARDUINO_INT_ADR 0x44 // Arduino registro per INTEGER type

#define I2C_ADDR_DEFAULT 0x02 // indirizzo I2CDevice
#define I2C_REG_VERSION 0x00 // versione del sensore
#define I2C_REG_VENDOR_ID 0x08 // nome del produttore
#define I2C_REG_DEVICE_ID 0x10 // tipo di sensore acc, gyro, ...
#define I2C_REG_CMD 0x41

string i2cReadString(byte port, byte adr, byte reg, byte cnt)
{
    // legge una stringa
    string temp;
    byte outbuf[];
    byte cmdbuf[];
    ArrayBuild(cmdbuf, adr, reg);
    byte nByteReady = 0;
    if(I2CBytes(port, cmdbuf, cnt, outbuf))
    {
        temp = ByteArrayToStr(outbuf);
    }
    return temp;
}

byte i2cReadByte(byte port, byte adr, byte reg)
{
    // legge un valore byte
    byte cnt = 1;
    byte value;
    byte outbuf[];
    byte cmdbuf[];
    ArrayBuild(cmdbuf, adr, reg);
    byte nByteReady = 0;
    if(I2CBytes(port, cmdbuf, cnt, outbuf))
    {
        value = outbuf[0];
    }
    return value;
}

int i2cReadInt(byte port, byte adr, byte reg)
{
    // legge un valore intero 2byte
    byte cnt = 2;
    int value;
    byte outbuf[];
    byte cmdbuf[];
    ArrayBuild(cmdbuf, adr, reg);
    byte nByteReady = 0;
    if(I2CBytes(port, cmdbuf, cnt, outbuf))
    {
        value = outbuf[0]+4*outbuf[1];
    }
    return value;
}

void ShowSensorInfo(byte port, byte address)
{
    ClearScreen();
    TextOut(0, LCD_LINE1, i2cReadString(port, address, I2C_REG_DEVICE_ID, 15), false);
    TextOut(0, LCD_LINE2, i2cReadString(port, address, I2C_REG_VENDOR_ID, 15), false);
    TextOut(0, LCD_LINE3, i2cReadString(port, address, I2C_REG_VERSION, 15), false);
}

task main() {
    SetSensorLowSpeed(ARDUINO_PORT);
    ShowSensorInfo(ARDUINO_PORT, ADDRESS_SEND);

    // Show sensor info, funzioni alternative
    /*
    TextOut(0, LCD_LINE1, I2CDeviceInfo(ARDUINO_PORT, I2C_ADDR_DEFAULT, I2C_REG_DEVICE_ID));
    TextOut(0, LCD_LINE2, I2CDeviceInfo(ARDUINO_PORT, I2C_ADDR_DEFAULT, I2C_REG_VENDOR_ID));
    TextOut(0, LCD_LINE3, I2CDeviceInfo(ARDUINO_PORT, I2C_ADDR_DEFAULT, I2C_REG_VERSION));
    */
}
```

(Continua)

Listato 3 - Segue

```

*/

// check sensor every 100msec
while(true)
{
    Wait (100);
    // legge una stringa
    // specificare un numero di caratteri uguale o superiore a quelli inviati da Arduino
    TextOut(0, LCD_LINE6, i2cReadString(ARDUINO_PORT, ADDRESS_SEND, ARDUINO_TXT_ADR, 5), false);

    ClearLine(LCD_LINE7);
    // legge solo gli 8 bit più significativi
    NumOut(0, LCD_LINE7, i2cReadByte(ARDUINO_PORT, ADDRESS_SEND, ARDUINO_BYTE_ADR), false);

    ClearLine(LCD_LINE8);
    // legge il valore a 10bit
    NumOut(0, LCD_LINE8, i2cReadInt(ARDUINO_PORT, ADDRESS_SEND, ARDUINO_INT_ADR), false);
}
}

```



Fig. 22 - Display NXT con programma per NXC.

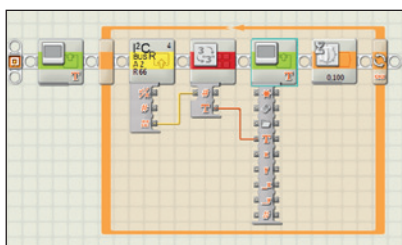


Fig. 23 - Programma per NXT-G per gestire Arduino.

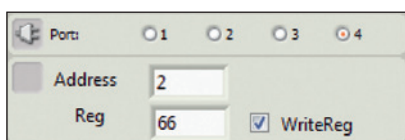


Fig. 24 - Definizione parametri per il blocco IIC_Read.



Fig. 25 - Display NXT con programma per NXT-G.

L'istruzione che permette di leggere una stringa da Arduino è `i2cReadString(ARDUINO_PORT, ADDRESS_SEND, ARDUINO_TXT_ADR, 5)` nella quale è sufficiente specificare la porta utilizzata, l'indirizzo di Arduino, l'indirizzo del registro ove si trova il dato da leggere ed il numero di caratteri da leggere. Notare come, nelle righe di *define* ad inizio programma, vengono impostati i valori di tutti i registri interessati mentre l'istruzione `ARDUINO_ADDRESS << 1` consente di traslare di un bit l'indirizzo specificato per Arduino (0x01) nel formato a 7 bit corrispondente al valore 0x02, valore di default per i sensori LEGO. Il sottoprogramma `i2cReadByte` legge un valore di tipo byte, passati come parametri, la porta (`ARDUINO_PORT`), l'indirizzo di Arduino (`ADDRESS_SEND`) e l'indirizzo del registro che contiene il dato (`ARDUINO_BYTE_ADR`). Il sottoprogramma `i2cReadInt` legge invece una variabile di tipo *integer*, ovvero ricompone i due byte del valore dell'ADC in una variabile di tipo *integer*. In Fig. 22 il display dell'NXT con il valore del sensore da noi realizzato. Regolando il trimmer vedrete gli ultimi due numeri cambiare da 0 a 255 per la variabile byte e da 0 a 1023 (10 bit) per la variabile *integer*.

L'ambiente di sviluppo NXT-

G non gestisce nativamente le periferiche I²C ma è sufficiente importare un blocco aggiuntivo, denominato "I²C Read block version 1.0b" (file `IIC_Read.zip`), per aggiungere la funzionalità di lettura da periferiche I²C. È sufficiente utilizzare il blocco `IIC_Read` per leggere dalla periferica, anche se con il limite della lettura di un byte alla volta.

Per funzionare correttamente il blocco richiede l'impostazione di tre parametri: la porta a cui è connessa la periferica, l'indirizzo della periferica e l'indirizzo del registro contenente il dato da leggere e tutti specificati come numeri decimali. Nel nostro caso la periferica è connessa alla porta 4, ha l'indirizzo 2 e il registro da leggere si trova all'indirizzo 0x42 ovvero 66 in decimale; ricordatevi la spunta su `WriteReg`.

Il valore ritornato è un byte contenente il valore dell'ADC con risoluzione a otto bit (0-255) che viene successivamente convertito in stringa e quindi visualizzato sul display. Un piccolo ritardo è necessario affinché sia avviata una nuova lettura solo quando è conclusa quella precedente. Nulla di complicato, come avete visto, ma questa soluzione vi offre l'opportunità di espandere il vostro NXT quando avete la necessità di un particolare sensore non disponibile nel catalogo LEGO Mindstorms.



WORLD SOLAR
CHALLENGE

WORLD SOLAR CHALLENGE

La competizione prenderà il via il 6 ottobre per concludersi il 13 e vedrà il debutto dell'italiana Emilia 3. Tremila km lungo l'Australia per testare le tecnologie più estreme in fatto di trazione elettrica ad alimentazione solare.



di DAVIDE SCULLINO

Da Darwin ad Adelaide, da nord a sud dell'Australia, sono circa 3.000 chilometri: una bella traversata, figurarsi a compierla con auto elettriche alimentate dall'energia del sole. Eppure World Solar Challenge è proprio questo: una competizione tra equipaggi a bordo di vetture elettriche ad alimentazione fotovoltaica, realizzate da team provenienti da tutto il mondo. Una consuetudine che si ripete da 26 anni e che ha assunto un'eco tanto rilevante da incentivare la nascita di eventi simili come la SA Solar Challenge, disputatasi lo scorso anno in Sudafrica.

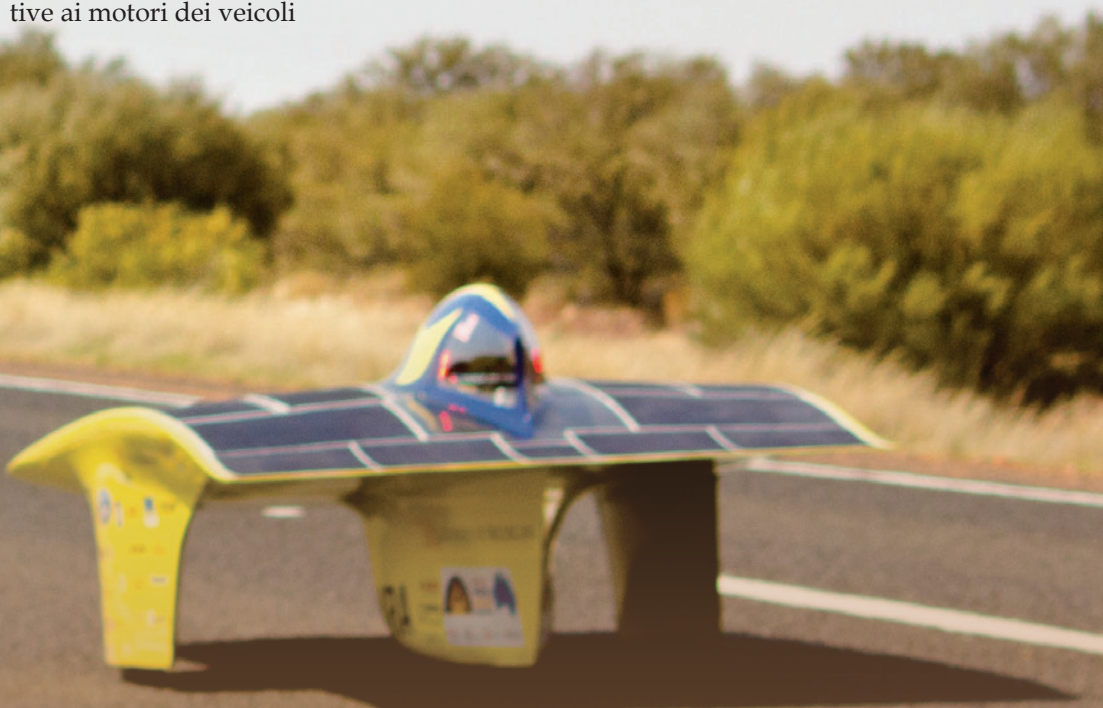
Da sempre seguiamo la World Solar Challenge con grande interesse, non solo per il fascino che esercita il vedere auto futuristiche attraversare un continente affascinante come l'Australia e percorrere in pochi giorni una distanza sulla strada pari al doppio di quella che c'è tra il punto più a nord e quello più a sud del nostro "stivale", ma anche perché uno degli equipaggi è quello dell'italiana Onda Solare (www.ondasolare.com), che quest'anno metterà in pista la sua più recente Emilia 3.

La competizione si tiene ogni due anni e l'edizione 2013 proporrà novità nel Regolamento Tecnico, che impone quattro

ruote invece delle tre dell'edizione 2011 e un cono di visibilità per il pilota all'interno del quale non ci devono essere parti di carrozzeria. Altra novità è la sponsorizzazione da parte di Bridgestone, marchio leader di pneumatici per auto, che equipaggerà le vetture in gara.

Nata nel 1982 da un'idea di Hans Tholstrup, World Solar Challenge ha preso ufficialmente il via nel 1987 e continua a mostrare al mondo lo sviluppo delle tecnologie alternative ai motori dei veicoli

convenzionali. Le automobili iscritte alla gara parteciperanno, in base alle loro caratteristiche, ad una delle tre classi di competizione: *Challenger*, *Michelin Cruiser* e *GoPro Adventure*. La Challenger Class, cui parteciperà l'ita-



liana Emilia 3, è la più futuristica e tecnologica: le vetture sono dei gioielli di aerodinamica e devono avere 4 ruote, un pilota, lunghezza massima di 4,5 metri, larghezza massima di 1,8 m, non più di 6 m² di pannelli solari. Questa

classe prevede una sola grande tappa (Darwin-Adelaide) da percorrere esclusivamente con energia solare risparmiando il più possibile energia, nel senso che vincerà la vettura che percorrerà la distanza prevista con la velo-

Il percorso

Le vetture partecipanti alla World Solar Challenge partono da Darwin e devono arrivare nel minor tempo possibile ad Adelaide: circa 3.000 chilometri più a sud. La sfida è sulla gestione dell'energia elettrica: sebbene un'automobile con motorizzazione da 1 kW completerebbe il percorso in 50 ore, richiedendo quindi 50 kWh, durante il rifornimento le vetture possono ricevere solo 5 kWh; il resto devono ricavarlo da sole mediante i pannelli fotovoltaici o recuperandolo in rallentamento sfruttando l'energia cinetica.

La classe Challenger (la più estrema) consiste in una sola lunga tappa da Darwin ad Adelaide. Una volta che le squadre hanno lasciato Darwin, hanno tempo fino alle 17.00 (perché la luce solare può essere sfruttata bene fino a quell'ora) per coprire quanti più chilometri possibile, perché a quell'ora si devono fermare e accampare nel deserto. Le altre classi hanno esigenze



diverse, ma tutte le squadre devono essere pienamente autosufficienti. Durante il viaggio ci sono sette punti di controllo obbligatori, in cui si cambiano gli osservatori e i team manager possono aggiornarsi con le ultime informazioni sul meteo e la loro posizione in classifica. Ai check-point le squadre possono solo effettuare il controllo della pressione dei pneumatici e la pulizia del veicolo.

città media più alta a parità di consumo. Nel 2011, la categoria Challenger Class è stata vinta da Tokai Challenger del team Tokai University (Giappone), che ha fatto registrare una velocità media di 91,54 km/h; al secondo posto si è piazzata la vettura Nuna 6 del team Nuon (Olanda) con una velocità media di 88,6 km/h, mentre terza classificata è stata la University of Michigan (U.S.A.) con la propria vettura Quantum, che ha marciato ad una velocità media di 84,33 km/h.

La Michelin Cruiser Class è destinata a veicoli più "stradali", aventi 4 ruote e posto per un pilota e un passeggero. La sfida prevede tappe con soste prestabilite, durante le quali, la notte, avverrà la ricarica delle batterie nei check-point lungo il percorso. Lo scopo di questa categoria è proporre vetture stradali omologabili nei Paesi di origine dei team (proponibili come alternative alle auto attualmente in circolazione) attraverso un test dove le auto in gara vengono utilizzate come si farebbe con auto qualsiasi, quindi rifornite periodicamente e testate sul piano dell'autonomia di marcia.

La GoPro Adventure, infine, è riservata a veicoli che avevano partecipato alle edizioni passate: le vetture avranno 3 ruote, un pilota, dimensioni massime di 5 metri per 1,8 m, con 6m² di pannelli solari fotovoltaici. Questa categoria vuole dare importanza all'esperienza: permette di tornare a gareggiare anche a coloro i quali non hanno avuto i mezzi per costruire una vettura rispondente al nuovo regolamento della World Solar Challenge.

I PARTECIPANTI

Quest'anno la competizione vedrà misurarsi, nelle tre classi previste, 44 team provenienti da 25 nazioni; per la Challenger Class si contenderanno il titolo le seguenti squadre: Anadolu Solar Team, Blue Sky Solar Racing, Cambridge University Eco Racing, EAFIT-EPM Solar Car Team, Eco Solar Breizh, Ecole de technologie superieure, Hachinohe Institute of Technology, ITS Solar Car Racing Team, ITU Solar Car Team, JU Solar Team, Kanazawa Institute of Technology YUMEKOBO, Kogakuin University Solar Vehicle Project, KUST (Kookmin University Solar

car Team), Nuon Solar Team, Onda Solare, Punch Powertrain Solar Team, Seraaj Solar Car Team, Solar Energy Racers, Solar Team Twente, Stanford Solar Car Project, Sun Shuttle, SunSPEC, Team Arrow, Tokai University, Uludag University Mechanical Community, UMPSolar, University of Michigan Solar Car Team, UWS Solar Car.

La Michelin Cruiser Class vedrà invece affrontarsi i team Apollo Solar Car Team, Goko High School, Hochschule Bochum SolarCar Team, Solar Team Eindhoven, TAFE SA Solar Spirit, University of Calgary Solar Car Team, University of Minnesota, UNSW Solar Racing Team-Sunswift. Infine, alla GoPro Adventure Class parteciperanno i team Antakari, Aurora Vehicle Association, IVE Solar Car Team, KAIT WSC Project, Mississippi Choctaw High School, SIKAT Solar Philippines, Team Solaris.

L'ITALIA IN GARA

Il team italiano che parteciperà alla competizione non è nuovo a questa manifestazione: è Onda Solare, squadra formata da docenti, studenti e ricercatori dell'Università di Bologna e del CNR che si occupano della progettazione, ma anche da artigiani, operai e imprese emiliane (tra cui anche Ferrari) che per raccogliere i fondi hanno fatto ricorso al crowdfunding tramite www.gofundme.com.

Onda solare nasce nel 2005 e finora ha prodotto tre veicoli: Emilia 1 (2005), giunta al terzo posto nel World Solar Cycling Challenge; Emilia 2, che ha partecipato all'edizione 2011 della World Solar Challenge, ed Emilia 3, che gareggerà quest'anno. Emilia 3 riprende il progetto base di Emilia 2, adeguato al nuovo regolamento tecnico della

competizione, che ha costretto i team a ripartire quasi da zero per quanto riguarda i calcoli sull'aerodinamica (lo studio aerodinamico è stato fatto esclusivamente al computer usando il codice di simulazione fluidodinamica STAR-CCM+ messo a disposizione da CD-adapco) e sulla velocità raggiungibile, stimata in 90 km/h.

Scocca e telaio riprendono quelli di Emilia 2: la prima è in fibra di carbonio e il secondo in nomex, con particolari in kevlar. Invece, ruote, cerchi, sistema sterzante, forcellone posteriore, scatola guida (il cui concept è derivato dalle tecnologie usate in Formula 1) sono stati riprogettati. Il materiale per la sospensione anteriore è stato sviluppato in collaborazione con i ricercatori del Dipartimento di Ingegneria Meccanica dell'Università di Bologna: le molle sono a balestra a sviluppo lineare in sandwich, che collega tra loro le ruote anteriori. La balestra è in legno di acacia stagionato e prepreg in fibra di carbonio.

La propulsione di Emilia 3 è a motoruota (integrazione dei motori elettrici all'interno delle ruote posteriori): una soluzione che permette di eliminare la trasmissione, aumentando il rendimento del veicolo e riducendone il peso. La motoruota è composta da un motore sincrono a magneti permanenti a rotore esterno che, grazie agli avanzati algoritmi di calcolo, all'utilizzo di software di simulazione e agli elementi utilizzati nella progettazione, raggiunge il 97% di efficienza. I 6 m² di celle fotovoltaiche al

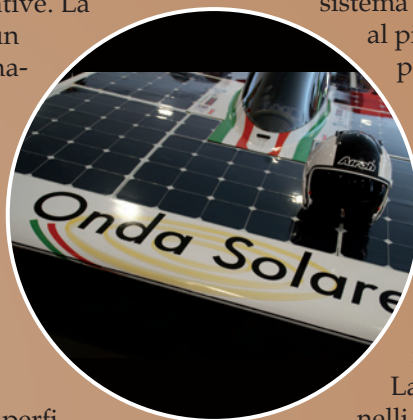
silicio monocristallino forniscono poco più di 1,3 kW grazie all'eccellente rendimento, superiore al 22%. La ricerca che ha portato alla creazione del pannello solare di Emilia 3 è stata incentrata sul massimo rendimento delle celle, sull'affidabilità e sull'ottimizzazione dei pesi in funzione della massima leggerezza della vettura. La laminazione del pannello fotovoltaico è stata curata da Solbian Energie Alternative. La laminazione è un processo estremamente delicato poiché le celle fotovoltaiche, costituite da silicio e spesse poco più di un decimo di millimetro, devono essere incapsulate per adattarsi alla superficie del veicolo solare senza danneggiarsi.

La gestione dell'elettricità ricevuta dai pannelli fotovoltaici, la ricarica degli accumulatori, nonché il controllo del motore e il recupero di energia elettrica da esso in fase di decelerazione, sono affidati a un sistema a microprocessore a 32 bit, adatto per sistemi "mission critical". Il firmware in ridondanza permette

di gestire al meglio, attraverso algoritmi di controllo integrati, l'uso delle fonti energetiche tanto in automatico quanto in semiautomatico (in modo che il pilota possa intervenire variando alcuni parametri di cruise control) e in manuale. Questa modalità comporta la gestione della vettura attraverso i comandi impartiti via radio dal team che si trova a bordo del mezzo di appoggio; il sistema di controllo darà

al pilota informazioni per agire sui rapporti del cambio in funzione della coppia necessaria a mantenere il mezzo alla velocità ottimale in base al percorso e all'energia disponibile.

La gestione dei pannelli fotovoltaici avviene con la tecnica MPPT, che permette di ottenere la massima potenza dai pannelli fotovoltaici. Il gruppo batteria di Emilia 3 è agli ioni di litio, pesa 21 kg e a piena carica può garantire un'autonomia di oltre 300 km in caso di scarso irraggiamento solare. La batteria è formata da 442 celle, gestite da un Battery Management System che tiene sotto costante controllo tensioni e temperature delle celle che costituiscono il pacco batteria, al fine di garantirne una perfetta equalizzazione.



BODYCOM: IL WIRELESS CHE AVANZA

All'orizzonte fa capolino una nuova tecnologia per la trasmissione dati senza fili a corto raggio: conosciamola.



di MATTEO DESTRO

Noi tutti siamo abituati alle tecnologie di trasmissione dati che sfruttano le canoniche frequenze di 434 e 868 MHz utilizzate per la realizzazione delle più svariate applicazioni, ad esempio per gli apri cancello, antifurto, sistemi di acquisizione dati remoti, ecc. In ogni caso da più parti si lavora per implementare nuove soluzioni che operano con protocolli innovativi o frequenze diverse, pur restando nei limiti normativi.

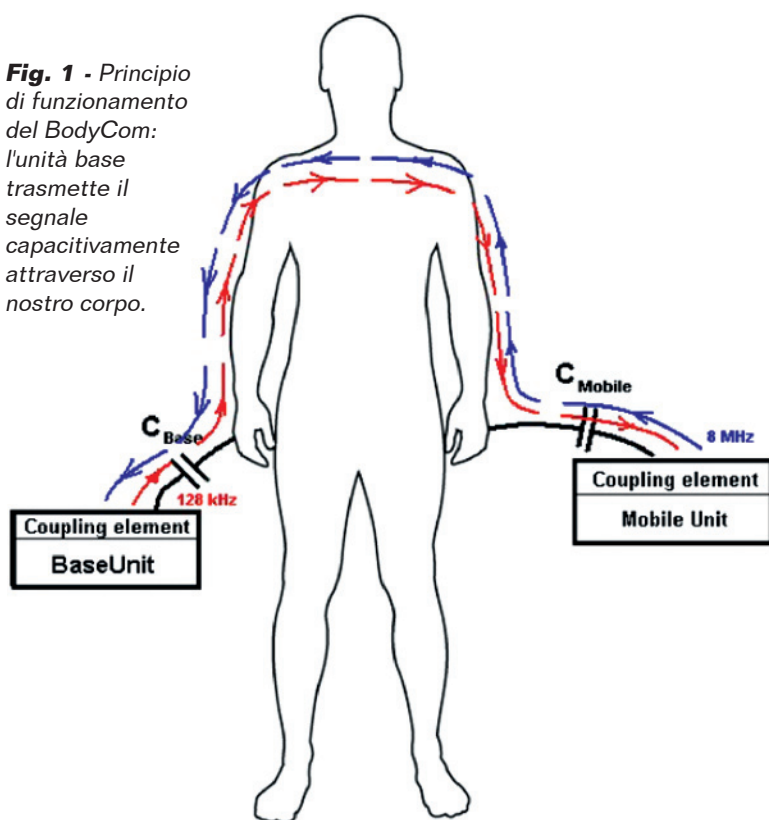
Microchip ha messo a punto una nuova tecnologia per la trasmissione dati senza fili su piccole distanze che sfrutta il corpo umano come mezzo trasmissivo: non c'è da preoccuparsi, perché è tutto nella norma e privo di effetti nocivi per la nostra salute. In pratica si sfrutta la capacità del corpo umano di trasmettere piccoli segnali per consentire la comunicazione tra due dispositivi elettronici in modo semplice e sicuro. Questa nuova tecnologia si

chiama BodyCom ed è nata con lo scopo principale di realizzare sistemi di accesso sicuro: si pensi ad esempio alla possibilità di aprire la porta di casa sfiorando semplicemente un pulsante capacitivo e tenendo in tasca il trasmettitore con i codici di sblocco. BodyCom può essere anche utilizzata per realizzare sistemi antifurto, sistemi di riconoscimento operatore a bordo macchina o altro ancora.

QUAL È LA NOVITÀ?

I sistemi di accesso remoto tradizionali usano una trasmissione RF in un canale che si trova tipicamente sulle frequenze di 434 oppure 868 MHz; di solito si basano sulla trasmissione manuale del comando da parte dell'utente, ad esempio mediante un telecomando o altro, e prevedono un livello di protezione aggiuntivo per prevenire lo "sniffing" dei dati trasmessi durante la comunicazione: lo sniffing

Fig. 1 - Principio di funzionamento del BodyCom: l'unità base trasmette il segnale capacitivamente attraverso il nostro corpo.



è l'intercettazione e lettura dei dati per carpire il codice in essi contenuto. Pensate ad esempio ai codici rolling-code per la realizzazione degli aprì cancello oppure per l'attivazione/disattivazione di un antifurto: sono una delle tecniche per prevenire la replica dei comandi, in quanto anche se qualcuno intercetta la trasmissione, non sa che farsene perché non è ripetibile. Per aumentare ulteriormente il livello di sicurezza di questi sistemi di controllo e attivazione è nata la tecnologia BodyCom, nella quale i segnali codificati non viaggiano nell'etere ma attraverso il corpo umano, sotto forma di debolissima corrente elettrica: i dati vengono accoppiati capacitivamente tra la base e l'unità mobile e trasmessi utilizzando una modulazione ASK (Amplitude Shift Key) a bassa frequenza ogniqualvolta l'utente e la base entrano in contatto. Quando

l'utente tocca il sensore capacitivo, il sistema inizia uno scambio di dati tra base e unità mobile che permane fino a quando dura l'accoppiamento capacitivo. In altre parole, la persona che deve usare il comando porta con sé il trasmettitore il cui segnale radio, tramite il corpo, raggiunge il ricevitore. Data l'ottima permittività del corpo umano alle basse frequenze, la sezione di trasmissione può essere realizzata con un semplice sistema operante tra i 60 kHz e i 30 MHz.

La progettazione di un sistema che sfrutta la tecnologia BodyCom deve soddisfare i seguenti requisiti:

- basso consumo, soprattutto dell'unità mobile;
- elevata velocità di risposta;
- comunicazione dati stabile e robusta;
- limitato campo di azione che permette l'identificazione dell'unità mobile quando è

indossata da chiunque tocchi il sensore capacitivo;

- basso costo e limitata complessità.

COME FUNZIONA

Con la tecnologia BodyCom i segnali vengono trasmessi utilizzando l'accoppiamento capacitivo tra il corpo umano, unità base e unità mobile, come mostrato in Fig. 1. Come si può osservare, l'unità base e l'unità mobile utilizzano due frequenze di trasmissione diverse perché l'attenuazione del segnale che attraversa il corpo umano cambia in base alla frequenza, ovvero è maggiore alle basse frequenze. Nel BodyCom l'unità mobile trasmette i propri dati a una frequenza di 8 MHz, il che consente di limitare la potenza impiegata e quindi il consumo dell'unità stessa; invece l'unità base opera a soli 128 kHz. Lavorare a tale bassa frequenza richiede l'erogazione di un segnale più potente e perciò maggior consumo, ma per l'unità base questo non è un problema, in quanto può essere alimentata dalla rete e non presenta problemi di autonomia.

COMUNICAZIONE DA UNITÀ BASE A UNITÀ MOBILE

In un sistema BodyCom la comunicazione viene sempre iniziata dall'unità base e per poter instaurare una comunicazione il sistema si aspetta che l'utente preme o avvicini la mano al sensore capacitivo, quindi l'attività principale del microcontrollore sarà testare il pulsante capacitivo per potere iniziare una trasmissione dati. Rilevata l'attivazione del pulsante capacitivo, il sistema smetterà di testarne lo stato e inizierà una trasmissione dati alla ricerca di un'unità mobile. La sequenza di trasmissione viene applicata a un circuito LC

fatto lavorare in risonanza. Il circuito stesso è collegato a un pad di accoppiamento che trasferisce il segnale al corpo umano, usando quest'ultimo come un'estensione dell'elemento di accoppiamento e permettendo così il trasferimento del segnale trasmesso all'unità mobile; questa deve trovarsi in prossimità del corpo umano: ad esempio nella tasca dei pantaloni. Dunque, il corpo umano diventa il mezzo di propagazione del segnale da base a unità mobile; non serve che le unità siano in contatto con il corpo ma è sufficiente che siano vicine, dato che l'accoppiamento è capacitivo. L'unità mobile di solito è in attesa di ricevere una trasmissione dall'unità base. Il segnale ricevuto dall'unità mobile "sveglia" quest'ultima, la quale provvederà a decodificare ed elaborare i dati ricevuti, nonché ad inviare una risposta verso l'unità base. La Fig. 2 mostra un diagramma a blocchi dell'unità base, la quale, come si può osservare, è formata da un sistema di ricezione completo, da un sistema di trasmissione e da un'interfaccia di comunicazione. Tutta questa

elettronica viene gestita da un microcontrollore.

COMUNICAZIONE DA UNITÀ MOBILE VERSO UNITÀ BASE

Ricevuta una trasmissione valida, l'unità mobile provvederà a decodificare i dati ricevuti e se necessario inoltrerà una risposta verso l'unità base. L'unità mobile è anch'essa accoppiata capacitivamente al corpo umano; i dati da essa trasmessi passeranno attraverso il corpo umano raggiungendo il pad capacitivo dell'unità base, la quale intercetterà la comunicazione e ne decodificherà il contenuto. L'unità base dispone di un circuito completo di ricezione formato da un preamplificatore, da un mixer step-down, da un filtro e da un decodificatore/demodulatore. Il dato così elaborato raggiungerà il microcontrollore, che provvederà alla sua decodifica. Lo schema a blocchi dell'unità mobile è mostrato in Fig. 6. Come si può osservare, anche in questo caso è presente un blocco di demodulazione del segnale ricevuto e un blocco di trasmissione dati realizzato sfruttando sempre la periferica DSM del

microcontrollore e un ponte a MOSFET in configurazione half-bridge. La sezione di demodulazione è molto più semplice rispetto all'unità base, in quanto il segnale ricevuto modula una portante a 128 kHz.

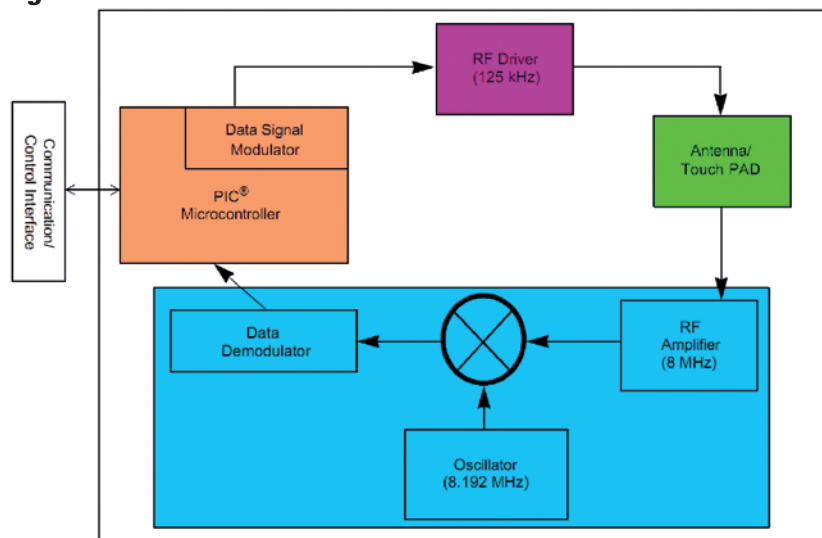
SENSORE TOUCH

Come già ampiamente detto, la comunicazione tra l'unità base e l'unità mobile inizia solo dopo l'intercettazione di un evento di pressione sul pulsante capacitivo o una rilevazione di prossimità. L'evento di prossimità viene intercettato solo quando il corpo umano, ad esempio la mano, si trova a meno di 1 cm dal sensore. Per la rilevazione dell'attivazione del pulsante capacitivo si può usare la tecnica CVD sfruttando un microcontrollore PIC. Per la trasmissione si usa un elemento capacitivo, il quale è strettamente legato al sensore capacitivo spiegato in precedenza, come evidenziato dalla Fig. 3. Quindi durante la trasmissione dei dati si deve avere la precauzione di disabilitare l'algoritmo di intercettazione della pressione del pulsante capacitivo, perché potrebbe intercettare delle false pressioni dovute alla tensione applicata all'elemento capacitivo durante la trasmissione.

Come indicato in Fig. 3, la capacità di accoppiamento tra l'elemento capacitivo e il pulsante è abbastanza alta, specialmente quando l'utente tocca i due elementi con la mano o le dita. Ciò potrebbe attenuare il segnale di trasmissione o ricezione, comunque sia questa tipologia di progetto permette delle buone performance.

Esistono altre due configurazioni da tenere in considerazione, diverse dalla precedente: la prima prevede che il pad di accoppiamento sia posizionato

Fig. 2



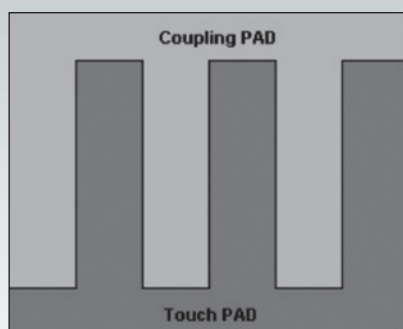


Fig. 3

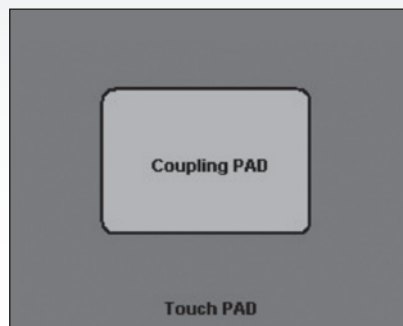


Fig. 4

al centro dell'area touch come mostrato dalla Fig. 4. La capacità di accoppiamento tra i due pad è minore rispetto alla precedente soluzione; l'utente tocca direttamente l'elemento di accoppiamento senza toccare l'area touch quindi l'attenuazione del segnale trasmesso sarà minima. La sensibilità del ricevitore è incrementata rispetto alla precedente soluzione e sono necessari anche in questo caso due pad con le relative connessioni verso il microcontrollore; inoltre questa soluzione è ottimale nel caso in cui sia necessaria la rilevazione di prossimità.

L'ultima soluzione prevede che il pad di rilevazione della pressione sia condiviso con l'elemento capacitivo di trasmissione; in questo caso sarà necessaria una sola linea di connessione verso il microcontrollore. Essendoci un solo pad, potrebbe accadere che durante la trasmissione dei dati, all'ingresso del microcontrollore siano presenti le alte tensioni necessarie per la trasmissione dei dati; ciò va evitato, ricorrendo

Preamble	Filter	Wake-up Pulse	Auto Baud Byte	SOF	Length	Data[0..n-1]	CRC Byte	EOF
6 ms ON, 0.5 ms OFF	x ms ON, y ms OFF	1 ms ON	0x55	0x7F	1 Byte	n Bytes	1 Byte	5 ms ON

Fig. 5

alle opportune protezioni. Con questa soluzione, inoltre, non è possibile implementare la rilevazione della prossimità.

STRUTTURA DEL PACCHETTO DATI

I dati che si desidera trasmettere sfruttando la tecnologia Body-Com devono essere impacchettati in maniera adeguata allo scopo di rendere la comunicazione tra i dispositivi sicura e robusta. I dati che vengono trasmessi, sia a 128 kHz che a 8 MHz, vengono ricevuti e decodificati dal front-end MCP2035 o MCP2030 di Microchip.

La Fig. 5 mostra come devono essere impacchettati i dati: per prima cosa si deve generare il preambolo necessario per attivare la circuiteria dell'integrato MCP2035 o MCP2030; il preambolo si genera portando il segnale a livello alto per 6 ms e basso per 500 µs; segue un filtro temporale configurabile via firmware per evitare dei falsi Wake-Up dell'hardware, ottimizzando i consumi della circuiteria. Per

ultimo si ha un impulso da 1 ms, detto di Wake-Up, necessario al microcontrollore per concludere la sua inizializzazione dopo il risveglio dalla fase di sleep. Tutto quello che segue a questo preambolo sono i dati da trasmettere, i quali possono essere organizzati con un protocollo a piacere dell'utente. Per quanto riguarda l'applicazione in esame (BodyCom) viene trasmesso per prima cosa un byte per l'autobaud della seriale, necessario per instaurare la comunicazione seriale; seguono un byte SOF (Start Of Frame) che identifica l'inizio del pacchetto dati, un byte che identifica il numero di byte da trasmettere, quindi tutti i dati da trasmettere seguiti da un byte di checksum. Per ultimo si deve generare un EOF (End Of Frame Sequence) formato da un livello logico alto della durata di 5 ms. I dati che devono essere trasmessi, sia dall'unità base che dall'unità mobile, vengono modulati sfruttando l'hardware del microcontrollore, ovvero il blocco

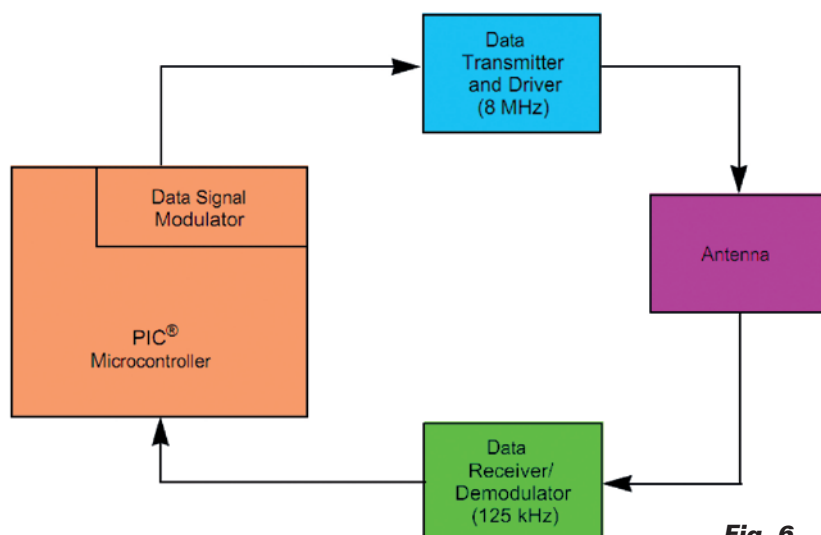


Fig. 6

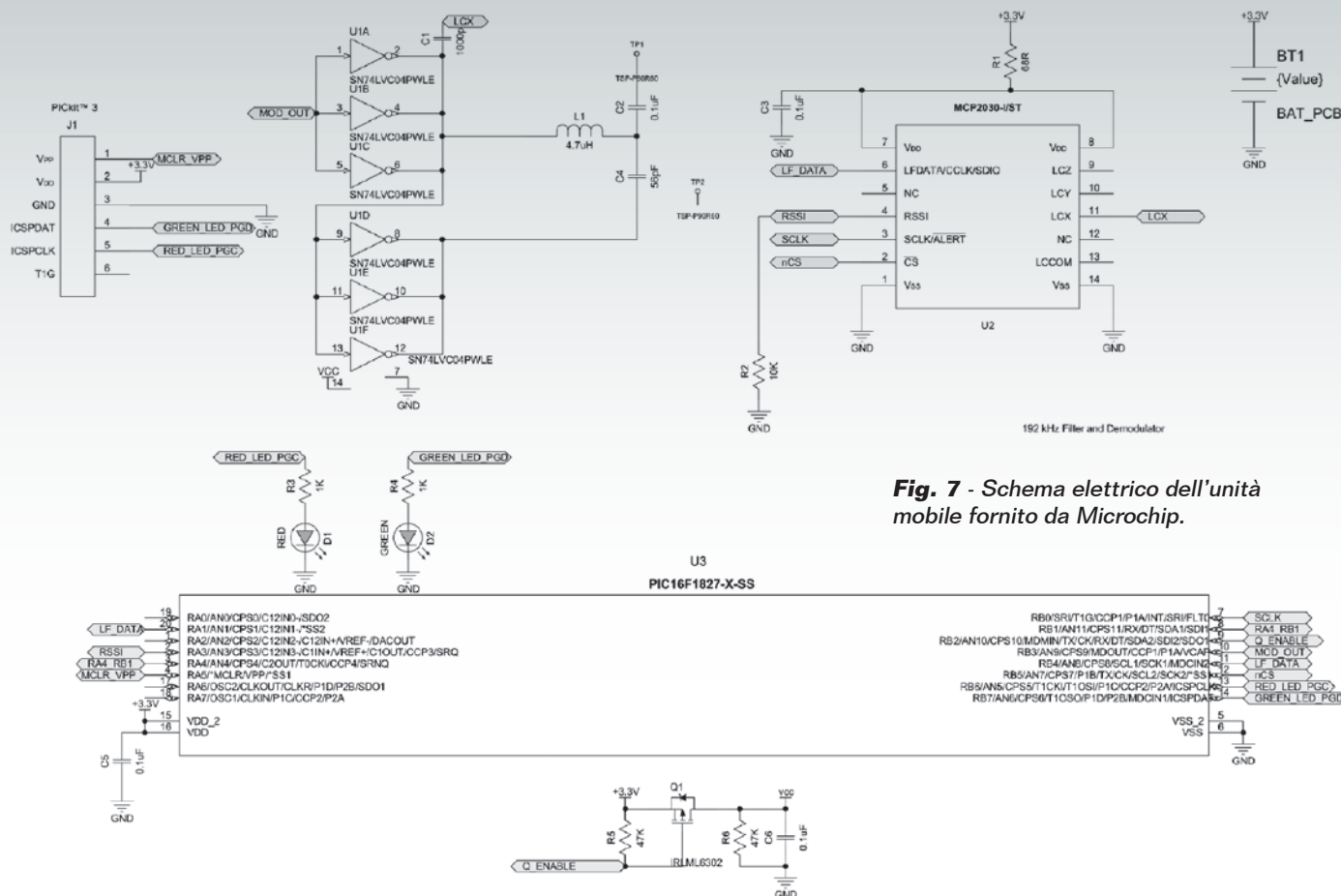


Fig. 7 - Schema elettrico dell'unità mobile fornito da Microchip.

DSM (Data Signal Modulator). Il segnale modulante viene generato sfruttando la periferica UART o la periferica SPI del microcontrollore; nel caso del BodyCom, si utilizza la periferica UART. La portante viene generata sfruttando la periferica on-chip clock del microcontrollore, sia per la trasmissione a 128 kHz sia per la trasmissione a 8 MHz. Il segnale così generato viene passato a un ponte a MOSFET in configurazione half-bridge o anche full-bridge, il quale pilota la rete LC collegata al pad di accoppiamento capacitivo dell'unità.

CONFIGURAZIONE DEL CIRCUITO RISONANTE LC

Il sistema BodyCom per la trasmissione e ricezione dei dati utilizza una rete LC collegata ad un unico elemento di accoppiamento capacitivo. La frequenza

di risonanza del circuito LC può essere calcolata utilizzando la seguente formula:

$$f_0 = \frac{1}{2 * \pi * \sqrt{L * C}}$$

In realtà il circuito serie LC comprende anche una resistenza (RLC) che può essere ricavata dalla seguente formula:

$$I_{ant} = \left(\frac{4}{\pi} \right) * \left(\frac{V_{dd}}{R_{LC}} \right)$$

Da cui ricaviamo:

$$R_{LC} = \left(\frac{4}{\pi} \right) * \left(\frac{V_{dd}}{I_{ant}} \right)$$

Il valore resistivo RLC è dato da un insieme di fattori tra cui RDRIVE (resistenza introdotta dal driver che pilota il circuito LC), RTRACE (resistenza della

pista di connessione, trascurabile), RRF (resistenza parassita dell'induttanza). Sulla base di queste indicazioni e supponendo di volere una corrente massima di 50 mA per pilotare la rete LC e una tensione di alimentazione di 5 Vcc, la resistenza RLC dovrà essere di circa 127,5 Ω (unità base). Per l'unità mobile, sia la tensione di alimentazione che la corrente disponibile per pilotare il circuito LC scendono rispettivamente a 2,2 Vcc e 25 mA e quindi la resistenza RLC ammonta a circa 112 Ω . Calcolata la resistenza RLC, si deve passare al calcolo della rispettiva induttanza sia per l'unità base che per la mobile, utilizzando la seguente formula:

$$Q = \frac{\omega_0 * L}{R_{LC}}$$

Il fattore di merito Q è importan-

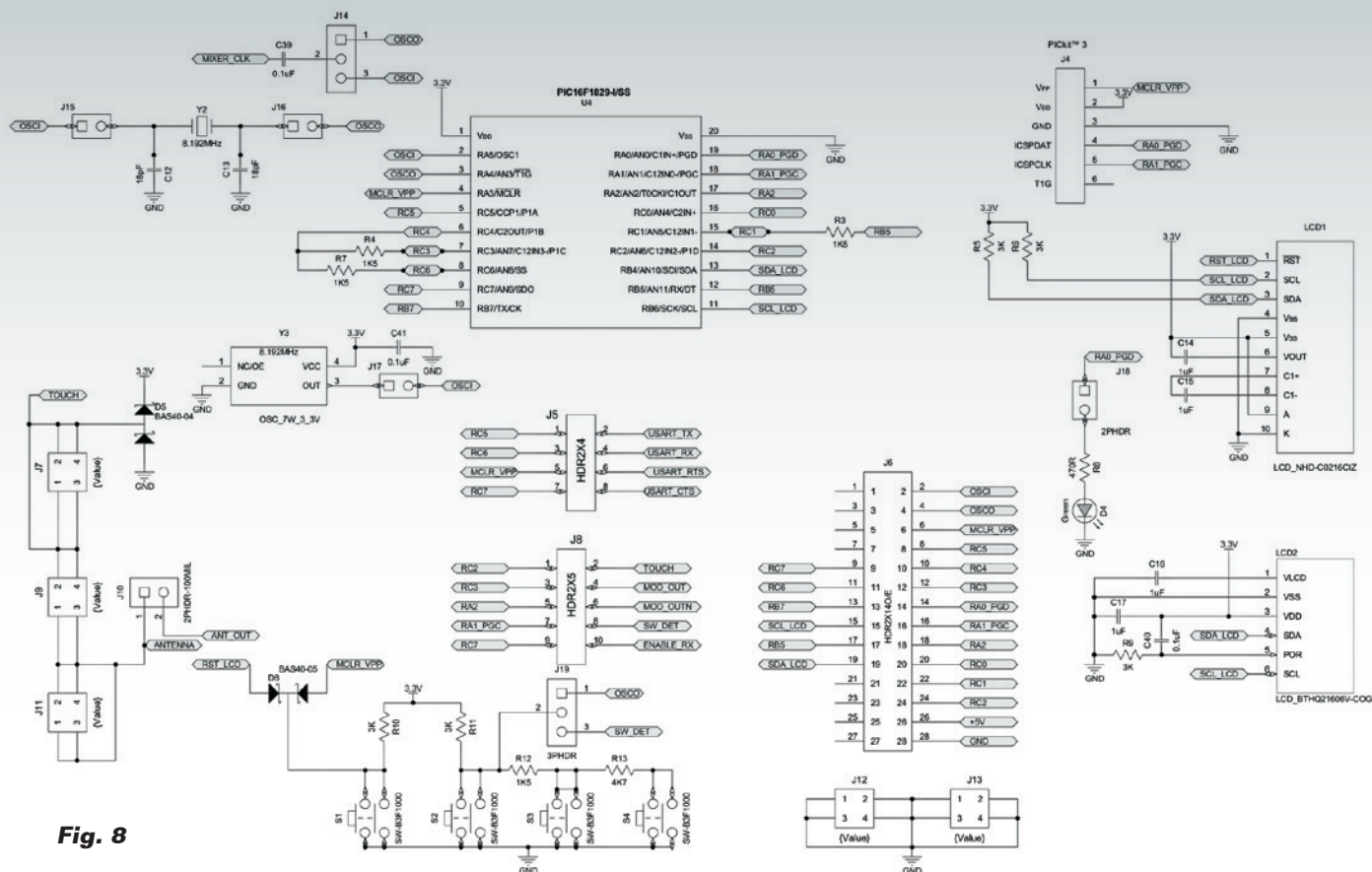


Fig. 8

te e va scelto in modo accurato perché se è troppo alto si rischia di ridurre la larghezza di banda, mentre se è eccessivamente basso causa l'accesso di frequenze estranee. Per un sistema BodyCom il giusto fattore di merito è 20, di conseguenza si ricavano i rispettivi valori per le induttanze dell'unità base e mobile (per l'unità base L deve essere minore o uguale a 3,17 mH, ad esempio 2,2 mH) mentre per l'unità mobile l'induttanza deve essere minore o uguale a 11,2 μH (ad esempio 4,7 μH). Per ultimo, si può calcolare il valore della capacità C, sfruttando la seguente relazione:

$$f_0 = \frac{1}{2 * \pi * \sqrt{L * C}}$$

Quindi, avendo un'induttanza

da 2,2 mH (unità base) il valore di C sarà circa 703 pF (valore standard più prossimo, 620 pF) mentre con un'induttanza da 4,7 μH (unità mobile) il valore di C sarà circa 80 pF (il valore standard più prossimo è 68 pF).

PROGETTARE IL SISTEMA

Il sistema BodyCom può essere utilizzato per realizzare svariate applicazioni; l'hardware e il firmware di gestione del sistema si implementano con facilità grazie al supporto fornito da Microchip. Nel progettare il sistema è consigliabile utilizzare dei microcontrollori con a bordo l'interfaccia DSM, il che facilita le operazioni di trasmissione del segnale modulato. Altro elemento fondamentale da tenere in considerazione è il front-end MCP2035 o MCP2030, necessario per la

decodifica del segnale ricevuto sia nell'unità base (sistema di ricezione più articolato per via della frequenza di trasmissione dell'unità mobile) che dall'unità mobile. In queste pagine trovate gli schemi elettrici sia dell'unità mobile (Fig. 7) che dell'unità base (Fig. 8, 9 e 10) forniti da Microchip e implementati nella DemoBoard BodyCom che la stessa Microchip ha realizzato.

Per l'unità base, lo schema elettrico è suddiviso in tre sezioni distinte:

- sezione microcontrollore e display LCD (Fig. 8); la scheda BodyCom è provvista di un display LCD per l'interazione con l'utente e di quattro pulsanti per la gestione delle varie sezioni del menu visualizzato sul display (per la comunicazione verso il

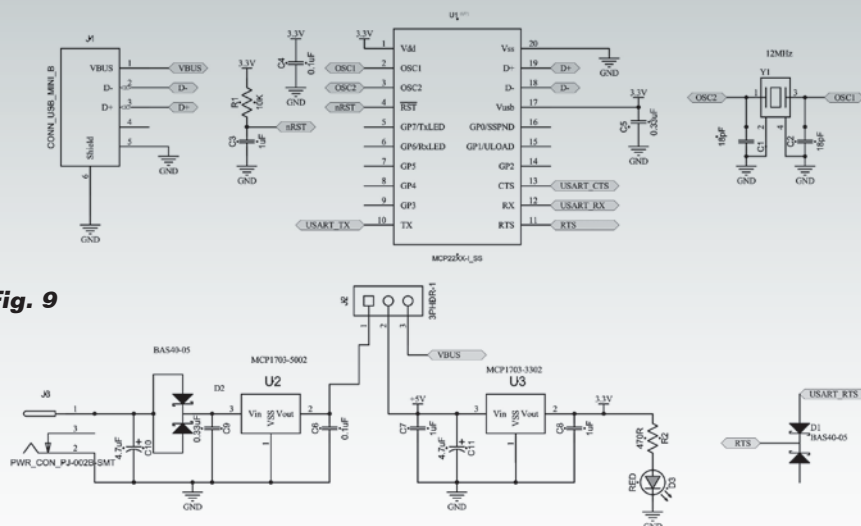
microcontrollore il display utilizza l'interfaccia I²C;

- sezione di alimentazione e comunicazione verso il PC (Fig. 9), dove per la comunicazione verso il PC si sfrutta l'integrato MCP22xx che è l'equivalente del converter TTL/USB dell'FTDI (FT232);
- sezione analogica per trasmissione e ricezione dati (Fig. 10).

Quest'ultima è a sua volta distinta in tre parti, che sono una sezione di trasmissione dati tramite ponte H completo, un circuito di ricezione del segnale con relativa amplificazione più mixer, ed una sezione di filtro e decodifica del segnale.

Assieme alla demoboard vengono fornite due unità mobili programmate con due diversi codici ID e un cavo di collegamento

Fig. 9



USB per connettere la scheda al PC. Sfruttando il software BodyCom GUI, scaricabile dal sito Microchip, si possono eseguire le seguenti operazioni: acquisire nuove unità mobili, cancellare dalla memoria della scheda base le unità mobili acquisite, eseguire il Ping e il Loopback verso una unità mobile tra quelle acquisite,

sfruttare il sensore capacitivo sia in modalità Touch che in modalità prossimità (quindi al tocco del pad capacitivo con in tasca l'unità mobile verrà visualizzato il rispettivo codice ID).

Si può inoltre effettuare il boot del microcontrollore della scheda base per caricare eventuali aggiornamenti.

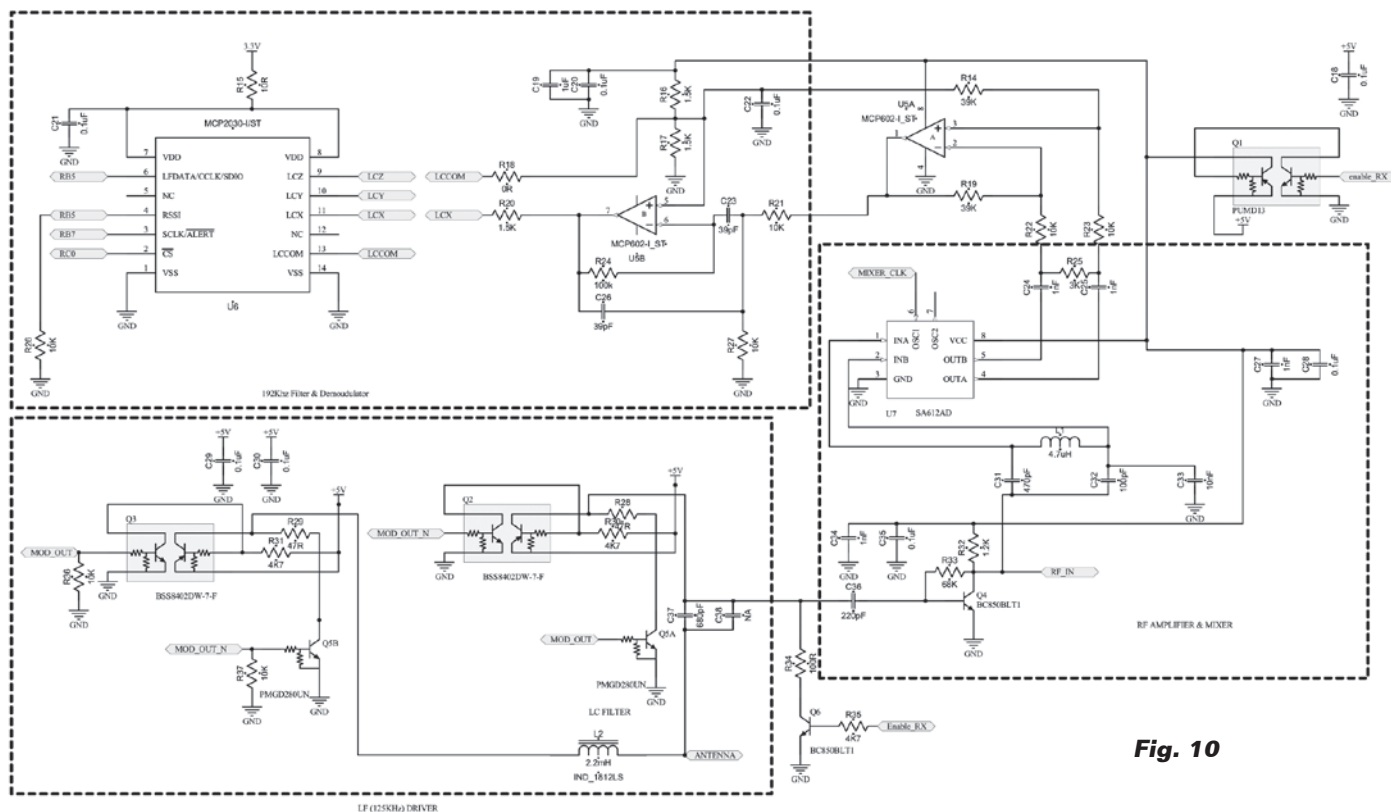


Fig. 10

Continuiamo il nostro viaggio alla scoperta di Flowcode, l'innovativo sistema di sviluppo grafico per microcontrollori proposto da Matrix Multimedia. In questa puntata ci occupiamo dell'implementazione degli interrupt in Flowcode e iniziamo ad analizzare i componenti. Terza puntata.

corso di programmazione in **FLOWCODE** di Francesco Ficili e Daniele Defilippi

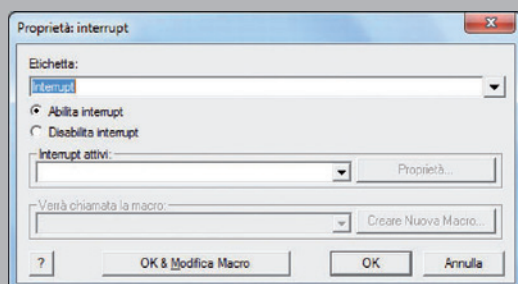
Nelle precedenti puntate abbiamo analizzato in dettaglio l'ambiente di sviluppo, descrivendone la struttura ed analizzando gli elementi di base come le strutture di controllo e le strutture dati. In questa puntata completeremo l'analisi delle strutture di controllo, descrivendo la gestione degli interrupt, e continueremo l'analisi dei componenti (che sono uno dei principali valori aggiunti di Flowcode) descrivendo il componente ADC ed il componente LCD. Successivamente passeremo alla descrizione della gestione delle memorie non volatili, analizzando il componente EEPROM. Sfruttando le nuove conoscenze acquisite realizzeremo un progetto prati-

co più evoluto rispetto ai precedenti: un termometro con display LCD. Per i progetti di questa puntata cambieremo microcontrollore, passando dal PIC18F4550 al PIC18F4580. Vedremo così quanto sia semplice, gestendo i progetti con Flowcode, cambiare target senza risentire di problematiche di configurazione e faremo qualche esperimento con un PIC diverso. Per agevolare i lettori nella sperimentazione abbiamo generato un file di configurazione base anche per il PIC18F4580, così come avevamo fatto in precedenza.

Gli interrupt

Come tutti gli ambienti di sviluppo per

Fig. 1
Proprietà
del blocco
interrupt.



microcontrollori, anche Flowcode prevede la gestione degli interrupt. In un sistema embedded un interrupt (o interruzione) è un segnale asincrono rispetto all'esecuzione del programma principale, proveniente da una specifica periferica, che segnala una situazione particolare che richiede una gestione differente rispetto al resto del programma. Un interrupt può provenire da una porta di comunicazione, da una risorsa interna, da un timer, eccetera. Normalmente, quando viene generato un interrupt, la gestione del programma si arresta, la posizione corrente del program counter ed il contesto di esecuzione vengono salvati nello stack e il programma salta a una locazione specifica, che è funzione dell'interrupt stesso e dell'architettura specifica del microcontrollore. A questo punto, a seconda dei casi, viene eseguita una funzione speciale (in genere denominata ISR, Interrupt Service Routine) che "serve" l'interrupt e che, una volta terminata, ritorna il controllo al programma principale, che ricomincia dal punto in cui era stato interrotto (recuperando tutte le informazioni necessarie dallo stack, all'interno del quale era stata salvata la posizione corrente ed il contesto).

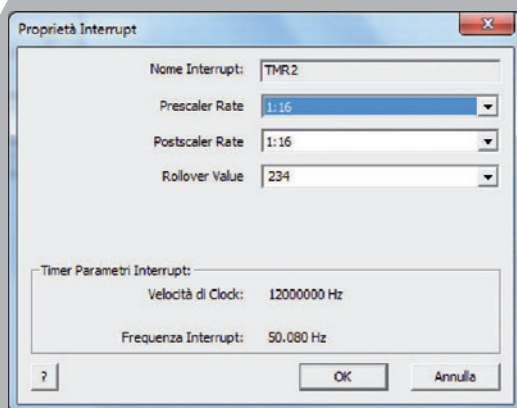
In Flowcode esiste un blocco speciale, presente all'interno della barra delle icone e denominato appunto "blocco interrupt", che serve ad implementare la gestione degli interrupt all'interno di un programma Flowcode. Se inseriamo questo blocco all'interno di uno schema Flowcode e clicchiamo due volte con il tasto sinistro, potremo accedere alla finestra proprietà del blocco, come riportato in Fig. 1. Da questa finestra è possibile selezionare la sorgente dell'interrupt che ci interessa, accedendo al menu a tendina "Interrupt Attivi". Analizzando questo menu, vedremo che per il PIC18F4580, sono disponibili

diversi interrupt sui timer, sugli ingressi della porta B, sulla ricezione della porta UART e altro ancora. Una volta selezionata la sorgente dell'interrupt, bisognerà indicare anche la relativa ISR. Flowcode gestisce questo aspetto, fornendo la possibilità di chiamare un'opportuna macro creata in precedenza. Tutto ciò non è molto diverso da quello che accade con altri sistemi di sviluppo, basati su linguaggi classici come, ad esempio, il C: in quel caso la ISR è generalmente una comune funzione scritta in linguaggio C, che ha però la particolarità di essere chiamata quando viene generato il relativo interrupt.

Il concetto di interrupt è abbastanza delicato in ambito di sviluppo embedded, in quanto gli interrupt sono largamente utilizzati nella gestione dell'hardware e sono di fondamentale importanza man mano che i sistemi sviluppati si fanno più complessi. Cerchiamo di fissare questo concetto con un esempio, riproponendo il progetto pratico visto nella prima puntata con una gestione ad interrupt.

Nella prima puntata avevamo fatto lampeggiare un LED alla frequenza di 1 Hz, sfruttando i ritardi software. Proviamo adesso a fare la stessa cosa, ma anziché generare il ritardo con dei cicli software vuoti, proviamo a generare la temporizzazione con una risorsa hardware interna al microcontrollore. Per farlo ci serviamo del Timer 2, che può generare un interrupt quando il suo valore corrisponde a quello di un registro di confronto. Procediamo quindi creando un nuovo progetto Flowcode, partendo dal template ConfigurazioneBase2, fornito insieme al materiale relativo a questa puntata, e cambiamo il nome del progetto in Interrupt.fcf. Per prima cosa creiamo la macro che useremo come ISR, nominandola LED. Eseguiamo i passi necessari come abbiamo visto nella puntata precedente e generiamo anche una variabile che chiameremo LedStatus (booleana, valore iniziale false) ed un'altra che chiameremo Counter (byte, valore iniziale 0). A questo punto passiamo sullo schema a blocchi principale, inseriamo un blocco interrupt, e a seguire un blocco ciclo. Il ciclo ci serve solo per evitare che il pro-

Fig. 2
Proprietà interrupt timer 2.



gramma termini e riparta dal principio, lo possiamo quindi considerare come la nostra "applicazione principale", anche se non fa nulla. A questo punto configuriamo il blocco interrupt. Nel menu a tendina, alla voce "interrupt attivi" selezioniamo TMR2 e clicchiamo su proprietà. Otterremo la finestra di **Fig. 2**, che ci permette di configurare l'interrupt sul Timer 2. Funzionando a 12 MHz, anche intervenendo sui vari pre e post-scaler non c'è modo di generare un interrupt a 1Hz. Configuriamo quindi l'interrupt con i valori riportati in figura, per ottenere un interrupt a circa 50 Hz. Partendo da questa base tempi ci basterà semplicemente utilizzare un contatore di servizio per generare la frequenza di 1Hz che ci serve per pilotare opportunamente il nostro LED. Completiamo la configurazione tornando sulla finestra precedente e selezionando la macro LED alla voce "verrà chiamata la macro". Questo fa sì che ad ogni interruzione generata dal timer 2 venga chiamata la nostra macro. A questo punto non ci rimane altro da fare che completare la macro precedentemente creata. Per vedere come abbiamo fatto ci serviamo della **Fig. 3**. Come si può vedere dallo schema a blocchi, ad ogni chiamata della macro viene incrementato il contatore "Counter". Se il valore è inferiore al valore della costante timeout (che abbiamo impostato pari a 50) non succede nulla, altrimenti si passa alla gestione effettiva del LED. In questo ramo del flowchart si controlla il valore della variabile LedStatus, invertendolo, di volta in volta, rispetto al suo stato attuale e pilotando di conseguenza il LED. A questo punto provate a "flashare" il microcontrollore con il flowchart fornito come esempio (notate che il micro è

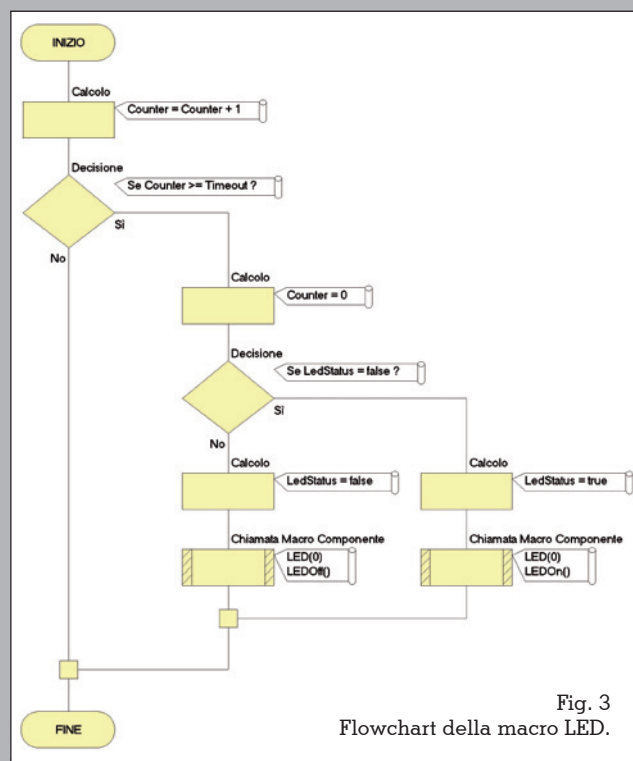


Fig. 3
Flowchart della macro LED.

cambiato e che ora il pin controllato è D0) e vedrete lo stesso effetto ottenuto nel caso dell'oscillazione generata con i ritardi software. Notate che questo esempio non è puramente didattico. Infatti, se volessimo aggiungere funzionalità a questo programma, ci basterebbe inserire i relativi blocchi sul ciclo principale che attualmente è vuoto, "dimenticandoci" della gestione ad interrupt, che non viene disturbata dall'esecuzione del main (in quanto asincrona rispetto a quest'ultimo), mentre la stessa cosa non si potrebbe dire per la versione fatta con i ritardi software.

Il componente ADC

Passiamo ora all'analisi di uno dei componenti che utilizzeremo nel prossimo progetto pratico: il componente ADC. Questo componente permette di configurare e gestire in maniera semplice un canale analogico di ingresso, sfruttando il convertitore integrato nel microcontrollore. Il componente ADC può essere inserito selezionandolo dalla barra dei componenti, sotto la voce "ingressi". Una volta inserito, nel pannello comparirà una manopola del tutto simile a quella che si trova nei pannelli frontali della strumentazione da laboratorio. Come abbiamo già visto nella

Fig. 4
Finestra
connessioni del
componente
ADC.

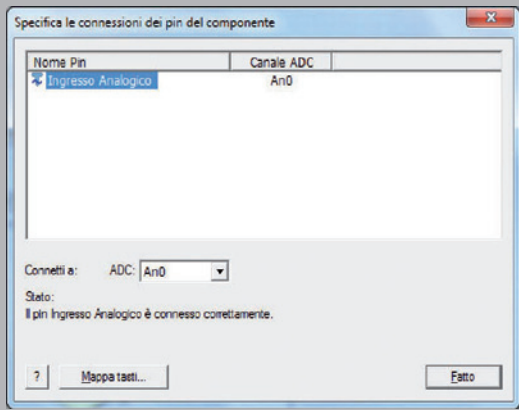
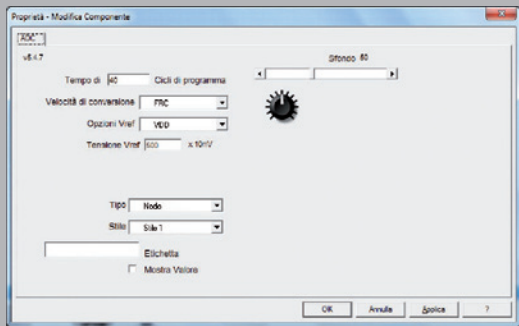


Fig. 5
Proprietà
extra del
componente
ADC.



puntata precedente per il componente LED, anche il componente ADC è dotato di “connessioni” e di “proprietà extra”. È possibile accedere a questi due menu cliccando con il tasto destro sull'icona del componente presente sul pannello. Cliccando sulla voce “connessioni” comparirà la finestra di Fig. 4, che permette di impostare il pin del microcontrollore da utilizzare come ingresso analogico (nell'esempio An0). Per accedere alle “proprietà extra”, basta selezionare la relativa voce nel menu che compare sempre cliccando con il tasto destro sull'icona del pannello. La finestra proprietà extra del componente ADC è riportata in Fig. 5. Com'è possibile vedere - impostazioni dell'icona a parte - sono disponibili alcu-

ne impostazioni base del componente, come la tensione di riferimento (Vref), la velocità di conversione e altro ancora. Per quanto riguarda invece le macro che il componente ADC può sfruttare, in **Tabella 1** sono elencate le principali macro disponibili. Il componente dispone anche di macro più specifiche, ma al momento non le prenderemo in considerazione in quanto non strettamente necessarie.

Il componente LCD

Passiamo ora all'analisi del componente LCD, che ci permetterà di aggiungere un output visuale ai nostri progetti. Il componente LCD permette di gestire un display a cristalli liquidi con controller Hitachi HD44780 (noto anche come Hitachi compatibile); questa interfaccia costituisce lo standard de facto per gli LCD alfanumerici monocromatici con interfaccia parallela. In particolare, il componente sfrutta la configurazione parallela a 4-bit, in modo da poter impiegare una singola porta per il controllo del dispositivo (oltre ai 4 bit di dato vengono infatti utilizzati 2 segnali di controllo). Il componente LCD può essere inserito dalla barra dei componenti, selezionando, sotto la voce “uscite”, il componente “LCD”. Una volta eseguita questa operazione, all'interno del pannello verrà posizionata l'immagine di un display alfanumerico 16x2. Come per gli altri componenti visti in precedenza, anche per il componente LCD sono disponibili le solite proprietà “connessioni” e “proprietà extra”. Se clicchiamo sulla voce connessioni, comparirà la finestra di Fig. 6, che ci permette di configurare le connessioni del display. Da questa finestra è

Tabella 1 - Principali macro disponibili per il componente ADC.

Nome Macro	Descrizione
LeggiComeByte	Il valore del canale selezionato è acquisito e il risultato è fornito con formato numerico e 8-bit di risoluzione.
LeggiComeIntero	Il valore del canale selezionato è acquisito e il risultato è fornito con formato numerico e 10-bit o 12-bit di risoluzione (a seconda del convertitore).
LeggiComeTensione	Il valore del canale selezionato è acquisito e il risultato è fornito con formato in tensione (il valore deve essere immagazzinato in una variabile di tipo float).
LeggiComeStringa	il valore del canale selezionato è acquisito e il risultato è fornito come stringa di caratteri (il valore deve essere immagazzinato in un array di caratteri).

possibile impostare le 6 linee necessarie all'interfacciamento del componente con il microcontrollore.

Se apriamo invece la finestra "proprietà extra", rappresentata in Fig. 7, sarà possibile configurare la dimensione del display (16x1, 16x2, 16x4, 20x4...), più altre caratteristiche che comunque sono utili più ai fini della simulazione che altro.

Diamo ora uno sguardo alle macro disponibili per questo componente, per capire come poterlo gestire efficacemente nei no-

stri progetti. In **Tabella 2** sono indicate le macro disponibili per il componente LCD correlative della relativa descrizione.

Le memorie non volatili

Come sappiamo, diversi PIC dispongono al loro interno di un certo quantitativo di memoria non volatile di tipo EEPROM (Electrical Erasable and Programmable Read Only Memory); questa può essere utilizzata per l'immagazzinamento di dati che non devono essere persi dopo uno spegnimento del sistema. Una on-board EEPROM può essere un grande vantaggio nel caso in cui si debba sviluppare un sistema embedded che ha esigenze di memorizzazione permanente di alcuni dati, perché in questo caso non è necessario ricorrere all'uso di una memoria esterna I2C o SPI. Il PIC18F4580 dispone di 256 byte di on-board EEPROM, garantiti per circa un milione di cicli di scrittura e con una ritenzione di più di 40 anni.

Il componente EEPROM

Flowcode dispone di un componente apposito per la gestione della EEPROM interna. Tale componente può essere inserito dalla barra dei componenti, selezionando sotto la voce "varie" il componente "EEPROM". Una volta inserito il componente vedrete comparire nel pannello un array bidimensionale che mostra la locazione di memoria e il suo contenuto. Per il componente EEPROM non sono presenti le connessioni, in quanto questo componente non è collegato esternamente in quanto il bus per la gestione della memoria è interno al chip. È disponibile invece l'opzione

Fig. 6
Connessioni del
componente
LCD.

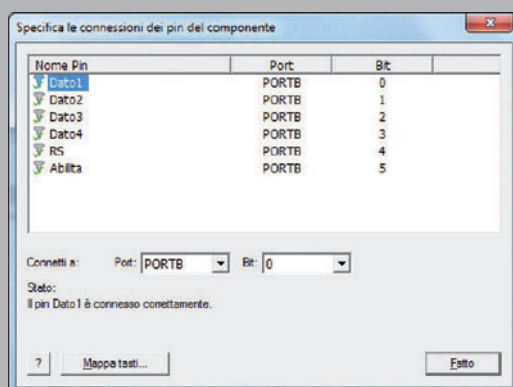


Fig. 7
Proprietà
extra del
componente
LCD.

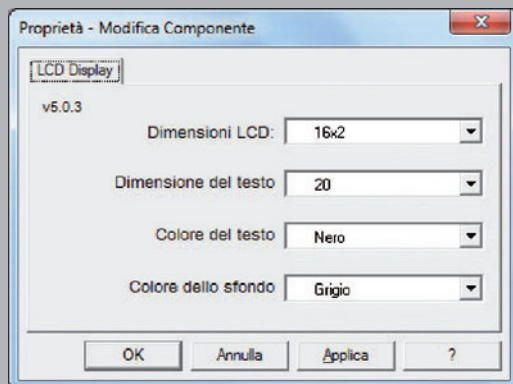


Tabella 2 - Principali macro disponibili per il componente LCD.

Nome Macro	Descrizione
Inizio	Inizializza il display.
Trasparente	Cancella tutte le righe del display.
VisualizzaASCII	Visualizza il carattere ASCII corrispondente al byte passato come argomento.
Comando	Invia il comando passato come argomento.
Cursore	Posiziona il cursore alla posizione indicata dai due argomenti x e y (coordinate del cursore).
VisualizzaNumero	Visualizza il numero passato come argomento già convertito in ASCII.
VisualizzaStringa	Visualizza la stringa passata come argomento
CancellaLinea	Cancella la linea passata come argomento.
Scrivi_RAM	Scrive nella RAM del controller i dati passati come argomento.

Fig. 8
Proprietà
extra del
componente
EEPROM.

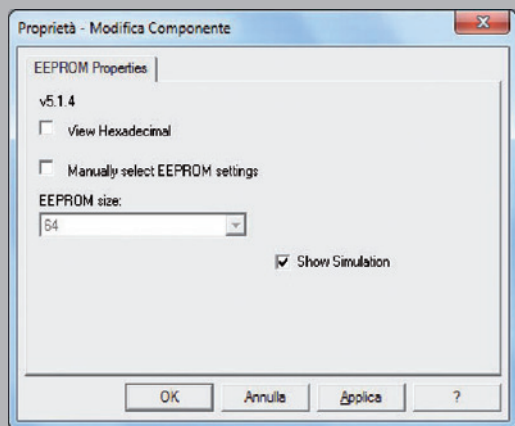


Tabella 3

Nome Macro	Descrizione
Read	Legge la locazione di memoria specificata con il parametro address e ritorna il valore letto nella variabile di ritorno.
Write	Scrive la variabile specificata con il parametro data all'interno della locazione specificata con il parametro address.

“proprietà extra”, che può essere selezionata sempre premendo il tasto destro sull'icona del componente presente sul pannello. La finestra delle proprietà è visibile in Fig. 8.

Sono disponibili più che altro proprietà relative alla simulazione ed è possibile settare a mano la dimensione della EEPROM, altrimenti l'ambiente imposta quella massima relativa al chip utilizzato.

In **Tabella 3** sono riportate le macro disponibili per il componente che sono tutte molto semplici.

L'unica accortezza da utilizzare è quella di lasciare un intervallo di almeno 200 microsecondi tra una scrittura e la successiva, per garantire il tempo necessario al completamento dell'operazione.

Per testare la EEPROM è stato realizzato un flowchart di prova che scrive in tutte le 256 locazioni di memoria un numero crescente da 0 a 256. Il sorgente Flowcode EepromTest.fcf è reperibile all'interno del materiale di supporto.

Progetto Pratico: Termometro LCD

Passiamo ora alla descrizione del progetto pratico relativo a questa puntata, nello sviluppo del quale metteremo in pratica le nuove conoscenze acquisite. Ciò che ci proponiamo di realizzare è un termometro digitale che impiega un LM335 ed un display LCD Hitachi compatibile. Le specifiche del nostro progetto sono:

- Lettura del termometro digitale sul canale analogico AN0,

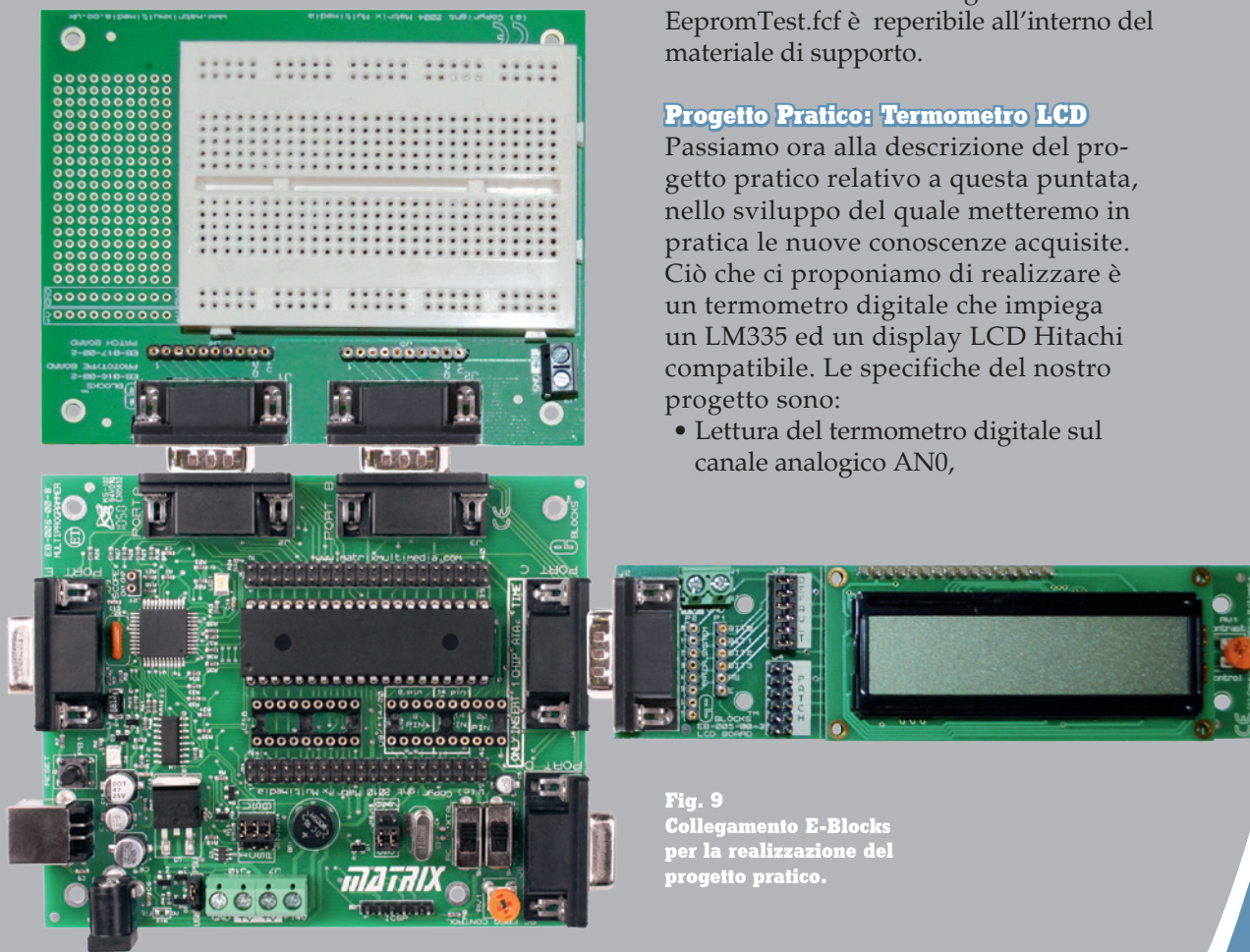


Fig. 9
Collegamento E-Blocks
per la realizzazione del
progetto pratico.

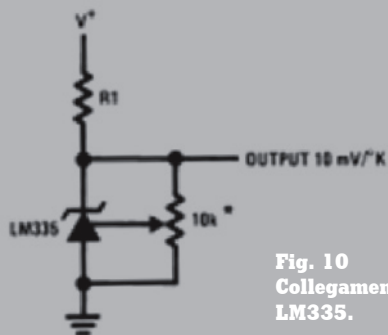


Fig. 10
Collegamento
LM335.

- Conversione del dato di temperatura in gradi Celsius,
- Visualizzazione continua della temperatura sul display.

Per la realizzazione del progetto utilizzeremo i seguenti componenti:

- EB006 "USB PICmicro multiprogrammer board" (PIC18F4580 e Quarzo 12MHz),
- EB005 "LCD board",
- EB016 "Prototype board".

Useremo la EB016 per la realizzazione del circuito di acquisizione della temperatura tramite l'LM335. Naturalmente Matrix dispone di alcune schede con sensori di temperatura, più o meno evolute, ma volemmo mostrare ai lettori come sia possibile interfacciare qualsiasi tipo di sensore usando la scheda di prototipazione.

Collegiamo le schede come segue:

- EB016 collegata alla porta A ed alla porta B della EB006,
- EB005 collegata alla porta C della EB006.

I collegamenti sono illustrati in **Fig. 9**.

Prima di andare avanti, bisogna realizzare il circuito di acquisizione della temperatura sulla scheda di prototipazione. Per farlo abbiamo fatto riferimento al datasheet dell'LM335, che riporta alcuni esempi di circuiti. La nostra scelta è caduta sul sensore con circuito di calibrazione, il cui schema è riportato in **Fig. 10**.

Questo circuito ci consente di avere una misura più precisa della temperatura, in quanto si utilizza un potenziometro per calibrare il sensore. Una volta completato il progetto, potremo eseguire la calibrazione semplicemente agendo sul potenziometro ed utilizzando un termometro calibrato per

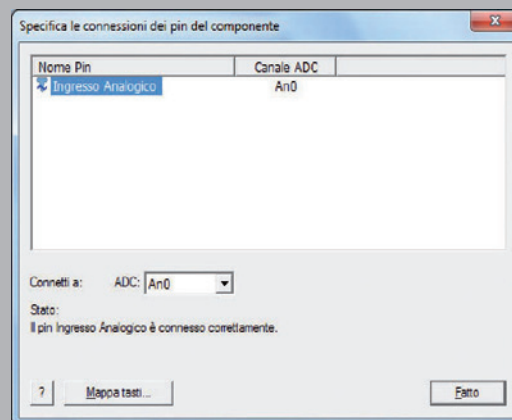


Fig. 11
Connessioni
del
componente
ADC.

ottenere la temperatura di riferimento. A questo punto procediamo alla realizzazione del circuito sulla breadboard e colleghiamo l'uscita del sensore all'ingresso analogico AN0 del microcontrollore, che troviamo disponibile sulla strip montata a ridosso del connettore J1 sulla EB016. Ora che i collegamenti di segnale sono ultimati, non ci resta che distribuire anche le alimentazioni ai vari E-Block. Per farlo preleviamo riferimento e tensione di alimentazione dalla scheda madre (presenti sulla morsettiera J10) e li portiamo alle morsettiere a vite J1 della EB005 e J3 della EB016.

Ora che l'hardware è pronto, possiamo procedere allo sviluppo delle componenti software. Per prima cosa preoccupiamoci di inserire e collegare opportunamente tutti i componenti che utilizzeremo nel progetto. Viste le specifiche, ci occorre sicuramente un componente ADC ed un componente LCD. Non abbiamo da memorizzare informazioni permanenti, quindi in questo progetto non avremo bisogno del componente EEPROM. Procediamo trascinando i componenti sul pannello. A questo punto effettuiamo le connessioni. Il componente ADC dovrà leggere il canale analogico 0 (AN0), quindi effettuiamo la connessione con quel canale, come illustrato in **Fig. 11**.

Completate le connessioni, occupiamoci delle proprietà extra del convertitore analogico-digitale. Per questo progetto impostiamo:

- Tempo di conversione: 50 cicli
- Velocità di conversione: Fosc/16
- Opzioni Vref: Vdd

Per quanto riguarda la voce "Tensione

Fig. 12
Connessioni
display LCD.

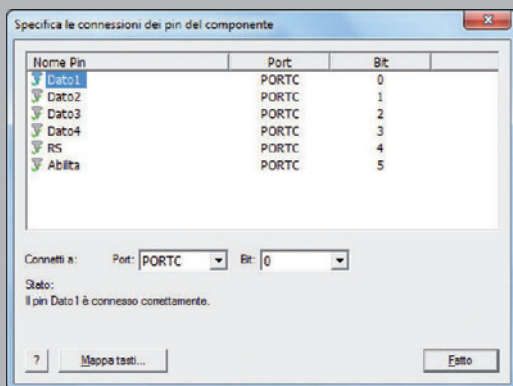
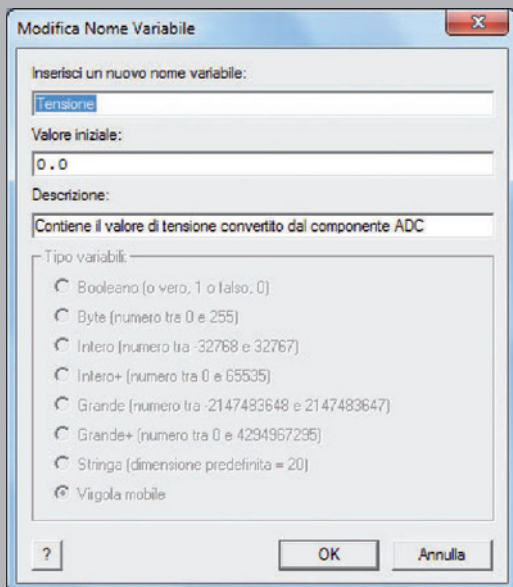


Fig. 13
Creazione
della
variabile
"Tensione".



Vref" possiamo impostare 4,5V se alimentiamo il sistema tramite la porta USB (è la tensione tipica) o 5V se alimentiamo dall'esterno con un regolatore stabilizzato. In ogni caso potremo aggiustare in seguito la misura, usando il circuito di calibrazione che ci siamo preparati in precedenza. A questo punto passiamo alle connessioni del display LCD, collegato sulla porta C, come illustrato in Fig. 12. Ricordiamo che l'interfaccia del display è parallela a 4-bit e utilizza due ulteriori segnali: RS ed Enable. Assicuriamoci di avere la finestra delle connessioni del display configurata come quella in Fig. 12 e di avere il jumper della EB005 posizionato su "default". Non ci sono proprietà extra rilevanti per il display, quindi per quanto riguarda la configurazione dei componenti e delle connessioni abbiamo concluso. Prima di passare all'analisi del flowchart ci serve ancora la definizione della struttura dati sulla quale andremo a lavorare. Il

componente ADC di Flowcode ci offre la possibilità di acquisire la lettura del canale analogico considerato direttamente in tensione, ritornando un float. Questa caratteristica risulta molto comoda perché così sarà più facile effettuare la conversione in temperatura. In virtù di queste considerazioni creiamo una variabile di tipo "virgola mobile" e chiamiamola "Tensione" (come illustrato in Fig. 13) con la quale immagazzineremo il valore di tensione letto sul canale AN0. Creiamo anche una ulteriore variabile, chiamandola "Temperatura", sempre di tipo "virgola mobile", all'interno della quale immagazzineremo il corrispondente valore di temperatura.

Passiamo ora alla realizzazione dello schema a blocchi che sarà composto da un flowchart principale e da tre macro:

- Inizializzazione del sistema,
- Lettura del sensore di temperatura,
- Visualizzazione della lettura.

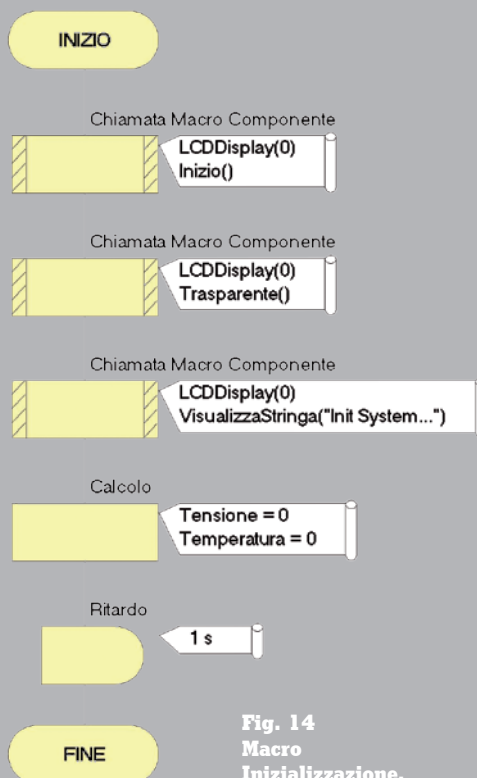


Fig. 14
Macro
Inizializzazione.

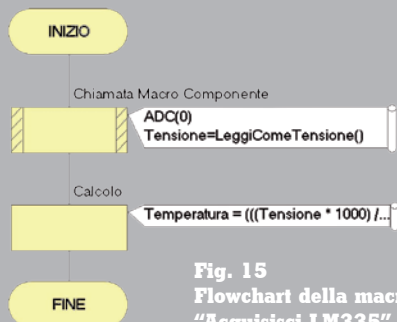


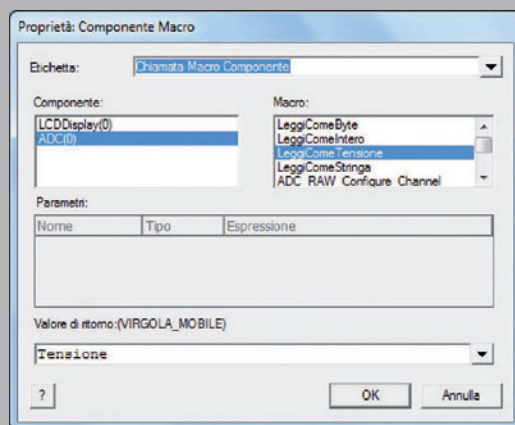
Fig. 15
Flowchart della macro
"Acquisisci LM335".

Partiamo dalla macro di inizializzazione, che ha lo scopo di inizializzare il display e le variabili e di scrivere una prima riga di testo per informare l'utente. Il flowchart della macro di inizializzazione è riportato in Fig. 14.

Come si vede, si utilizzando tre blocchi "Chiamata Macro Componente", un blocco "Calcolo" e un blocco "Ritardo".

In sequenza le operazione eseguite sono le seguenti: si inizializza il display, lo si pulisce, si visualizza la stringa "Init System...", si inizializzano le variabili e si genera un

Fig. 16
Finestra
proprietà
del blocco
"Chiamata
Macro
Componente".



ritardo di 1 secondo. Molto semplice. A questo punto passiamo alla macro di lettura del sensore di temperatura, che abbiamo chiamato "AcquisisciLM335", il cui flowchart è riportato in Fig. 15. Questa macro è il cuore del programma, eppure la potenza di Flowcode ci permette di ridurla a due soli blocchi. Il compito da eseguire è quello di acquisire il dato in tensione ed effettuare la conversione in temperatura, in modo che possa poi essere facilmente visualizzata dal display. Per prima cosa si sfrutta la macro "LeggiCo-

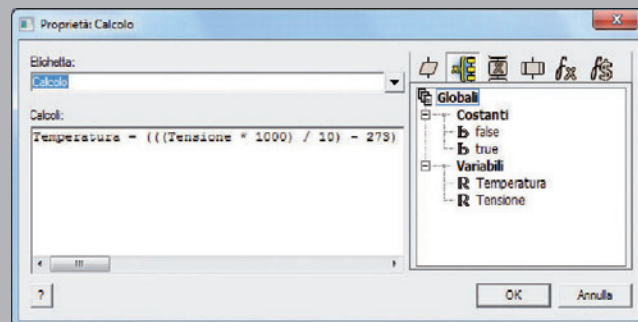


Fig. 17
Conversione
del dato in
tensione nel
corrispondente
valore di
temperatura.

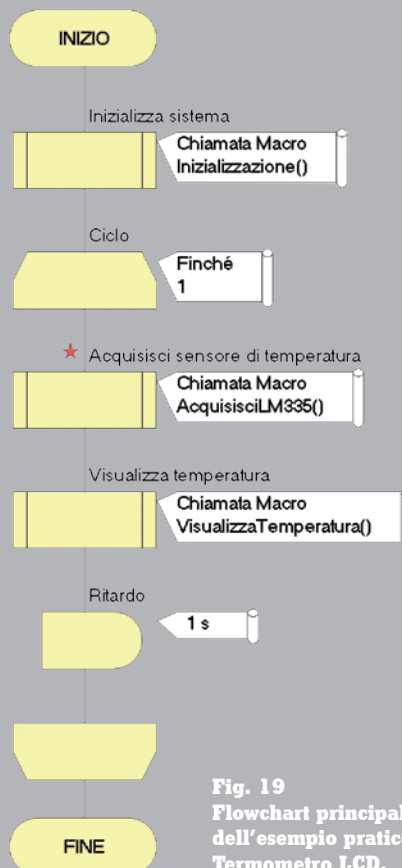
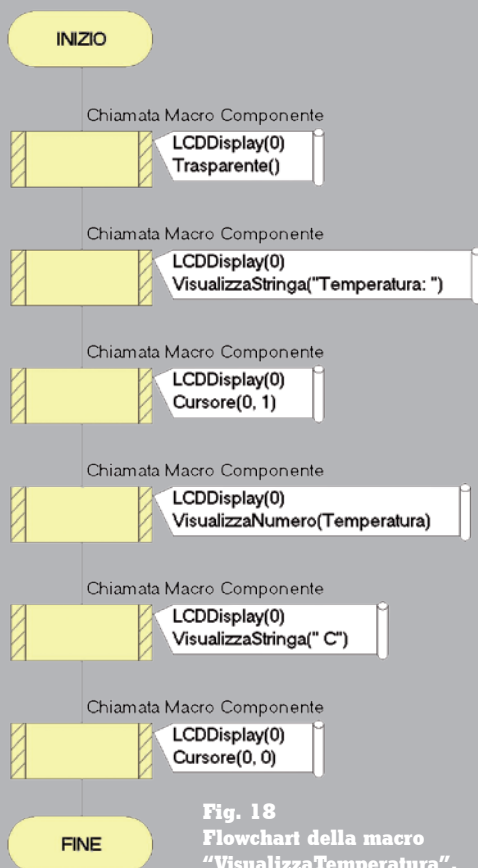
meTensione" del componente ADC, che permette di acquisire un canale analogico riportando direttamente il dato in tensione (in Volt), associabile ad una variabile di tipo float (virgola mobile).

Noi associamo il valore di ritorno alla variabile "Tensione", che avevamo creato in precedenza. In Fig. 16 è riportata la finestra di proprietà del blocco "Chiamata Macro Componente" che realizza l'operazione.

A questo punto bisogna convertire il dato letto in un dato di temperatura. L'LM335 ha un segnale in uscita proporzionale alla temperatura con fattore di proporzionalità di 10mV/Kelvin. Invertiamo la formula, scaliamo il valore di tensione da V a mV e sottraiamo 273 per ottenere la temperatura in gradi Celsius. Per introdurre questa formula nel codice usiamo un blocco calcolo, come riportato in Fig. 17.

Il dato immagazzinato all'interno della variabile "Temperatura" è già correttamente convertito in gradi Celsius, quindi basterà convertirlo in codifica ASCII per ottenere la visualizzazione corretta. Ora anche la seconda macro è completata. Passiamo all'ultima macro da realizzare, la macro di visualizzazione, che chiameremo "VisualizzaTemperatura", il cui flowchart è riportato in Fig. 18.

In questo flowchart usiamo solo chiamate a macro di componenti, in quanto il dato è già pronto e l'unica cosa che dobbiamo fare è scriverlo correttamente sul display, sfruttando le funzioni a corredo del componente LCD nonché le stringhe di contorno. Per prima cosa si pulisce il display, in modo che non rimangano porzioni di stringhe scritte in precedenza. Fatto questo si visualizza la stringa "Temperatura:", come stringa di intestazione. A questo punto si usa la funzione "Cursore" per muoversi sulla seconda riga, che verrà usa-



ta per visualizzare il dato. Qui possiamo usare la macro "VisualizzaNumero" per visualizzare su display il valore immagazzinato nella variabile "Temperatura" che, come abbiamo visto prima, contiene il valore della temperatura in gradi Celsius. Ora non ci rimane altro da fare che visualizzare il carattere "C" per indicare che la temperatura è in gradi Celsius e riportare il cursore alla posizione di partenza: la nostra terza macro è così pronta. Ora abbiamo le tre macro pronte all'uso, l'ultima operazione da fare è metterle insieme sul flowchart principale e lanciare la compilazione. Il flowchart principale è riportato in Fig. 19.

Come potete vedere è stata chiamata prima di tutto la macro "Inizializzazione". Poi, all'interno di un ciclo while infinito, sono state inserite le due chiamate alle macro "AcquisisciLM335" e "VisualizzaTemperatura". La prima macro acquisisce il valore di tensione proveniente dal sensore di temperatura e passa i dati alla seconda, che li visualizza sul display LCD. Per evitare un eccessivo effetto "Refresh" sul display

è stato inserito un ritardo software di un secondo.

Possiamo ora procedere con la compilazione e l'esecuzione del programma. Una volta programmato il micro, se abbiamo eseguito tutto correttamente, vedremo comparire il valore della temperatura letta dal sensore sul display. Se la lettura non fosse corretta, possiamo agire sul potenziometro del circuito di calibrazione per regolarla.

In questa puntata abbiamo iniziato a prendere confidenza con i componenti di Flowcode, realizzando un primo programma di un certo peso (un dispositivo che ha una reale funzione), e abbiamo cominciato a renderci conto di come, inserendo i componenti appositi e collegando pochi blocchi, sia possibile implementare in breve tempo anche funzioni di una certa complessità. Nella prossima puntata continueremo questo percorso, esaminando in dettaglio le interfacce di comunicazione che ci permettono di realizzare sistemi più complessi, interfacciati a periferiche esterne più intelligenti.