

In omaggio
SOLUZIONI EMBEDDED 16 bit
MICROCHIP

mensile di progettazione elettronica, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica

€ 6,00

Anno XX - n. 183
Febbraio 2014

FORM1

**LA PRIMA STAMPANTE 3D
IN STEREO LITOGRAFIA
LOW-COST**



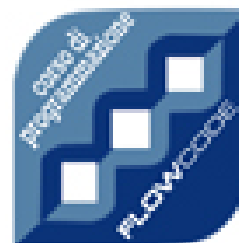
**Telecontrollo
da Smartphone
con Arduino e Bluetooth**



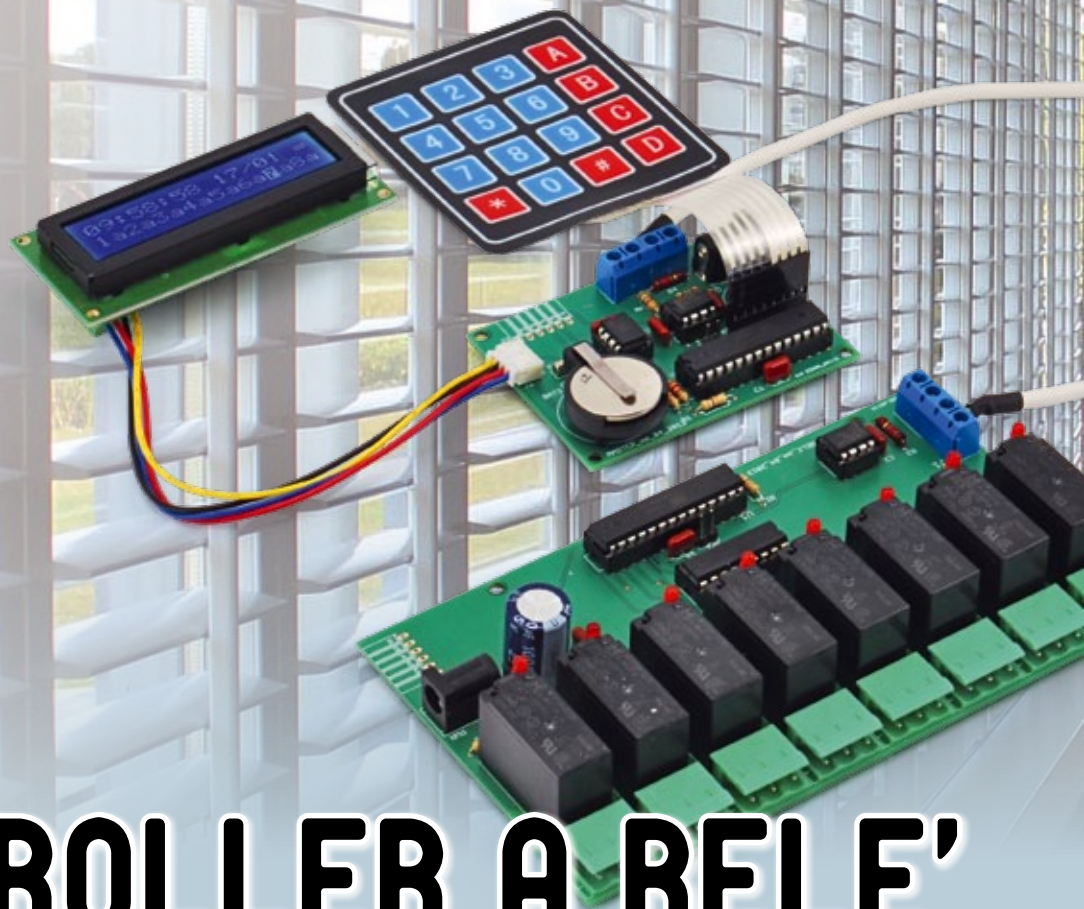
**HeartThrob
batticuore a LED**



- Controllo relè remoto via RS485
- Shield I²C per Raspberry Pi
- myKIT, misura di irradianza
- RFTide e Arduino per una rete domotica
- Ricevitore per radiocomando mono e bicanale



Sistema programmabile composto da una scheda a 8 relé installabile dove si trovano gli utilizzatori e comandata a distanza da un'unità Master via seriale ad elevata immunità ai disturbi.



CONTROLLER A RELE' GESTITO DA RS485

..... del Prof. MICHELE MENNITI e LEONARDO MILIANI

In commercio proliferano schede a relé di ogni tipo, ma la gran parte di esse offre prestazioni limitate: quasi tutte sono predisposte per essere pilotate da sistemi programmabili, il che costringe l'utente a dover scrivere lunghi firmware per ottenere un certo modo di funzionamento; esistono poi le schede direttamente programmabili, ma nella maggior parte dei casi si tratta di impostare semplici comandi ON-OFF... al massimo con una temporizzazione aggiuntiva. Inoltre molte delle unità a relé reperibili in commercio non sono in grado di commutare tensioni di 220-250 Vca, quindi non possono gestire direttamente carichi alimentati con la tensione di rete e devono avvalersi di servorelé. Queste limitazioni possono essere superate realiz-

zando il circuito proposto in questo articolo, che è un sistema di gestione a relé in cui l'unità di commutazione (cioè quella che ospita i relé) permette di controllare dispositivi funzionanti a 220 Vca ed è separata dall'unità di controllo, il che permette di delocalizzarla portandola sul luogo dove si trovano gli attuatori da comandare. Questo approccio consente di limitare la lunghezza dei collegamenti di potenza, limitando i cavi lunghi a quelli della sola linea seriale di comando.

Dal punto di vista hardware, il progetto prevede tre distinte sezioni: la centralina Master (quella che governa l'unità a relé), il display LCD (per visualizzare le funzioni e l'attività) e la centralina Slave/relé (ossia l'unità che contiene i relé di comando).

La comunicazione tra centralina Master e centralina Slave avviene tramite un canale seriale RS485, che garantisce un collegamento poco sensibile alle interferenze elettriche (idoneo quindi agli ambienti disturbati) e realizzabile a distanze dell'ordine del km.

SCHEMA ELETTRICO CENTRALINA MASTER

Il cuore della centralina è il notissimo microcontrollore ATmega328P-PU (U1) della ATMEL, qui usato in configurazione con oscillatore interno a 8 MHz. Ai suoi pin 12÷19 è collegata (mediante il connettore J4) una tastiera a matrice a membrana, del tipo esadecimale; sui pin 16÷19 (quelli relativi alle righe della tastiera) sono attive le resistenze interne di pull-up del microcontrollore, in modo da mantenere questi pin (impostati come ingressi) allo stato logico HIGH; i pin 12÷15 (relativi alle colonne della tastiera) sono impostati, sempre da software, come uscite LOW multiplexate (cioè viene posto a livello logico basso uno solo dei

pin alla volta); la combinazione determinata dalla pressione di ogni singolo tasto fornirà al microcontrollore l'indicazione per la sua identificazione.

I pin 2, 3 e 26 sono coinvolti nella comunicazione tra le due centraline, che avviene ricorrendo al protocollo RS-485; essi svolgono rispettivamente la funzione di trasmissione dati (TX), ricezione dati (RX), commutazione trasmissione/ricezione (EN_RS), e controllano di conseguenza il chip U2 (MAX485), il quale a sua volta opera mediante le due linee A e B, responsabili appunto dello scambio dati con un chip "gemello" localizzato sull'altra centralina. Sul connettore J2, oltre ai segnali A-B, arriva l'alimentazione, il cui compito è fornire la tensione necessaria al funzionamento della centralina Slave. Quindi con soli quattro fili, la cui lunghezza, in base allo standard, potrebbe raggiungere i 1.000 metri, la centralina Master pilota quella Slave e i suoi otto relé. Nello schema è visibile un jumper JP1 collegato al pin 23: esso

permette di escludere il circuito RS485 e di attivare l'opzionale circuito RF che permette la comunicazione wireless tra le due centraline. In questo caso i pin coinvolti, oltre ai già citati 2 e 3 (TX e RX), diventano il 25 (commutazione TX/RX) ed il 24 (abilitazione trasmissione RF). Allo stato attuale l'opzione RF non è disponibile, quindi se decidete di implementarla successivamente dovrete togliere dal circuito il chip U2 e i suoi componenti di contorno C2, R4, R5 e R6. Allo stato attuale il jumper JP1 si può non montare o deve essere portato su OFF.

I pin 27 e 28 di U1 sono destinati alla comunicazione di bus mediante protocollo I²C, con le due indispensabili resistenze di pull-up (R2 e R3). Su tale bus si possono collegare svariati dispositivi, ognuno dei quali diventa raggiungibile mediante un comando diretto al suo indirizzo, che è univoco e viene stabilito in fabbrica o si può impostare mediante i piedini riservati all'impostazione. Nel nostro caso,

CARATTERISTICHE TECNICHE della Centralina GP

Unità di controllo a microcontrollore programmabile mediante tastiera esadecimale e display LCD, alimentazione 5 Vcc.

Unità di attuazione con 8 relé 220 Vca - 10 A, con indicazione a LED della condizione ON/OFF, alimentazione 5 Vcc.

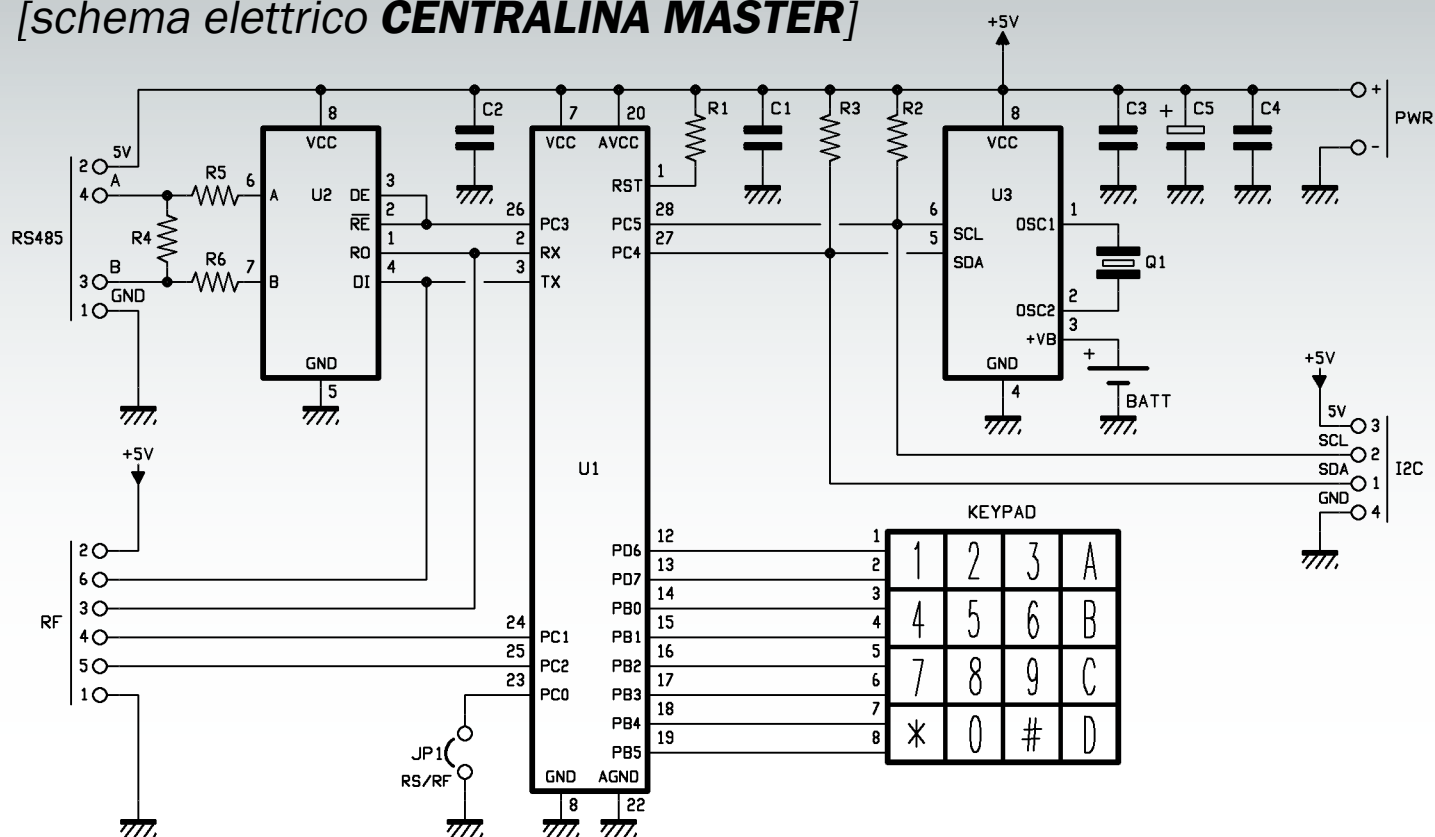
Sistema di comunicazione RS485 tra Unità di controllo e Unità di attuazione, mediante 4 fili a distanza massima di 1km.

Predisposizione (opzione) per il collegamento wireless tra le due Unità.

Sei modalità di programmazione degli attuatori:

- **"a"** - lo stato ON/OFF dell'uscita è liberamente gestibile;
- **"b"** - lo stato ON/OFF dell'uscita è gestibile previo inserimento di una password di controllo (che viene chiesta per entrambe le operazioni di attivazione/disattivazione del relé);
- **"c"** - lo stato ON/OFF dell'uscita è liberamente gestibile ma la durata dell'attivazione è temporizzata: una volta selezionata l'uscita bisogna inserire il tempo di durata dell'attivazione (max. 99 ore e 99 minuti), al termine del quale l'uscita si disattiva automaticamente;
- **"d"** - identica alla modalità "c" ma con password di protezione;
- **"*"** - lo stato ON/OFF dell'uscita è completamente temporizzato: al momento dell'impostazione vengono chiesti l'orario di attivazione e quello di disattivazione;
- **"#"** - questa modalità è molto particolare: l'uscita è gestibile, all'interno di un range orario impostato dall'amministratore, solo da un gruppo di utenti dotati di password personale; ogni utente dispone solo di una propria fascia oraria, sempre stabilita dall'amministratore; l'utente, pur disponendo di password di accesso, può gestire l'uscita solo nella fascia oraria a lui attribuita, e sempre che essa sia compatibile con il range orario associato all'uscita stessa. L'uscita si disattiva o al termine della fascia oraria dell'utente o al termine del proprio range orario, anche se la fascia dell'utente dovesse andare oltre.

[schema elettrico **CENTRALINA MASTER**]



al bus I²C abbiamo collegato due dispositivi: il gestore dell'orologio e l'elettronica di controllo del display LCD; mentre il primo si trova direttamente sul PCB della centralina Master, il secondo è collegato ad esso mediante il connettore a 4 poli J1. L'integrato U3, il DS1307 della Dallas (indirizzo dalla libreria RTCLib-master), è in grado di memorizzare e fornire l'ora esatta, con precisione legata al quarzo Q1 (il tipico quarzo a cilindretto degli orologi), e mantenimento del dato in memoria anche a centralina spenta, grazie a una batteria tampone (siglata BAT1 nello schema elettrico) che è una pila a bottone di tipo CR2032. I condensatori da 100 nF (C1÷C4) servono a filtrare eventuali disturbi, mentre la R1 si occupa di mantenere stabilmente HIGH lo stato logico del pin di RESET del microcontrollore U1. L'alimentazione deve essere fornita da una fonte di 5 V c.c. con circa 1 A di corrente erogabile, in

modo da alimentare entrambe le centraline ed il display LCD; il condensatore elettrolitico C5 è dimensionato in modo da garantire stabilità della tensione anche in condizioni di massimo assorbimento di picco all'accensione.

SCHEMA ELETTRICO DEL DISPLAY LCD

Un normale display a cristalli liquidi basato sul controller HD44780 o compatibile, nella configurazione a 4 bit richiede al minimo sei linee di collegamento, che diventano sette se si vuole gestire in modo automatico anche la retroilluminazione, come nel nostro caso. Il collegamento della tastiera a matrice e la gestione della comunicazione, con predisposizione per l'RF, nonché la connessione della sezione orologio, purtroppo lasciano liberi un numero inferiore di pin; ecco perché abbiamo adottato la soluzione del display collegato via bus I²C. Visto però il costo piuttosto elevato dei display

dotati nativamente di questa interfaccia, abbiamo pensato di progettare una scheda molto semplice ed economica che consente di trasformare un normale display LCD nella equivalente versione I²C.

Il cuore del circuito è rappresentato dall'integrato PCF8574A (U1), un 8 bit I/O expander per bus I²C capace di pilotare ben otto linee I/O mediante le due sole linee SDA/SCL del bus I²C (per i nostri scopi ne sono sufficienti sette). La gestione del PCF8574A richiede una libreria specifica chiamata *LiquidCrystal_I2C*. In tal modo il display LCD (nel nostro caso il comune 16x2, ma potrebbe essere anche un 20x2 o un 20x4: funziona comunque) viene usato nella classica configurazione a 4 bit di dati più i segnali RS e E; un'ulteriore linea è dedicata al controllo via software della retroilluminazione, che consiste nel pilotare un transistor PNP, per cui la condizione LOW



Fig. 1
Collegamento tra unità Display e centralina Master.

del pin 12 di U1 lo porta in conduzione, alimentando così i LED della retroilluminazione. La R2 ha il compito di limitare la corrente; il valore attuale è adatto ai display della precedente generazione, sprovvisti della resistenza di limitazione interna, per i quali R2 va calcolato sulla base delle informazioni ricavabili dal data-sheet del display stesso. Se usate un display con resistore interno già dimensionato dal costruttore, R2 non serve, per cui può essere un semplice ponticello. Il trimmer R3 serve per la regolazione del contrasto del display mentre i tre pin 1, 2, 3 (A0, A1, A2) di U1, posti tutti a massa, fissano l'indirizzo I²C-Bus dello stesso a 0x38. Nel procurare questo integrato prestate attenzione perché esiste anche la versione PCF8574 (senza la "A" finale) che funziona ugualmente su questo circuito, ma ha un indirizzo diverso, che nella stessa

configurazione corrisponde a 0x20; se avete questa versione dovete modificare il firmware da noi previsto (scaricabile dal nostro sito www.elettronica-in.it).

SCHEMA ELETTRICO DELLA CENTRALINA SLAVE

L'unità Slave si basa ancora su un microcontrollore ATmega328P con clock ad 8 MHz; la parte relativa al collegamento (via cavo o RF) con la centralina Master è identica a quella già vista in precedenza, quindi non stiamo a spiegarla nuovamente. Ci limiteremo a descrivere la parte dedicata agli attuatori, che fa capo a otto relé (RL1÷RL8) con bobina a 5V e richiedono ognuno circa 50 mA di corrente; siccome questa corrente è troppa per un pin del microcontrollore, abbiamo interfacciato il micro con ULN2803, che è un array di otto Darlington NPN, ognuno dei quali sopporta fino a 500mA. Quest'ultimo è anche il valore della corrente massima erogabile dall'intero array; in pratica si dispone di 500 mA sfruttabili

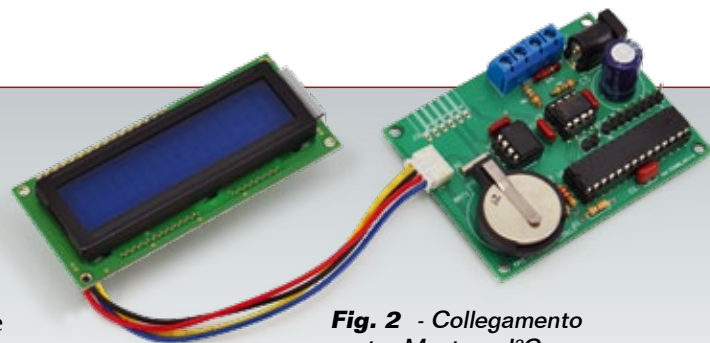


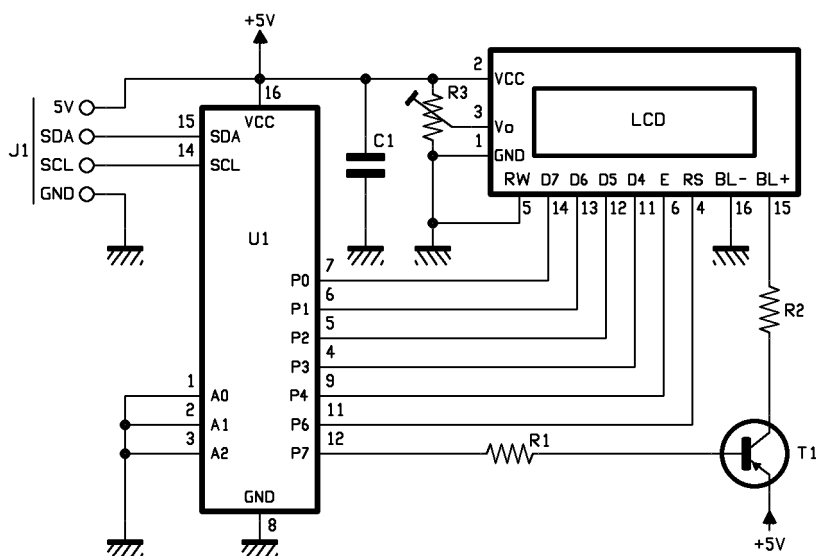
Fig. 2 - Collegamento tra Master e I²C.

su una sola uscita o suddivisi tra più uscite. Nel nostro caso avremo $50\text{mA} \times 8 = 400\text{mA}$ di erogazione totale quando tutti i relé sono attivi, quindi restiamo nei limiti. Ogni Darlington collega a massa il proprio relé di riferimento (del quale un capo è direttamente collegato ai 5 V) e al relativo LED, col risultato che quando il relé si eccita, si accende il LED corrispondente.

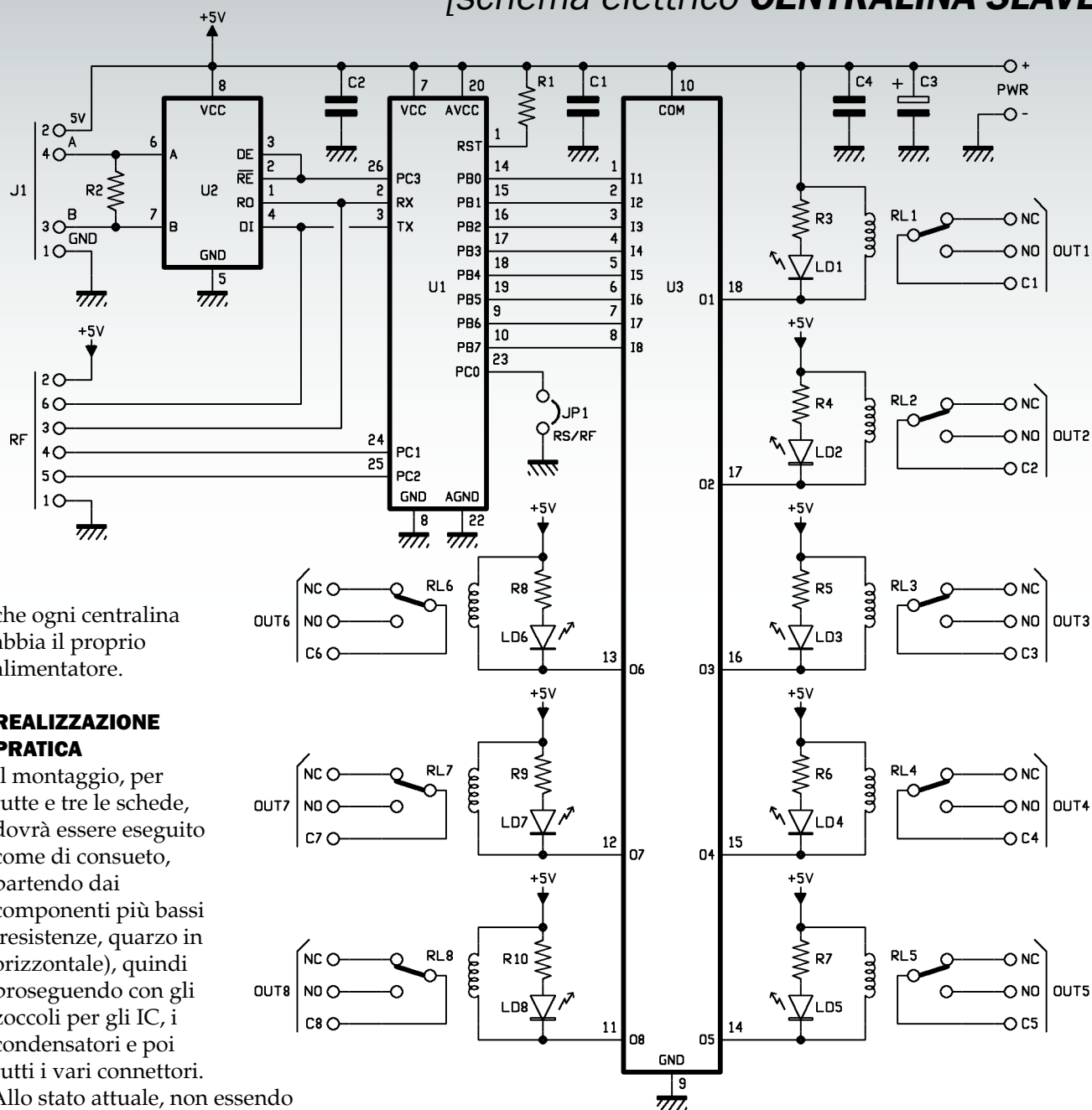
Il circuito è replicato per otto volte e gli otto relé, tramite l'ULN2803, sono pilotati dagli 8 bit del PORT B del microcontrollore, il cui funzionamento sarà approfondito quando affronteremo la parte software del progetto.

Anche in questo caso abbiamo previsto una presa esterna per l'alimentazione a 5 V, quindi ora è necessario chiarire come si debbano usare le due prese delle due centraline Master e Slave: se il collegamento tra le due centraline è fatto via cavo (RS485), basta collegare un alimentatore ad una qualsiasi delle due prese, in quanto il cavo porta l'alimentazione in modalità bidirezionale; in questo caso la presa lasciata libera potrà essere utilizzata anche come uscita 5 V di servizio, con corrente disponibile dipendente dall'alimentatore usato, tolti i consumi del circuito (ovviamente il cavetto di collegamento tra le due centraline, riguardo ai due fili dell'alimentazione, dovrà essere di sezione adeguata). Se invece si dovesse far ricorso al collegamento opzionale RF tra le due centraline, sarà necessario

[schema elettrico DISPLAY LCD]



[schema elettrico **CENTRALINA SLAVE**]



che ogni centralina abbia il proprio alimentatore.

REALIZZAZIONE PRATICA

Il montaggio, per tutte e tre le schede, dovrà essere eseguito come di consueto, partendo dai componenti più bassi (resistenze, quarzo in orizzontale), quindi proseguendo con gli zoccoli per gli IC, i condensatori e poi tutti i vari connettori.

Allo stato attuale, non essendo disponibile l'opzione RF potete evitare di montare il jumper JP1 ed il connettore RF sia sul PCB Master (J3) che sul PCB Slave (J2). Il PCB LCD-I2C, che ospita il circuito controller I2C per il display andrà collegato al retro del display LCD mediante una fila di 16 strip, saldata su entrambe le parti (Fig. 1). Il collegamento tra le schede è estremamente semplice: il PCB della centralina Master si collega, tramite il connettore

J1, al connettore J1 del circuito dell'unità display ad interfaccia I2C-Bus (Fig. 2); i due connettori sono diversi ma il collegamento va fatto pin-to-pin (1 con 1, 2 con 2, ecc.).

Il PCB della centralina Master si collega tramite il connettore J2 al connettore J1 del PCB SLAVE, sempre pin-to-pin (Fig. 3). Infine il connettore femmina del tastierino esadecimale a

membrana si infila sullo strip J4 (Fig. 4).

CARICAMENTO DEL FIRMWARE

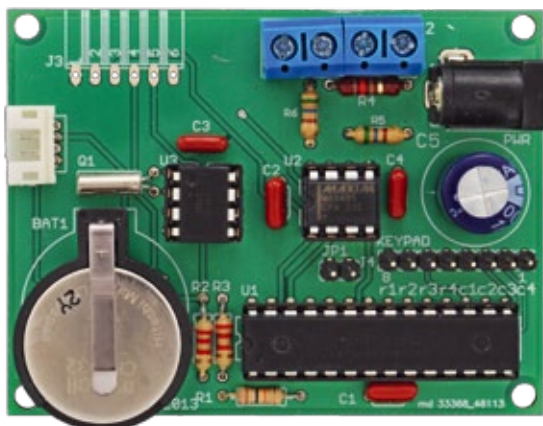
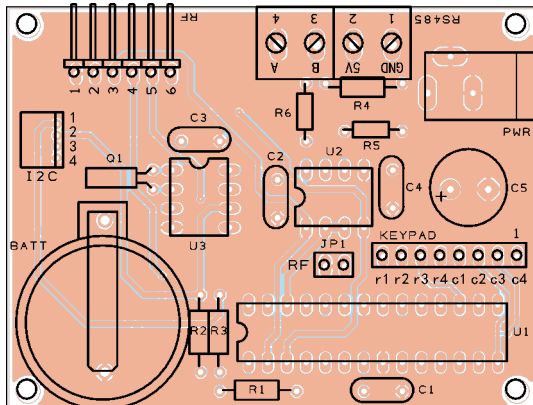
Prima di poter utilizzare i circuiti, i loro microcontrollori dovranno essere opportunamente programmati per il funzionamento dell'intero circuito; potrete scaricare tutto il materiale necessario dalla sezione download del sito *www*.

Elenco Componenti:

R1: 10 kohm
 R2, R3: 2,2 kohm
 R4: 120 ohm 1/2W
 R5, R6: 560 ohm
 C1 ÷ C4: 100 nF poliestere
 C5: 1000 μ F 16VL
 elettrolitico
 U1: ATMEGA328P-PU
 (MF1108)
 U2: MAX485
 U3: DS1307
 Q1: Quarzo 32768Hz
 KEYPAD: Tastiera
 a matrice 4x4

Varie:

- Morsetto 2 poli (2 pz.)
- Zoccolo 4+4 (2 pz.)
- Zoccolo 14+14
- Jumper
- Plu alimentazione
- Strip maschio 2 vie
- Strip maschio 8 vie
- Strip femmina 6 vie 90°
- Strip femmina 8 vie
- Presa JST 4 poli
- Porta batterie CR2032
- Batteria CR2032
- Circuito stampato



Effettuate queste operazioni preliminari si può passare alla programmazione dei microcontrollori, ricorrendo alla tecnica ISP, più volte spiegata sulle pagine di questa Rivista (in particolare nei numeri 160 e 164).

Prima del caricamento del firmware le due MCU devono essere settate mediante i "fuse" affinché possano operare alla frequenza di 8 MHz usando l'oscillatore interno; i valori da impostare sono i seguenti: Low Fuse: \$e2, High Fuse: \$de, Extended Fuse: \$05. Terminata la programmazione dei due micro sarà possibile inserirli nei rispettivi zocchi dei PCB Master e Slave, ovviamente badando bene a non confonderli tra essi, e quindi dare alimentazione al circuito.

UTILIZZO DELLA CENTRALINA

Tutte le operazioni di setup e di comando avvengono tramite la centralina Master; sulla Slave

elettronica.in, sotto forma di un'unica cartella compressa, denominata *CentralinaGP.rar*, la quale contiene:

- *Boards.txt*: è un file di testo, contenente le righe da aggiungere al file originale dell'IDE, avente sempre nome boards.txt; queste righe permetteranno

di operare correttamente con la programmazione ISP dei microcontrollori ATmega328P;

- *Firmware*: è una cartella contenente i due sketch per la programmazione dei microcontrollori; sono contrassegnati dalla denominazione iniziale master e slave, seguiti dalla relativa versione (master_x_yy.ino e slave_x_YY.ino);
- *Libraries*: sono tre cartelle contenenti altrettante librerie indispensabili per il buon funzionamento del firmware; esse andranno semplicemente inserite nella cartella "libraries" dell'IDE 1.0.5 di Arduino (se non si dispone di questa versione è vivamente consigliato scaricarla, perché è molto più stabile e funzionale delle precedenti 1.0.x).

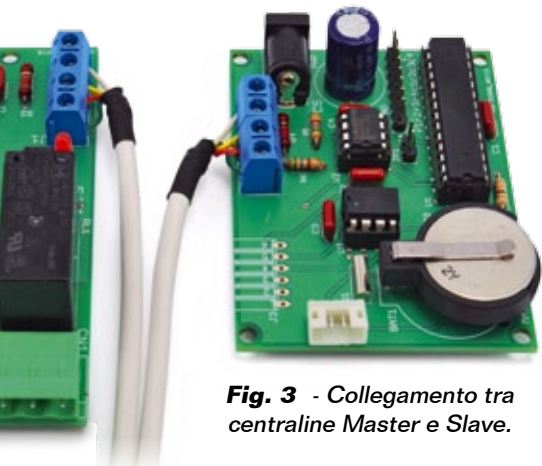


Fig. 3 - Collegamento tra centraline Master e Slave.



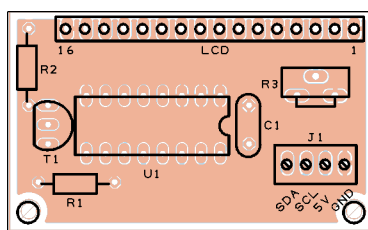
Fig. 4 - Collegamento tra centralina Master e Keypad.

Elenco Componenti:

R1: 2,2 kohm
R2: Vedi testo
R3: Trimmer 10 kohm
C1: 100 nF poliestere
T1: BC328
U1: PCF8574A
LCD: Display LCD 16x2

Varie:

- Zoccolo 8+8
- Morsetto 2 poli (2 pz.)
- Circuito stampato



l'utente non interviene direttamente. Appena avviato, la Master mostra sul display un messaggio di benvenuto riportante la versione del firmware attualmente in uso. Successivamente la centralina esegue alcuni test di autodiagnosi e visualizza a video le informazioni sui passi in esecuzione (**Fig. 5**). Prima di tutto viene controllato lo stato dell'integrato RTC, per verificare se l'orario è preservato oppure no: in quest'ultimo caso la Master visualizza il messaggio di inizializzazione dell'RTC con un orario predefinito. In seconda istanza viene controllato il canale di comunicazione utilizzato per dialogare con la Slave mediante la lettura dello stato del jumper JP1 e successivamente la Master verifica se effettivamente sia riuscito a comunicare con lo Slave riportando a video l'esito dell'operazione: se la connessione si è stabilita senza problemi, verrà visualizzata la scritta "Slave CONNESSO", altrimenti sarà stampata la scritta "Slave ASSENTE". In quest'ultimo caso è bene ricontrollare i collegamenti fra le due schede, altrimenti non sarà possibile pilotare la Slave. Quando il collegamento è assente, entrambe le unità cercano il dia-

logo con l'altra scheda: è sufficiente ripristinare la connessione per far sì che la comunicazione venga riattivata in automatico dall'autodiagnosi.

Inizia ora la configurazione delle uscite e delle eventuali password di attivazione (avviso a video: "Caricamento dati"); al primo avvio del programma, questo esegue la formattazione della EEPROM integrata nel microcontrollore, segnalando a display l'avanzamento dell'operazione, altrimenti vengono recuperati, sempre dalla EEPROM, i dati già memorizzati.

Una volta terminate le operazioni di verifica iniziale del sistema, la centralina entra in modalità operativa, visualizzando l'orario e la data correnti e lo stato delle uscite dell'unità Slave. Il display della master appare normalmente come in **Fig. 6**.

Il carattere "=" che si vede in alto a destra simboleggia due cavi e sta ad indicare la connessione RS485. La riga inferiore riporta otto coppie di caratteri alfanumerici così composte:

- un numero, che indica l'uscita ed il suo stato; se il numero appare normalmente, l'uscita è disattivata, mentre se viene visualizzato negativo rispetto

- allo sfondo, l'uscita è attiva;
- una lettera in minuscolo, che indica la modalità dell'uscita.

Alla prima accensione tutte le uscite si troveranno in modalità "a", quindi basterà premere il tasto del relativo numero 1÷8 per attivarle o (con una nuova pressione) disattivarle. In queste condizioni il tasto "*" permette di accendere tutte le otto uscite contemporaneamente, mentre il tasto "#" le spegne; l'uso di questi due tasti è sempre valido ed agisce esclusivamente sulle uscite impostate con modalità "a".

Per personalizzare lo stato delle uscite si deve accedere al menu amministratore, premendo il tasto "0" del tastierino per un tempo superiore a 1,5 secondi. La centralina risponde chiedendo la password di amministratore (**Fig. 7**), che per impostazione predefinita è fissata a "123456" (può essere cambiata direttamente dal menu di amministrazione), quindi appare la prima voce del

Centralina
elettronica 4.04

Inizializzo RTC

RS485 in uso

Slave.. CONNESSO

Caricamento dati

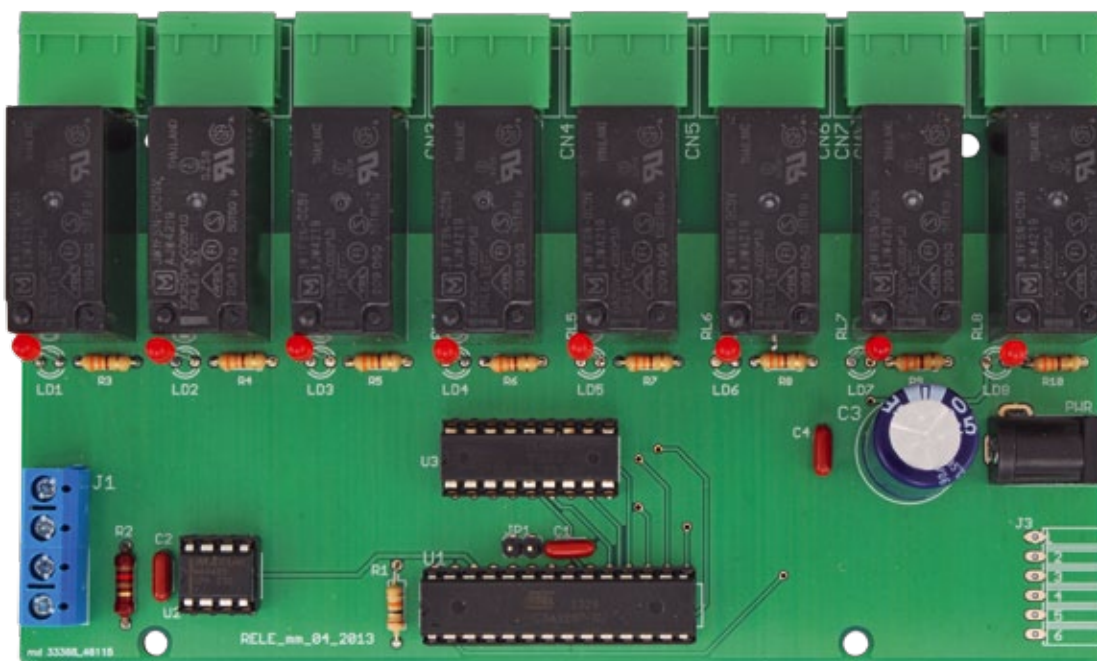
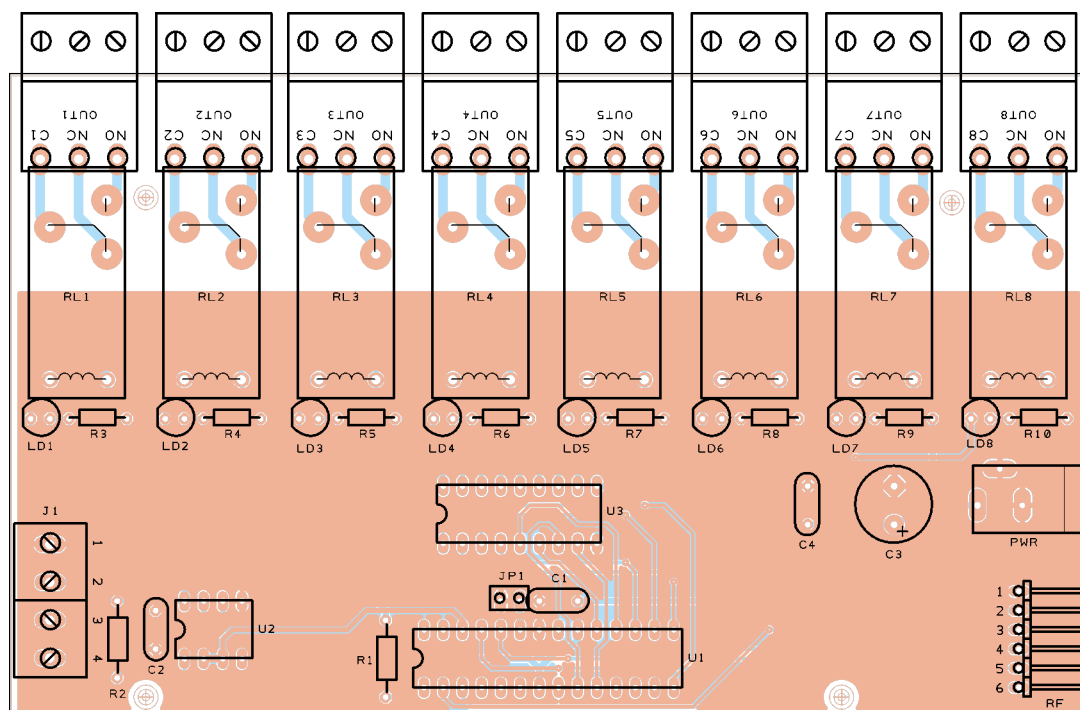
Formatto EEPROM
Attendi ***

Fig. 5 - I messaggi durante la procedura di inizializzazione.

10:10:10 18/08 =
1a2a3a4a5a6a7a8a

Fig. 6 - Il display mostra lo stato delle uscite.

[piano di montaggio **CENTRALINA SLAVE**]



Elenco Componenti:

R1: 10 kohm

R2: 120 ohm 1/2W

R3÷R10: 330 ohm

C1, C2, C4: 100 nF poliestere

C3: 1000 µF 16VL elettrolitico

LD1÷LD8: LED 3 mm rosso

U1: ATMEGA328P-PU (MF1108)

U2: MAX485

U3: ULN2803

RL1÷RL8: Relé 5V 1 scambio

Varie:

- Zoccolo 4+4

- Zoccolo 9+9

- Zoccolo 14+14

- Jumper

- Plu alimentazione

- Morsetto 2 poli (2 pz.)

- Morsetto 3 poli componibile (8 pz.)

- Strip maschio 2 vie

- Strip femmina 6 vie 90°

- Circuito stampato

menu, la cui struttura è la seguente (le voci sono riportate così come vengono visualizzate):

1. **Set uscite**
 - 1.1. Imposta usc.
 - 1.2. Reset uscite
 - 1.3. Esci

2. **Set utenti**
 - 2.1. Crea utente
 - 2.2. Mod. utente
 - 2.3. Canc. utente
 - 2.4. Canc. tutti
 - 2.5. Esci

3. **Set orologio**
 - 3.1. Imposta ora
 - 3.2. Imposta data
 - 3.3. Esci

4. **Funz. Avanzate**
 - 4.1. Cambia pswd
 - 4.2. Avv. c/ripr.
 - 4.3. Riavvia
 - 4.4. RESET
 - 4.5. Esci

5. **Info**
 - 5.1. Ver.firmware
 - 5.2. SRAM libera
 - 5.3. Esci

6. **Esci**

Ecco come muoversi in questo ambiente:

- il tasto "*" permette di tornare indietro di una posizione;
- il tasto "#" permette di andare avanti di una posizione;
- il tasto "0" seleziona la voce visualizzata in quel dato momento;
- il tasto "D" serve per terminare un'operazione in corso nei menu secondari.

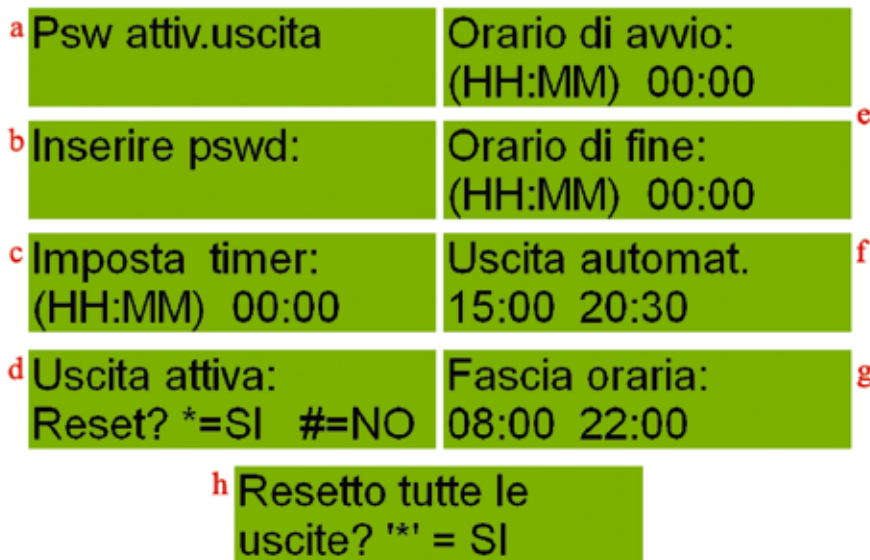
PASSWORD ADMIN:

Fig. 7 - La richiesta di password amministratore.

Seleziona uscita
1a2a3a4a5a6a7a8a

Fig. 8 - La selezione delle uscite.

Fig. 9 - Messaggi della fase di programmazione e uso riguardanti Set uscite.



Per impostare le modalità delle uscite è sufficiente confermare con "0" la prima voce di menu: "1. Set uscite", così da accedere al sottomenu secondario; anche qui bisogna selezionare la prima voce: "1.1 Imposta usc.". Premendo "0" sul display comparirà l'immagine di cui alla Fig. 8. Alla base del numero "1" apparirà un trattino lampeggiante: è il cursore di selezione che si può spostare a sinistra o a destra mediante i soliti tasti "*" (sinistra) e "#" (destra).

Premendo "0" il numero dell'uscita indicata dal cursore viene visualizzato invertito, ad indicare la selezione dell'uscita e la possibilità di modificarne la tipologia; quest'ultima si imposta premendo direttamente il tasto corrispondente: "A", "B", "C", "D", "*" o "#". Se la modalità selezionata necessita di dati aggiuntivi, questi vengono chiesti all'utente: per esempio se si attiva un'uscita di tipo "b", la centralina risponderà richiedendo l'assegnazione della password di attivazione dell'uscita (Fig. 9a); l'amministratore ha dieci secondi di tempo per inserirla. La lunghezza della password varia da 1 a 6 caratteri ed è ammesso

qualunque carattere fra i 16 della tastiera. Immettendo 6 caratteri, l'inserimento termina immediatamente e la password viene memorizzata; volendo impostare una password più corta, basta immettere i caratteri desiderati e poi attendere il time-out (la scadenza dei dieci secondi), alla fine del quale il software memorizza solo i caratteri inseriti.

Una volta terminato l'inserimento della password il software torna alla schermata precedente. Per terminare la modifica delle uscite, premere il tasto "D": il display riporta alla voce di menu "1.1 Imposta usc.". Per terminare occorre prima selezionare la voce "1.3 Esci" (ritorno al menu principale) e poi la voce "6. Esci" (uscita dal menu di amministrazione), ritornando così alla modalità di funzionamento normale della centralina.

Sul display comparirà adesso la lettera "b" a destra del numero corrispondente all'uscita che è stata appena modificata. Per verificare la correttezza dell'operazione, premere il numero corrispondente all'uscita: il firmware risponderà chiedendo l'inserimento della password che è stata precedentemente impostata (Fig. 9b). Se la password inserita

Applicazioni del sistema

Ecco alcuni esempi d'uso delle sei modalità di programmazione previste.

- a) in questa modalità tutte le uscite funzionano con la classica modalità on-off libera; si può realizzare una sorta di pannello di comando per attivare a necessità attuatori di qualsiasi tipo: luci, elettrodomestici, tapparelle, ventilatori.
- b) questa modalità è simile alla precedente, ma l'aggiunta di un accesso controllato da password permette di entrare in un'area riservata mediante controllo di un accesso elettrico o di accendere un attrezzo in una palestra riservato ai professionisti; può anche servire per evitare che la modalità "a" possa essere controllata inavvertitamente da bambini.
- c) il timer integrato in questa modalità permette l'utilizzo a tempo di un attuttore: un asciugacapelli in palestra, l'impianto luci di un campo da calcetto, o anche una console da videogiochi in casa.
- d) è la modalità "c" associata ad una password e permette tutte le precedenti applicazioni senza la necessità di un controllo supervisore, limitando l'uso degli attuatori alle sole persone autorizzate (quindi in possesso di password).
- * la predefinizione di una fascia oraria di funzionamento ad attivazione/disattivazione automatica distinta per ciascun'uscita è utile in tutti quei casi in cui si voglia automatizzare un attuttore, come quello delle luci esterne di una villa o l'impianto di irrigazione o un sistema di protezione perimetrale.
- # questa modalità permette di distribuire una fascia oraria globale impostata per un attuttore, tra i sedici possibili utenti previsti. Immaginiamo di disporre di un solo macchinario il cui uso è a noleggio orario presso la sede; l'orario in cui il macchinario può lavorare è stabilito dalle 14:00 alle 20:00 (fascia oraria preimpostata dall'admin); gli utenti abilitati, e quindi dotati di

"password utente #", sono in questo caso quattro:

- **Utente1:** fascia oraria personale dalle 14:00 alle 14:59;
- **Utente2:** fascia oraria personale dalle 15:00 alle 16:59;
- **Utente3:** fascia oraria personale dalle 18:00 alle 18:59;
- **Utente4:** fascia oraria personale dalle 19:00 alle 20:59.

Come si vede, gli utenti 1 e 3 hanno a disposizione un'ora, mentre gli utenti 2 e 4 dispongono di due; ogni utente può operare solo durante la propria fascia oraria e se tenta di accedere al di fuori di essa, la password, pur essendo valida, non viene accettata. Se un utente inizia ad operare dopo l'ora di avvio prevista, non può andare oltre l'orario finale ad esso assegnato. In casi di "conflitto orario" con quello globale della Centralina (errore di impostazione da parte dell'admin) vale comunque l'orario di quest'ultima più di quello dell'utente. Nell'esempio descritto la centralina ha un range 14:00-20:00, mentre l'Utente 4 dovrebbe terminare alle 20:59; questo non accadrà mai, in quanto alle 20:00 la centralina bloccherà l'uscita, per riprendere la propria attività alle 14:00 del giorno dopo. In questi casi si posticipa l'orario finale della Centralina o si anticipa quello dell'utente. L'altra possibilità di conflitto riguarda la sovrapposizione oraria tra due utenti; in questo caso ognuno dei due può usare l'apparecchio. L'utente che per primo attiva l'uscita acquisisce il diritto di uso esclusivo, gestito automaticamente dal firmware. Ogni utente abilitato ha accesso ad una uscita predefinita, e la sua password vale solo per essa nell'orario di attivazione. Per l'accesso ad un'altra uscita funzionante in modalità "#" l'amministratore deve creare una nuova utenza specifica, ovviamente dotata di altra password. Il numero totale di utenti è preimpostato a 16; tale numero è aumentabile modificando le impostazioni del firmware, ma tenendo conto del fatto che la EEPROM dell'ATmega328P è di soli 1.024 byte e che un numero elevato di utenti porterebbe all'esaurimento della stessa, con comportamenti imprevedibili da parte della centralina. Indicativamente è possibile arrivare a 55 utenti.

è corretta la centralina attiverà l'uscita, altrimenti visualizzerà un messaggio di password errata e lo stato dell'uscita non verrà modificato. Se si attiva l'uscita, ripremendo il numero viene nuovamente richiesta la password per disattivare l'uscita stessa. La modalità "c" è temporizzata; l'amministratore deve solo impostare la modalità e la procedura di programmazione è identica a quella appena vista per la modalità "b" (ma senza pre-inserimento password). Tornando in modalità operativa e premendo il tasto corrispondente all'uscita, la centralina chiederà di inserire per quanto tempo dovrà tenere

attiva l'uscita (**Fig. 9c**). L'inserimento si effettua premendo i tasti numerici ed inserendo un tempo nel formato HH:MM in cui HH sono le ore (max. 99) e MM sono i minuti (max. 59) di attivazione dell'uscita. I tasti "*" e "#" permettono di spostare il cursore rispettivamente a sinistra ed a destra, la conferma si dà con il tasto "D"; subito dopo la centralina attiva l'uscita per tutto il tempo impostato, e al termine di esso la disattiva. Se si seleziona nuovamente un'uscita in modalità "c", mentre è in fase attiva, la centralina informa di questo status e chiede se si vuole resettarla per reimpo-

stare il timer (**Fig. 9d**), cosa utile in caso di errore d'impostazione. Premendo "*" si conferma il reset e bisogna reinserire la durata dell'attivazione; premendo "#", oppure attendendo qualche secondo senza effettuare alcuna selezione, si torna alla visualizzazione principale ed il valore precedente resta immutato. La modalità "d" è praticamente l'unione delle due precedenti: l'amministratore deve prevedere una password per l'attivazione dell'uscita e l'utente, subito dopo aver inserito tale password deve impostare la durata dell'attivazione. La modalità "*" indica un'uscita che si attiva automa-

ticamente ad un dato orario e si disattiva automaticamente ad un orario posteriore; il range viene preventivamente impostato dall'amministratore (Fig. 9e), ovviamente i valori accettati sono quelli delle 24 ore, quindi da 00:00 a 23:59. Quindi le uscite impostate con questa modalità non sono comandabili dagli utenti ma operano in modo assolutamente automatico. Premendo il numero corrispondente ad un'uscita di tipo "*" il display si limita a mostrare il range di funzionamento dell'uscita stessa (Fig. 9f).

La modalità "#" è una variante della modalità precedente, anche in questo caso l'amministratore imposta un range orario di funzionamento, che però non sarà automatico; l'uscita di tipo "#" deve essere attivata, e a ciò è abilitato solo un ristretto gruppo di utenti (al massimo 16, valore aumentabile modificando il firmware) che dispongono di una password personale univoca assegnata loro sempre dall'amministratore.

Quando l'uscita "#" è disattivata, la pressione del corrispondente tasto permette di vedere sul display la fascia oraria in cui si può utilizzare l'uscita (Fig. 9g) mentre quando essa è attivabile il simbolo "#" lampeggia.

Il menu 1 prosegue con la voce "1.2.Reset uscite" che permette di resettare tutte le uscite, por-

tando a riposo quelle attivate e riportandole tutte alla modalità iniziale "a" (Fig. 9h).

Una volta impostata almeno un'uscita tipo "#" l'amministratore deve attivare una o più password utente da distribuire a coloro i quali sono abilitati all'uso dell'uscita. A tal fine, sempre dal menu amministratore bisogna selezionare la voce "2. Set utenti" e poi "2.1. Crea utente"; verrà a questo punto chiesto di inserire una password, che potrà essere lunga da 1 a 6 caratteri e contenere una qualunque combinazione di tutti i 16 caratteri del tastierino (Fig. 10a). Subito dopo bisognerà inserire l'uscita "#" da associare (Fig. 10b) e quindi la fascia oraria di abilitazione dell'utente (Fig. 9e). Terminato l'inserimento, la password e la fascia oraria saranno salvate in EEPROM e potranno essere utilizzate per attivare o disattivare l'uscita di pertinenza in modalità "#", purché nella fascia oraria associata. Si possono creare fino a sedici diverse password utente; nella fase di inserimento il display indica il numero associato all'Utente (1÷16).

Per modificare una password Utente bisogna selezionare la voce "2.2 Mod. utente" e, nella successiva videata, l'utente da modificare; nella prima riga è visualizzato l'utente con la relativa password e nella seconda c'è il menu di navigazione, che accetta

i seguenti comandi: tasto "*" per passare all'utente precedente, tasto "#" per quello seguente, tasto "0" per selezionare l'utente visualizzato e tasto "D" per uscire senza selezionare niente (Fig. 10c).

Una volta selezionato un utente, è possibile modificare i suoi dati (password, uscita associata e range orario, con la stessa modalità della creazione), ma prima occorre confermare con "*", mentre "D" permette ancora di uscire senza fare modifiche (Fig. 10d).

Per cancellare gli utenti esistono due voci del menu 2: "2.3.Canc. utente" e "2.4.Canc. tutti".

La prima voce permette di selezionare un utente e cancellare la password (e, ovviamente, il range orario ad essa associato) premendo "0" (Fig. 10c), operazione seguita dalla richiesta di conferma (Fig. 10e) e dal messaggio di cancellazione eseguita; in alternativa il tasto "D" permette di uscire senza eliminare nulla. La seconda voce chiede, appunto, la conferma dell'eliminazione di tutti gli utenti in memoria (Fig. 10f) e successivamente mostra il messaggio di avvenuta cancellazione dei dati corrispondenti.

LE FUNZIONI DI AMMINISTRATORE

I menu 3, 4 e 5 contengono comandi specifici per l'amministrazione della centralina. Come si è visto nello schema elettrico, a bordo dell'unità Master è presente un sistema di orologio quarzato (RTC) che si occupa di mantenere sempre attivo l'orologio/datario, informazione indispensabile per il corretto funzionamento della Centralina. Le voci "3.1.Imposta ora" e "3.2.Imposta data" permettono, appunto, di assegnare all'RTC i valori esatti (Fig. 11a); sarà

a	Utente n.: 2 PSWD:	U01: A1----	d
		*=Modif - D=NO	
b	Utente n.: 2 Uscita:	U01: A1----	e
		*=Elimin - D=NO	
c	U01: A1----	Eliminare tutti	f
	<* #> 0=OK D=NO	Utenti? *=S-D=N	

Fig. 10 - Messaggi della fase di programmazione e uso riguardanti Set utenti.

Fig. 11 - Messaggi della fase di programmazione e uso relativi al sottomenu 3.

Imposta l'ora: 00:00:00	RESET TOTALE? * = Si # = No	d
Imposta la data 01/01/14	Vers. firmware: M 4.10 S 2.30	e
Ripr.dopoRiavvio * = Si # = No	SRAM libera: 1358 bytes	c
Riavvia sistema per conferma		

compito della batteria tampone mantenerli costantemente in memoria.

Il menu 4 riguarda le seguenti funzioni:

- **4.1.Cambia pswd;** permette di cambiare la password amministratore;
- **4.2.Avv. c/ripr.;** permette di mantenere in memoria lo stato delle uscite in caso di black-out; impostando la voce su "SI" (Fig. 11b), se viene meno l'alimentazione, al ritorno le uscite riprendono il loro stato, in base alla programmazione attuale; in caso contrario, al ritorno tutte le uscite saranno disattivate;
- **4.3.Riavvia;** esegue un riavvio sia dell'unità Master che della Slave; viene richiesta conferma prima di eseguire l'operazione (Fig. 11c);
- **4.4.RESET;** esegue un reset hardware, che equivale alla cancellazione totale di tutti i dati memorizzati ed al ripristino dei valori predefiniti (ripорта la centralina alle condizioni originali); viene eseguita la formattazione della memoria, e successivamente viene riavviata la centralina (Fig. 11d).

Il menu 5 fornisce esclusivamente informazioni sulla versione del firmware e sulla quantità di SRAM libera; esso

è utile nel caso si volessero implementare nuove funzionalità (Fig. 11e). Ricordiamo che il firmware è scaricabile in formato sorgente, molto ben documentato, per cui aggiungere funzionalità alla centralina diventa facile; va tuttavia detto che già quelle previste coprono un discreto numero di casi presentabili nella vita quotidiana.

INSTALLAZIONE DELLA CENTRALINA

La centralina Master e la Slave/relé possono essere interconnesse secondo due modalità: via cavo (solo 4 fili) mediante protocollo RS485 oppure wireless. Il protocollo RS485 permette, nelle condizioni ottimali, di coprire fino a 1.000 m di distanza, il che significa che è possibile installare la centralina (Master + keypad + display LCD-I²C) in casa o in un ufficio e la scheda relé in prossimità degli attuatori; questo metodo permette un notevole risparmio in termini di cablaggio, perché il collegamento avviene mediante fili di piccolo diametro.

Se il sistema deve operare in spazi ridotti (ad esempio in un'unica stanza o un appartamento di piccole dimensioni) oppure se gli attuatori sono sparpagliati, si può anche prevedere il montaggio dei tre PCB

nello stesso contenitore, collegando Master e Slave con uno spezzone di cavetto a quattro poli.

Per quanto riguarda l'opzione wireless, per il momento ci siamo limitati a prevedere il collegamento hardware mediante un connettore sulle due schede, al quale si può collegare un qualsiasi sistema di trasmissione radio del tipo RTX capaci di supportare il data-rate previsto.

Il collegamento wireless permette di raggiungere distanze anche superiori ai 1.000 metri dell'RS485, ma bisogna valutare anche i costi e la sicurezza; infatti molti sistemi di trasmissione non prevedono sistemi antidisturbo, per cui i dati in viaggio sono continuamente artefatti diventando quindi inaffidabili.

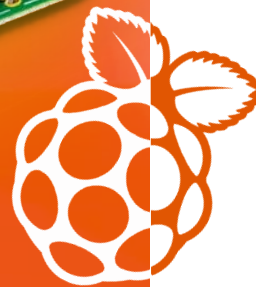
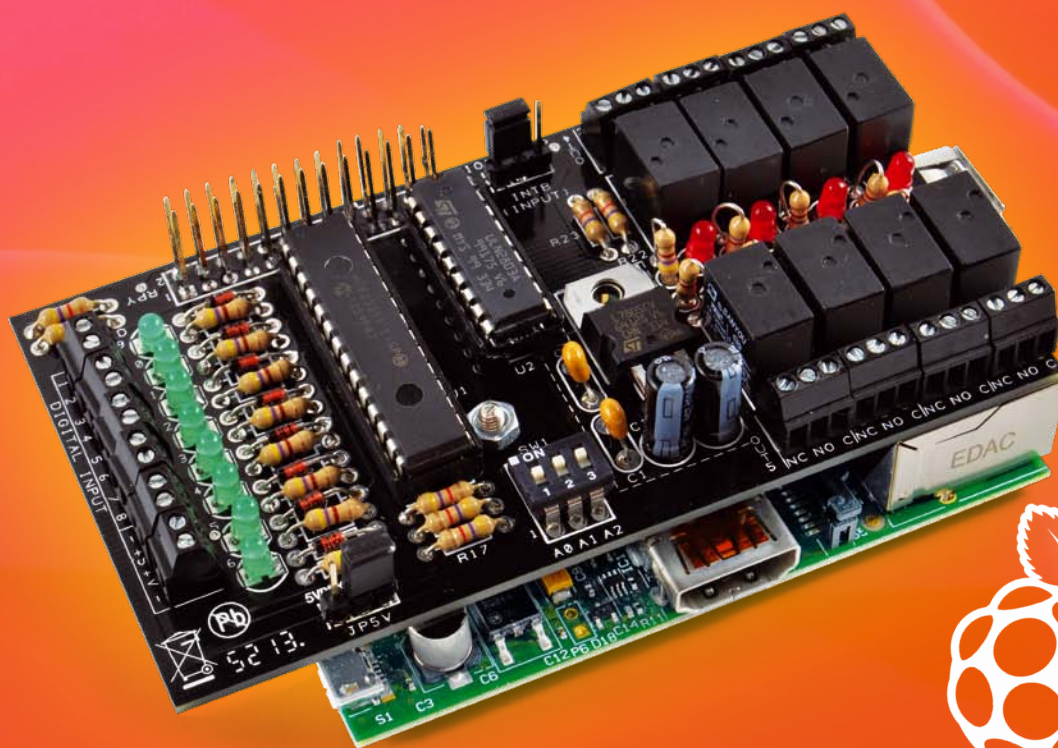
Bisogna inoltre aver chiara una cosa: non è sufficiente collegare due schede RTX ai connettori RF per mettere in collegamento Master e Slave; bisogna anche implementare il relativo protocollo di comunicazione, che dovrà operare al posto di quello previsto per l'RS485. L'implementazione è possibile in quanto forniamo il sorgente dei due firmware per Master e Slave. ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. I master dei circuiti stampati ed il firmware utilizzato nei microcontrollori (eseguibile più sorgente) sono scaricabili gratuitamente dal sito della rivista (www.elettronica.in.it).

Comandiamo fino a 64 uscite a relé e leggiamo un massimo di 64 ingressi digitali impegnando due sole linee di Raspberry Pi: quelle del bus I²C.



bile lo stato. Per accedere agli ingressi/uscite dello shield si utilizza il bus PC. L'indirizzo dello shield è configurabile mediante un DIP Switch a tre posizioni, così da permettere il collegamento e la gestione di un massimo di otto shield contemporaneamente, ciascuno con un indirizzo diverso; ciò consente di espandere il sistema fino a un totale di 64 ingressi e



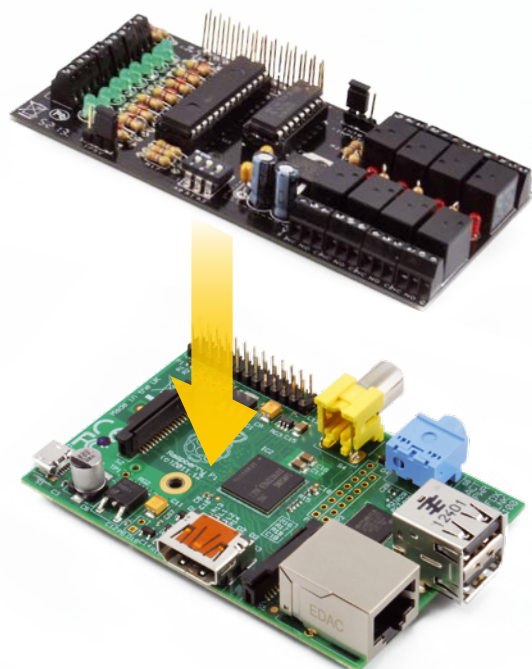
64 uscite pilotabili singolarmente. In questo articolo vi presentiamo anche la libreria Wiring-PI2, una libreria Open Source in licenza GNU LGPLv3 utilizzabile in Python, che permette di accedere ai singoli ingressi e pilotarli in modo decisamente semplice. Ricordiamo che lo shield di espansione presentato in questo articolo convive tranquillamente con gli altri shield che abbiamo presentato ed anche con alcuni di quelli di terze parti. Una ricchezza di possibilità di connessione che lascia spazio alla realizzazione di applicazioni in grado di gestire ambienti anche notevolmente complessi. La realizzazione di applicazioni di questo tipo richiede un salto di livello rispetto ai progetti che abbiamo presentato sino ad ora. Principalmente è necessario progettare e realizzare architetture software in grado di disaccoppiare l'utilizzo fisico delle periferiche esterne rispetto ai moduli di elaborazione veri e propri. In particolare, bisogna realizzare programmi in grado di reagire agli "eventi" esterni gene-

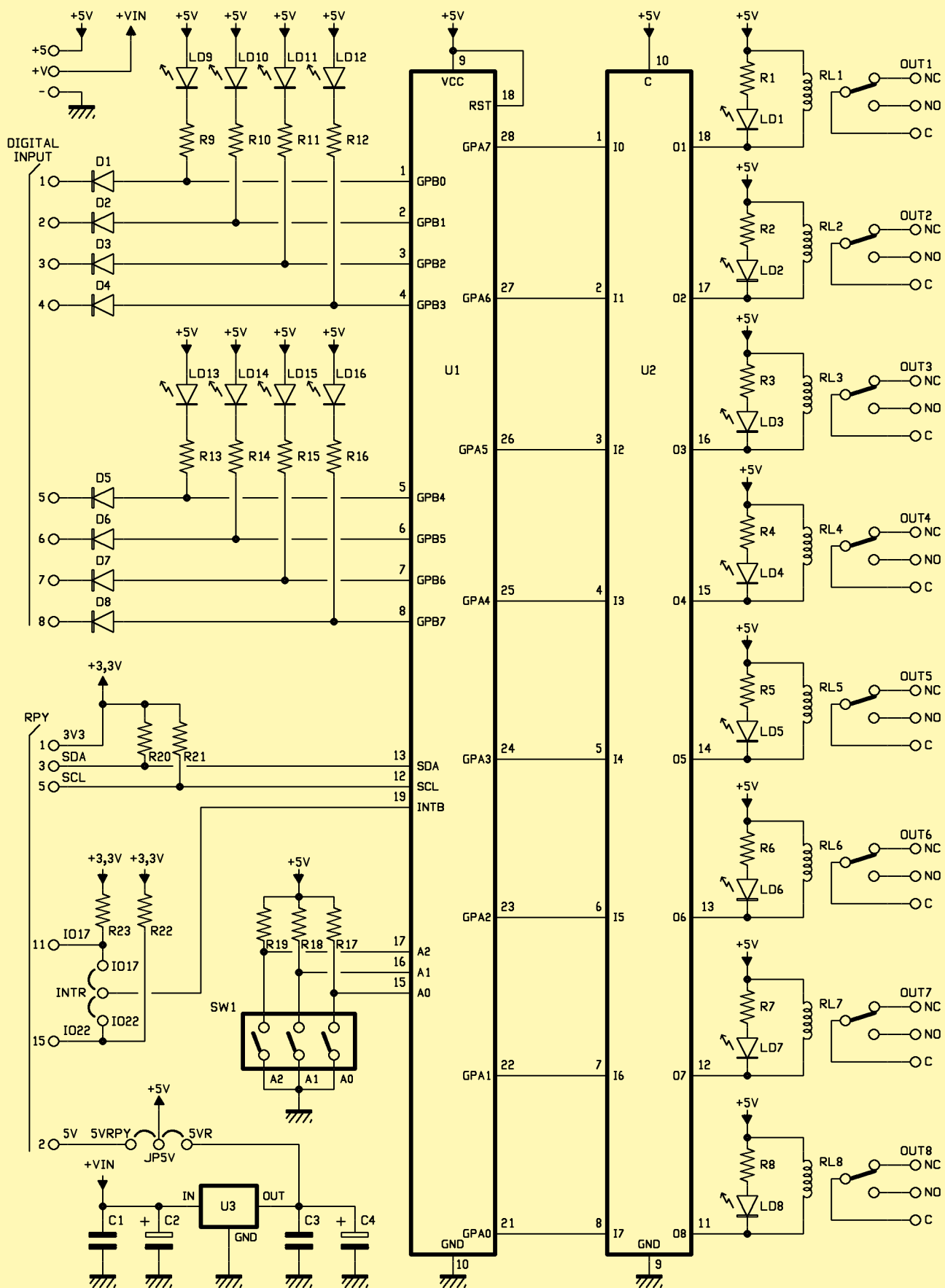
rati dagli ingressi e dai sensori ad essi collegati che, generalmente, si presentano in modo asincrono. È anche necessario progettare correttamente i programmi che gestiscono le periferiche che insistono sulla stessa risorsa di comunicazione. Per esempio, diverse applicazioni possono voler accedere a periferiche differenti, ma che utilizzano lo stesso bus I²C: è proprio il caso dello shield di espansione che presentiamo in questo articolo. Se un'applicazione tiene già impegnato il bus, per leggere o scrivere lo stato di un I/O di sua "competenza", le altre applicazioni che richiedono di impegnare lo stesso bus per gestire gli I/O di loro competenza, trovandolo occupato (busy), non potranno fare altro che andare in errore. La prima soluzione che potrebbe venire in mente è di realizzare un'unica enorme applicazione che gestisca tutte le funzionalità in modo monolitico, un po' come in un enorme microcontrollore. È intuitivo che questo modo di procedere comporta più aspetti negativi che positivi: rigidità dell'applicazione, difficoltà di progettazione, realizzazione, manutenzione e aggiornamento, difficoltà di temporizzazione tra le diverse parti, fragilità (se una parte, anche insignificante, va in errore, cade tutta l'applicazione). Per risolvere un'esigenza applicativa di questo tipo bisogna rivolgersi ad architetture in grado di supportare applicazioni concorrenti e collaboranti e di gestire le comunicazioni tra i diversi attori, basate su server che orchestrano l'utilizzo di risorse da condividere e dove le diverse funzionalità applicative devono essere realizzate come client che richiedono l'utilizzo delle risorse "centralizzate" mediante richieste basate su protocolli condivisi. Vi sono molte alternative possibi-

li: socket TCP/IP, funzionalità database, semplici file e così via. L'importante è che i meccanismi di comunicazione siano in grado di gestire correttamente il fenomeno, per esempio generando code o gestendo "semafori" e "blocchi" per coordinare correttamente le richieste dei client. I client, infine, quando hanno bisogno di accedere ad una delle risorse gestite centralmente, non vi devono accedere direttamente ma devono utilizzare il canale di comunicazione predefinito per quella risorsa, con il protocollo e le modalità previste. In sintesi, nel mondo embedded la progettazione non deve essere rivolta alla singola applicazione ma deve tenere conto dell'architettura che si vuole realizzare, nella sua totalità. Meglio o peggio rispetto ai microcontrollori? Questa domanda non ha una risposta definitiva. Dipende da quello che si vuole realizzare e dai requisiti associati. Sicuramente, in futuro, ci troveremo sempre più nella necessità di integrare e far collaborare questi due ambienti, insieme ad altri come FPGA e PLC. Dai prossimi numeri della rivista inizieremo ad affrontare questo tipo di problematiche, forse più attinenti alla disciplina "informatica" che a quella elettronica, ma ormai, come abbiamo sostenuto molte volte, dobbiamo fare i conti con la necessità, per chi realizza applicazioni basate su dispositivi elettronici destinati al mondo "embedded", di allargare le proprie competenze anche al mondo dell'informatica e della gestione delle reti di comunicazione.

SCHEMA ELETTRICO

Il circuito è progettato attorno all'integrato MCP23017, di Microchip, che offre 16 ingressi/uscite digitali pilotabili mediante il bus I²C. In questo modo, pos-



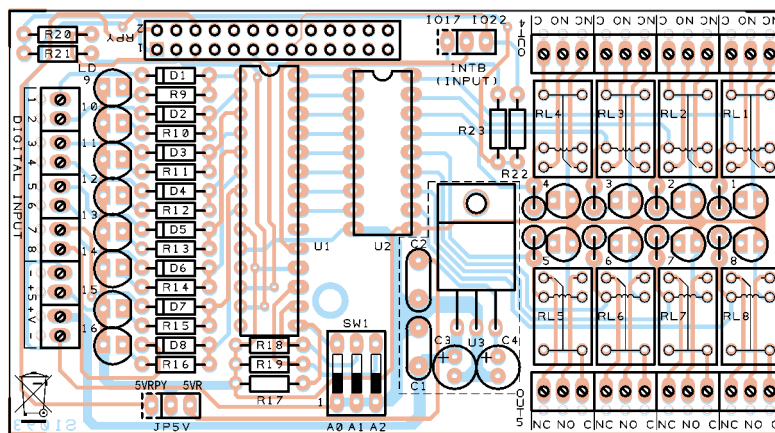
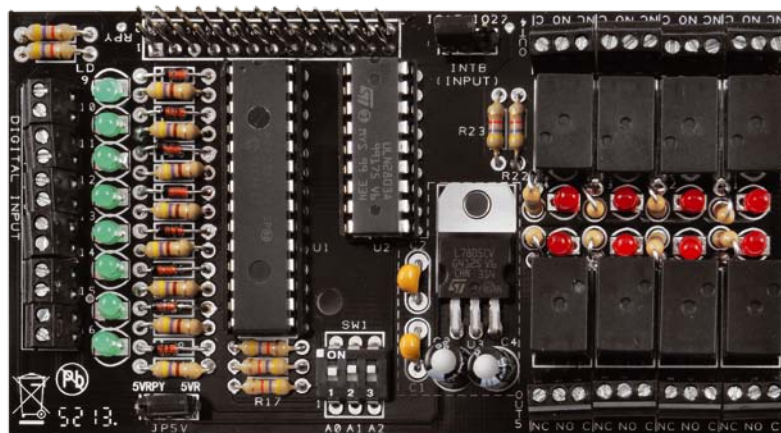


Elenco Componenti:

R1÷R16: 470 ohm
 R17÷R23: 4,7 kohm
 C1, C2: 100 nF multistrato
 C3, C4: 100 µF 16 VL elettrolitico
 D1÷D8: 1N4148
 LD1÷LD8: LED 3 mm rosso
 LD9÷LD16: LED 3 mm verde
 SW1: Dip-Switch 3 vie
 RL1÷RL8: Relé miniatura 5V
 U1: MCP23017-E/SP
 U2: ULN2803
 U3: 7805

Varie:

- Zoccolo 14+14
- Zoccolo 9+9
- Strip maschio 3 poli (2 pz.)
- Jumper (2 pz.)
- Connettore Femmina 2x13 cod. CTF/13+13
- Morsetto 2 poli passo 2,54 mm (6 pz.)
- Morsetto 3 poli passo 2,54 mm (8 pz.)
- Torretta M/F 18 mm
- Dado 3 MA
- Vite 8 mm 3 MA
- Circuito stampato



siamo pilotare gli otto ingressi e le otto uscite digitali del nostro shield impegnando i due soli pin del connettore di Raspberry Pi che fanno capo al bus I²C. Ovviamente il bus non viene utilizzato in modo esclusivo dal nostro shield, ma vi si possono connettere altre periferiche, purché configurate con indirizzi diversi. L'integrato MCP23017 (U1 nello schema elettrico), che abbiamo descritto approfonditamente nel numero 178 della rivista, mette a disposizione 16 linee di I/O suddivise in due banchi da otto, denominati GPA e GPB. Ciascuna linea dell'integrato è configurabile singolarmente come ingresso o come uscita. Nel nostro shield, il banco GPA è tutto

dedicato alla gestione delle uscite con i relé mentre il banco GPB è dedicato alla gestione degli ingressi digitali. Le linee di uscita digitale, provenienti dal banco GPA dell'integrato MCP23017, sono collegate ai pin di ingresso dell'integrato U2, un ULN2803, appositamente progettato per pilotare carichi induttivi, come le bobine dei relé.

All'interno dell'integrato, il cui schema è visibile in Fig. 1, sono presenti otto stadi a Darlington collegati con gli emettitori in comune (terminanti sul pin 9) e i collettori alle uscite, ciascuna delle quali può erogare fino a 500mA. Per inciso, anche se non è il nostro caso, per pilotare carichi che richiedono correnti

superiori ai 500mA, possono essere collegati in parallelo più stadi dell'integrato. All'interno del chip sono presenti anche i diodi di protezione, necessari per pilotare carichi induttivi. Applicazioni tipiche sono il controllo di motori, l'azionamento di relé, solenoidi, lampade, ecc. Gli ingressi 1-8, caratterizzati da un'alta impedenza di ingresso, possono essere pilotati direttamente dalle porte di I/O di un microcontrollore o di un integrato come l'MCP23017. Ma torniamo al nostro schema elettrico. Analizziamo una sola linea di uscita delle otto disponibili in quanto lo schema delle altre sette si ripete in modo identico. Quando, ad esempio, il pin GPA7 si trova

a livello logico alto (oltre i 2,5V fino a 5V), la base del transistor Darlington collegata all'ingresso 1 dell'integrato ULN2803, è polarizzata mediante il partitore resistivo RA, RB ed RC, visibile nello schema di **Fig. 1**. In questa condizione il transistor va in saturazione ed il suo collettore si porta ad un potenziale di poche centinaia di millivolt, lasciando fluire corrente nella bobina del relé. Lo stato di conduzione del transistor pilota anche l'accensione del LED LD1, polarizzato per mezzo della resistenza R1. Quando il pin GPA7 si trova a livello logico basso, il Darlington corrispondente si trova in stato di interdizione, e quindi il suo collettore, come un interruttore aperto, non permette passaggio di corrente nella bobina del relé. Anche il LED, non essendo alimentato, non si accende. Il diodo DA serve a sopprimere le extratensioni inverse generate dalla bobina del relé quando passa dallo stato eccitato allo stato di riposo, tensioni che potrebbero danneggiare il transistor. Sui morsetti a vite della scheda sono disponibili tutti i terminali del relé: il terminale centrale C, il terminale "normalmente chiuso" NC ed il "normalmente aperto" NO. La connessione ai terminali sui morsetti è visibile nel piano di montaggio. Come già anticipato, gli ingressi digitali fanno capo ai pin del banco GPB dell'integrato MCP23017. Anche in questo caso la descrizione di un modulo vale per tutti gli altri. Prendiamo l'ingresso digitale 1, che fa capo al al pin GPB0. La resistenza di pull-up R9, a riposo, mantiene l'ingresso a livello alto. Infatti è collegata al positivo della tensione di alimentazione tramite il LED LD9 che, in questo stato, rimane spento. Il diodo D1 serve a proteggere

l'ingresso dell'integrato, limitando a 5V la tensione massima che può giungere al pin d'ingresso. Infatti, applicando una tensione superiore, D1 non conduce più ed il pin GPB0 "vede" solamente i 5V dell'alimentazione, per effetto della presenza di R9/LD9. Questo accorgimento consente di fare funzionare il circuito con livelli logici, in ingresso, di 12 e più volt, salvaguardando sempre gli ingressi. Tutti gli ingressi digitali hanno la massa in comune, che corrisponde al negativo di alimentazione; negativo e positivo sono presenti anche su due contatti della morsettiera, in modo da avere facilmente a disposizione i 5V con i quali fornire alimentazione ai circuiti esterni. Lo shield richiede un'alimentazione esterna con una tensione compresa tra 9 e 12V, separata rispetto all'alimentazione di Raspberry Pi, in quanto l'uscita a 5V di quest'ultima non è in grado di fornire la corrente necessaria all'alimentazione dello shield. Il circuito integrato U2 (LM7805) provvede ad abbassare e a stabilizzare la tensione d'ingresso, quindi a fornire i 5V necessari alla scheda. Lo shield, invece, è in grado di alimentare Raspberry Pi. Se si desidera mantenere separate le alimentazioni dello shield e della scheda Raspberry Pi, il ponticello JP5V deve essere posizionato tra il pin centrale ed il pin 5VR. Per fare in modo che l'alimentatore dello shield alimenti anche la scheda Raspberry

Pi, il ponticello JP5V deve essere posizionato tra il pin centrale ed il pin 5VRPY.

L'uscita INTB, che permette di gestire l'interrupt proveniente dagli ingressi del banco GPB può essere collegata, mediante il ponticello INTR, al GPIO22 o GPIO17 di Raspberry Pi, in modo da renderla disponibile ad eventuali applicazioni. Il Dip Switch collegato ai pin A0, A1 e A2 dell'integrato, permette di impostare l'indirizzo I²C da assegnare all'integrato. Nel piano di montaggio è possibile vedere la disposizione dei componenti. Unico suggerimento, a parte le usuali attenzioni, è di posizionare il doppio connettore a pettine in modo da distanziare lo stampato dello shield da quello di Raspberry Pi in modo da permetterne il montaggio "a

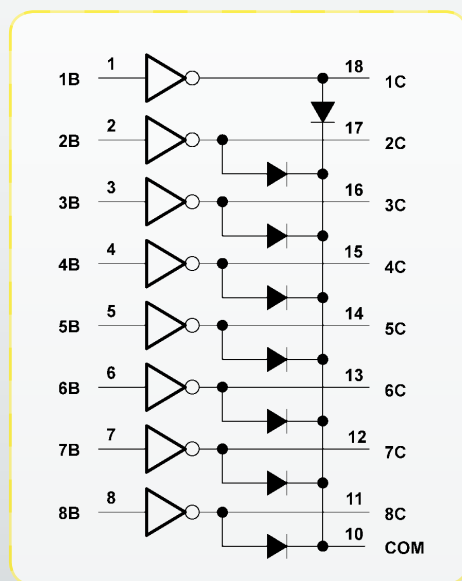
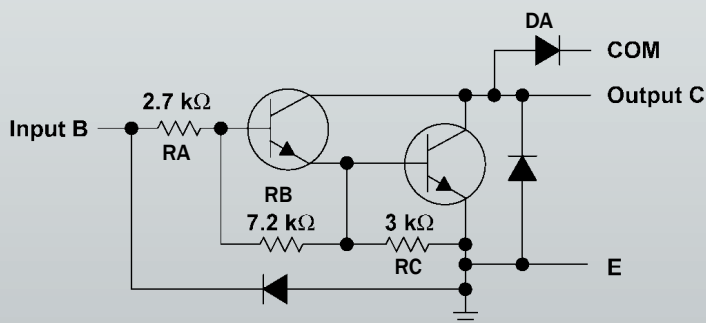
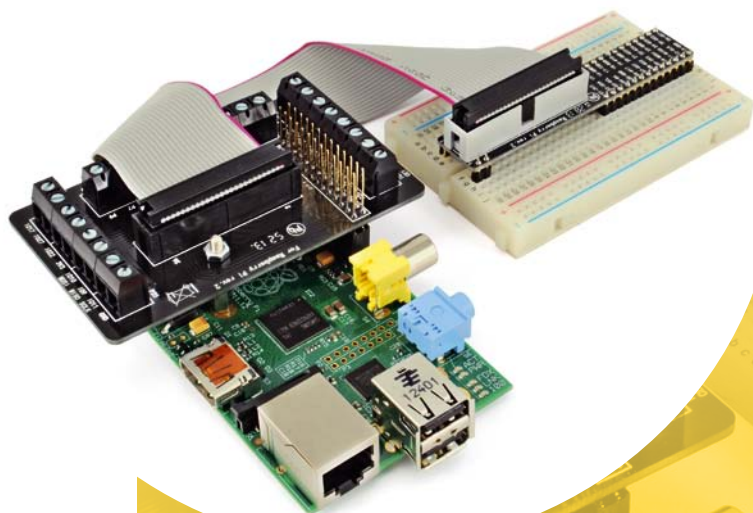


Fig. 1



Adattatore con morsettiera per Raspberry Pi

Lo shield adattatore per Raspberry Pi (FT1104) semplifica la vita a chi desidera sperimentare le possibilità del mondo embedded. È possibile realizzare circuiti o collegare periferiche esterne senza la necessità di realizzare circuiti stampati o di dover utilizzare il saldatore. Lo shield è dotato di morsettiera a 26 poli, pin strip M/F a passo 2,54 mm e connettore maschio a 26 pin, che consentono di portare all'esterno tutte le linee I/O presenti sul connettore a 26 pin della board Raspberry Pi. Il connettore passante a 26 poli dello shield permette di installare altri shield a "catasta" per realizzare complesse applicazioni multifunzione. Utilizzando il cavo piatto incluso nel kit dello shield e l'adattatore FT1072K, (non incluso) è possibile collegare in modo molto semplice Raspberry Pi ad una comune breadboard sulla quale è stata realizzata la propria applicazione. Il vantaggio principale di questa soluzione è la grande flessibilità di utilizzo non disgiunta dalla maggior sicurezza di utilizzo di GPIO, che rimangono protetti da rischi di cortocircuiti o contatti accidentali dei pin tra di loro o con altre parti metalliche di Raspberry Pi. Questi prodotti sono disponibili presso Futura Elettronica (www.futurashop.it).

Tutti gli altri pin di Raspberry Pi sono semplicemente passanti.

UTILIZZO PRATICO DELLO SHIELD

Collegiamo lo shield sulla scheda Raspberry Pi, facendo attenzione alla corrispondenza dei pin dei rispettivi connettori e verificando che la parte inferiore dello shield non vada in contatto con i connettori USB o Ethernet. In caso di dubbio, proteggiamo i connettori stessi con del nastro isolante. Colleghiamo un alimentatore da 9 Vcc ai morsetti di alimentazione dello shield (+V e -) e alimentiamo la scheda Raspberry Pi. Per comunicare con l'integrato MCP23017 è necessario utilizzare il bus I²C e di conseguenza dobbiamo attivare il modulo di gestione del bus stesso che, come ormai saprete, se ci seguite sulla rivista, nell'installazione predefinita di Raspbian, è disabilitato. Per chi ci segue costantemente, questa operazione non costituisce un problema; per chi è nuovo all'ambiente Raspbian, il sistema operativo GNU/Linux di Raspberry Pi, rimandiamo alle indicazioni riportate nel libro "Raspberry Pi il mio primo Linux embedded" (venduto da www.futurashop.it). Nell'articolo descriviamo solo le operazioni specifiche per utilizzare lo shield

pacchetto" nel caso lo si voglia utilizzare congiuntamente ad altri shield, come ulteriori shield di espansione o, per citarne uno, lo shield GSM che abbiamo presentato nel numero 179 della rivista. La possibilità di impilare più schede è data dal fatto che l'integrato MCP23017 è collegato al GPIO di Raspberry Pi unicamente con i pin necessari alla comunicazione I²C ed all'eventuale alimentazione.

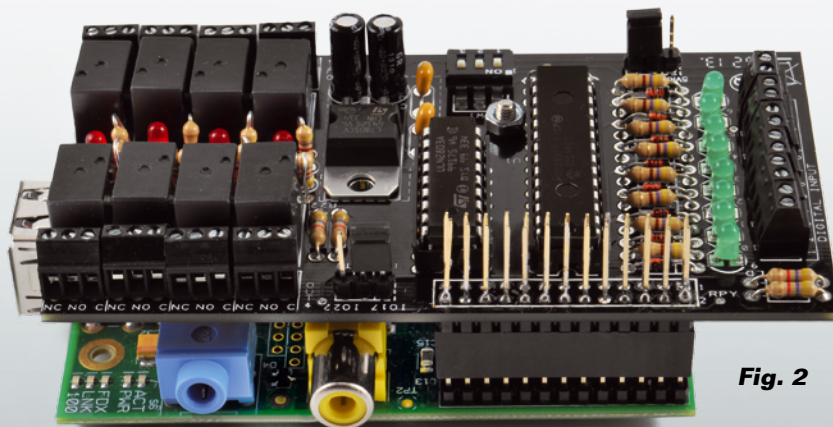


Fig. 2

di I/O. Ricordiamo che per gli articoli sulla rivista dedicati a Raspberry Pi abbiamo adottato il sistema operativo Raspbian (in costante aggiornamento e miglioramento) e gli strumenti di gestione remota, tramite protocollo SSH, utilizzando PuTTY (o KiTTY) e WINScp per la connessione. Dunque, posizioniamo tutti i DIP switch di indirizzamento dello shield su "ON" (GND), il jumper JP5V nella posizione 5VR (alimentazione separata), montiamo lo shield sul connettore di Raspberry Pi, eseguiamo tutti i collegamenti e diamo tensione. Niente fumo? Bene, siamo già a buon punto!

Ora dobbiamo abilitare il driver per la gestione del bus I²C, installare la libreria Python per la gestione dell'integrato MCP23017 e realizzare i primi programmi per accertare che tutto funzioni a dovere. Ricordiamo brevemente il processo per aggiornare il sistema operativo ed abilitare il driver per il bus I²C: innanzitutto utilizziamo PuTTY per aprire una finestra di terminale, ci logghiamo con l'utente "root" e digitiamo i comandi necessari ad aggiornare il database dei pacchetti del sistema operativo:

```
apt-get update
apt-get upgrade
```

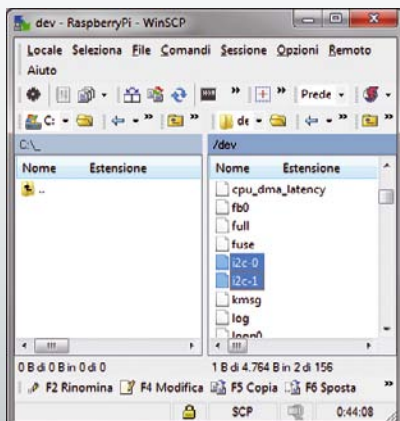


Fig. 7

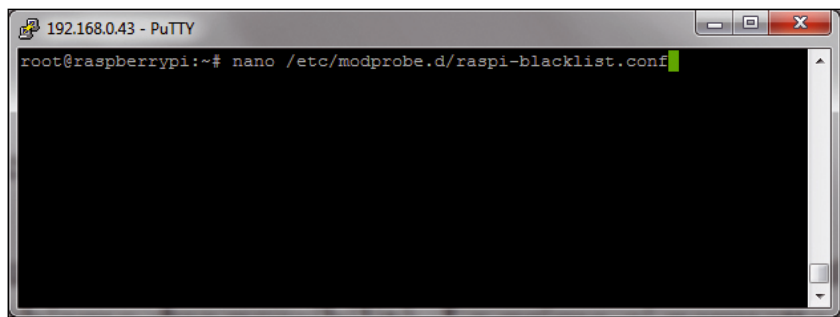


Fig. 3

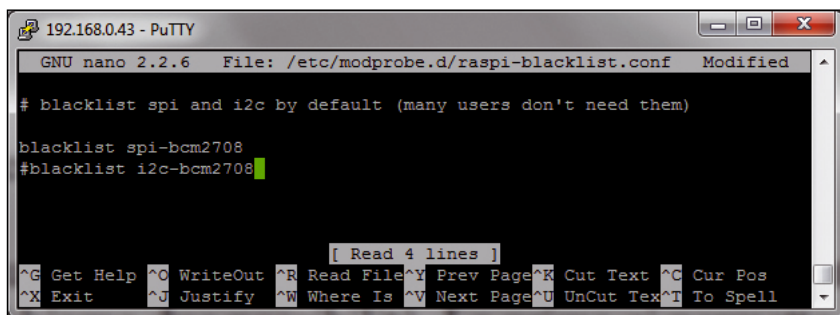


Fig. 4

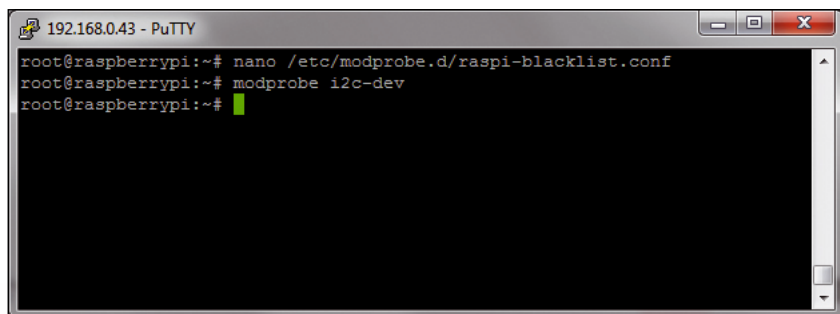


Fig. 5

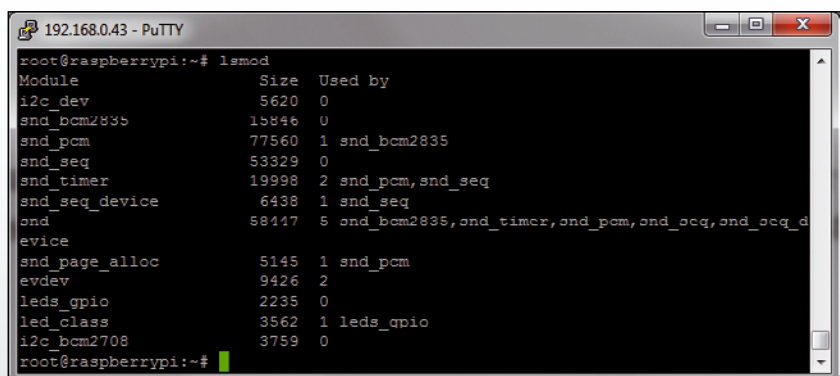


Fig. 6

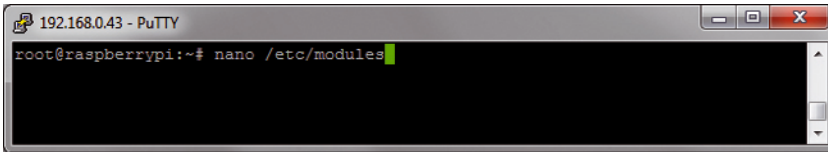
Per ... resuscitare e rendere utilizzabile il modulo di gestione del bus I²C è necessario toglierlo dalla blacklist che lo tiene "prigioniero" e poi "aggiungerlo" all'insieme di moduli conosciuti dal kernel. Apriamo il file di con-

figurazione che contiene l'elenco dei moduli blacklisted (oscurati), con il comando (Fig. 3):

```
nano /etc/modprobe.d/raspi-blacklist.conf
```

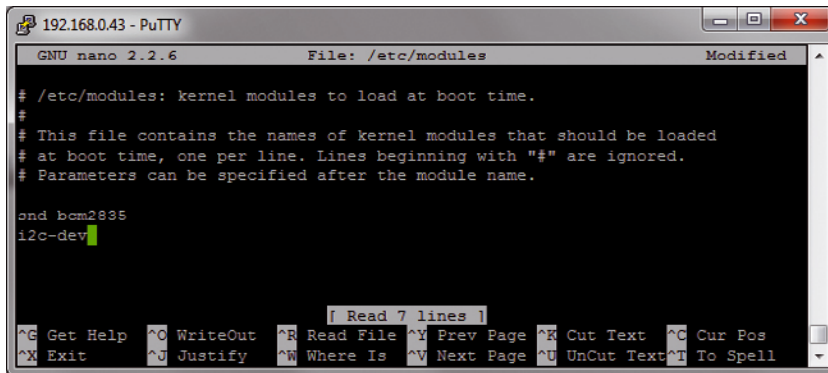
Nano è un editor di testo mini-

Fig. 8



```
192.168.0.43 - PuTTY
root@raspberrypi:~# nano /etc/modules
```

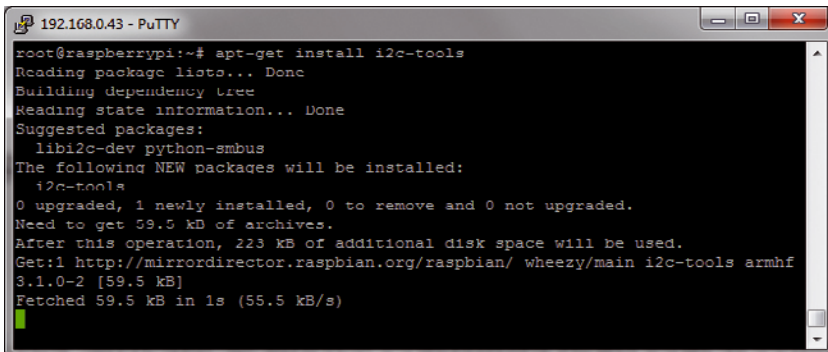
Fig. 9



```
GNU nano 2.2.6 File: /etc/modules Modified
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with '#' are ignored.
# Parameters can be specified after the module name.

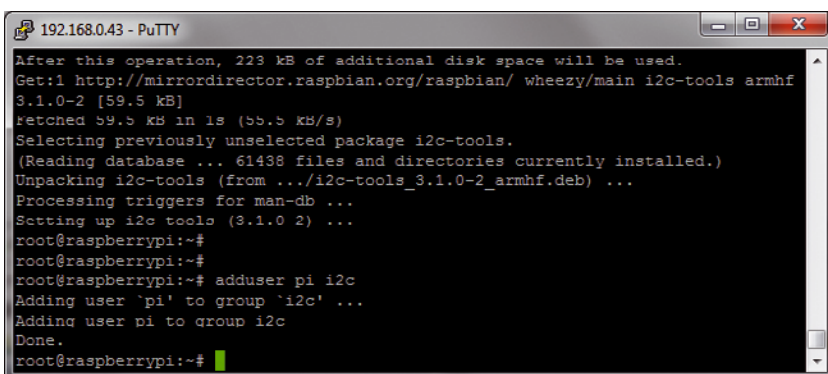
snd_bcm2835
i2c-dev
```

Fig. 10



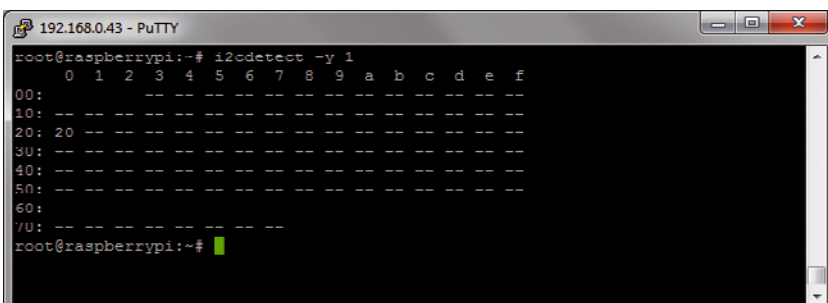
```
192.168.0.43 - PuTTY
root@raspberrypi:~# apt-get install i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  libi2c-dev python-smbus
The following NEW packages will be installed:
  i2c-tools
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 59.5 kB of archives.
After this operation, 223 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main i2c-tools armhf
3.1.0-2 [59.5 kB]
Fetched 59.5 kB in 1s (55.5 kB/s)
```

Fig. 11



```
192.168.0.43 - PuTTY
After this operation, 223 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main i2c-tools armhf
3.1.0-2 [59.5 kB]
Fetched 59.5 kB in 1s (55.5 kB/s)
Selecting previously unselected package i2c-tools.
(Reading database ... 61438 files and directories currently installed.)
Unpacking i2c-tools (from .../i2c-tools_3.1.0-2_armhf.deb) ...
Processing triggers for man-db ...
Setting up i2c-tools (3.1.0-2) ...
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~# adduser pi i2c
Adding user 'pi' to group 'i2c' ...
Adding user pi to group i2c
Done.
root@raspberrypi:~#
```

Fig. 12



```
192.168.0.43 - PuTTY
root@raspberrypi:~# i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: 20 -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
root@raspberrypi:~#
```

male che funziona in ambiente terminale.

Eliminiamo il modulo I²C dalla blacklist cancellando la riga o, come abbiamo preferito noi, commentandola con un “#” (Fig. 4). Premiamo Ctrl-x e poi y alla richiesta di salvare il file, dopo le modifiche.

Eseguiamo un reboot per rendere effettive le modifiche.

Ora dobbiamo fare in modo che il modulo “liberato” venga caricato e diventi parte integrante del kernel. Per questa operazione abbiamo due possibilità: la prima ci permette di caricare il modulo a comando, ed ha validità per tutto il tempo nel quale RaspberryPi rimane acceso. Al boot successivo, il modulo dovrà essere caricato nuovamente da comando.

La seconda possibilità ci permette di caricare il modulo, in automatico, al boot del sistema operativo, e renderlo subito disponibile alle applicazioni, condizione indispensabile in un sistema server unattended.

La prima possibilità richiede l'utilizzo del comando `modprobe`. Digitiamo (Fig. 5):

```
modprobe i2c-dev
```

Possiamo vedere il buon esito dell'attivazione dei driver con il comando che mostra la lista di tutti i moduli installati (Fig. 6):

```
lsmod
```

Dato che in Linux tutto (o quasi) è un file se andiamo nella cartella /dev vediamo apparire i file di collegamento ai device i2c-0 ed i2c-1 (Fig. 7).

Il comando `modprobe` permette di caricare e di scaricare i moduli a run time e mantiene i suoi effetti fintanto che RaspberryPi è acceso. In caso di spegnimento, o anche solo di reboot, i moduli

aggiuntivi dovranno essere ricaricati manualmente. Questa condizione non è adatta a funzionare con un'applicazione stand alone, che deve funzionare in modo automatico. Il comando `modprobe`, con l'opzione `remove`, può essere utilizzato per disattivare un modulo caricato in precedenza.

`modprobe -r i2c-dev`

Se si desidera che il modulo `i2c-dev` venga caricato all'accensione di RaspberryPi, è necessario abilitare il caricamento permanente, che si realizza modificando il file di configurazione `/etc/modules`, che contiene la lista dei driver da caricare al momento del boot. In caso contrario dobbiamo ricordarci di caricare il modulo ad ogni accensione con `modprobe`. Per modificare il file possiamo usare il comando (Fig. 8):

`nano /etc/modules`

ed aggiungere una nuova linea, al file di configurazione, che contenga il comando (Fig. 9):

`i2c-dev`

Premete CTRL-x e poi y per salvare le modifiche al file ed uscire. Ora installiamo il pacchetto `i2c-tools`, che ci fornisce una serie di funzioni utilizzabili a linea di comando per verificare il funzionamento del bus I²C, non prima, come sempre, di avere aggiornato la lista dei pacchetti della distribuzione con il comando:

`apt-get update`

Proseguiamo con il comando (Fig. 10):

`apt-get install i2c-tools`
Aggiungiamo il nostro utente pi

al gruppo I²C (Fig. 11), con:

`adduser pi i2c`

Per attivare le nuove configurazioni, eseguiamo il reboot di Raspberry Pi tramite il comando:

`reboot`

Dopo che RaspberryPi si è riavviata, vi siete riconnessi con PuTTY; impartito, se necessario, il comando `modprobe i2c-dev`, verificate se l'integrato è visibile sul bus I²C, tramite il comando:

`i2cdetect -y 0` per RaspberryPi rev. 1

oppure con il comando:

`i2cdetect -y 1` per RaspberryPi rev. 2

Dovrete ottenere un risultato simile a quello visibile in Fig. 12, dove l'indirizzo 0x20 identifica l'integrato MCP23017.

Ora è tempo di installare la libreria per la gestione dell'integrato MCP23017. Abbiamo optato per la libreria `wiringpi2`, un eccellente sistema di gestione del GPIO di Raspberry Pi scritta in C da Gordon "Drogon" Henderson e "adattata" all'utilizzo con il linguaggio Python da Phil "Gadgetoid" Howard. Questa libreria, distribuita con licenza GNU LGPLv3, include diversi moduli, finalizzati all'utilizzo di specifici integrati che utilizzano i bus I²C e SPI: uno di questi è l'MCP23017, utilizzato nel nostro shield.

```

root@raspberrypi:~# python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import wiringpi2
>>> wiringpi2.piBoardRev()
2
>>> exit()
root@raspberrypi:~#

```

Fig. 13

L'approccio adottato nella libreria per l'utilizzo con l'MCP23017 è di assegnare un riferimento numerico a ciascun pin di ingresso/uscita dell'integrato. La numerazione parte da un numero definito a piacere, rendendo possibile utilizzare più di un integrato su una stessa scheda, ciascuno con i pin riconoscibili da un diverso intervallo di numeri. Come vedremo tra poco, noi abbiamo assegnato un intervallo di numeri ai pin dell'integrato che prendono i valori da 65, per il pin GPA0, a 80 per il pin GPB7.

Prima di descrivere qualche esempio di utilizzo della libreria, procediamo ad installarla. La libreria `wiringpi2` è disponibile come "package" aggiuntivo al linguaggio Python. Per gestire i package di Python abbiamo bisogno del gestore di pacchetti Python "pip". Pip è un acronimo ricorsivo che sta per "Pip Installs Python". Quindi in sequenza installiamo il pacchetto "pip" e poi, tramite questo, la libreria "wiringpi2". Aggiorniamo di nuovo la lista dei pacchetti della distribuzione con il comando:

`apt-get update`

Installiamo le dipendenze richieste dal pacchetto pip, con:

`apt-get install python-dev`

ed infine il pacchetto pip stesso:
`apt-get install python-pip`

Listato 1

```
#!/usr/bin/python

# rele_test.py

import wiringpi2 as wiringpi # importa libreria wiringpi2
from time import sleep

pin_base = 65      # imposta la numerazione dei pin a partire da 65
i2c_addr = 0x20    # indirizzo ottenuto con i pin A0, A1, A2 a GND

wiringpi.wiringPiSetup()          # inizializza libreria wiringpi2
wiringpi.mcp23017Setup(pin_base,i2c_addr) # imposta pin e indirizzo i2c
wiringpi.pinMode(65, 1)           # imposta GPA0 come output
wiringpi.digitalWrite(65, 0)      # imposta GPA0 a 0 (0V, off)
wiringpi.pinMode(66, 1)           # imposta GPA1 come output
wiringpi.digitalWrite(66, 0)      # imposta GPA1 a 0 (0V, off)
wiringpi.pinMode(67, 1)           # imposta GPA2 come output
wiringpi.digitalWrite(67, 0)      # imposta GPA2 a 0 (0V, off)
wiringpi.pinMode(68, 1)           # imposta GPA3 come output
wiringpi.digitalWrite(68, 0)      # imposta GPA3 a 0 (0V, off)
wiringpi.pinMode(69, 1)           # imposta GPA4 come output
wiringpi.digitalWrite(69, 0)      # imposta GPA4 a 0 (0V, off)
wiringpi.pinMode(70, 1)           # imposta GPA5 come output
wiringpi.digitalWrite(70, 0)      # imposta GPA5 a 0 (0V, off)
wiringpi.pinMode(71, 1)           # imposta GPA6 come output
wiringpi.digitalWrite(71, 0)      # imposta GPA6 a 0 (0V, off)
wiringpi.pinMode(72, 1)           # imposta GPA7 come output
wiringpi.digitalWrite(72, 0)      # imposta GPA7 a 0 (0V, off)

wiringpi.pinMode(73, 0)           # imposta GPB0 come input
wiringpi.pullUpDnControl(73, 0)   # pull-up e pull-down interni disattivati
wiringpi.pinMode(74, 0)           # imposta GPB1 come input
wiringpi.pullUpDnControl(74, 0)   # pull-up e pull-down interni disattivati
wiringpi.pinMode(75, 0)           # imposta GPB2 come input
wiringpi.pullUpDnControl(75, 0)   # pull-up e pull-down interni disattivati
wiringpi.pinMode(76, 0)           # imposta GPB3 come input
wiringpi.pullUpDnControl(76, 0)   # pull-up e pull-down interni disattivati
wiringpi.pinMode(77, 0)           # imposta GPB4 come input
wiringpi.pullUpDnControl(77, 0)   # pull-up e pull-down interni disattivati
wiringpi.pinMode(78, 0)           # imposta GPB5 come input
wiringpi.pullUpDnControl(78, 0)   # pull-up e pull-down interni disattivati
wiringpi.pinMode(79, 0)           # imposta GPB6 come input
wiringpi.pullUpDnControl(79, 0)   # pull-up e pull-down interni disattivati
wiringpi.pinMode(80, 0)           # imposta GPB7 come input
wiringpi.pullUpDnControl(80, 0)   # pull-up e pull-down interni disattivati

try:
    while True:
        if not wiringpi.digitalRead(73): # logica invertita (pull-up esterno)
            wiringpi.digitalWrite(65, 1) # imposta GPA1 a 1 (3V3, on)
        else:
            wiringpi.digitalWrite(65, 0) # imposta GPA1 a 0 (0V, off)

        if not wiringpi.digitalRead(74):
            wiringpi.digitalWrite(66, 1)
        else:
            wiringpi.digitalWrite(66, 0)

        if not wiringpi.digitalRead(75):
            wiringpi.digitalWrite(67, 1)
        else:
            wiringpi.digitalWrite(67, 0)

        if not wiringpi.digitalRead(76):
            wiringpi.digitalWrite(68, 1)
        else:
            wiringpi.digitalWrite(68, 0)

        if not wiringpi.digitalRead(77):
            wiringpi.digitalWrite(69, 1)
        else:
            wiringpi.digitalWrite(69, 0)

        if not wiringpi.digitalRead(78):
            wiringpi.digitalWrite(70, 1)
        else:
```

(Continua)

Ora, utilizzando il gestore di pacchetti pip possiamo installare la libreria vera e propria:

pip install wiringpi2

Terminato questo processo, possiamo verificare immediatamente se tutto ha funzionato a dovere. Apriamo l'interprete Python a linea di comando, digitando semplicemente "python" al prompt della finestra di terminale, e verifichiamo la corretta installazione della libreria wiringpi2 digitando le istruzioni (**Fig. 13**):

```
import wiringpi2
wiringpi2.piBoardRev()
```

La risposta, contenente il numero di versione della libreria, ci conferma che tutto funziona correttamente.

Ora utilizziamo la libreria per realizzare qualche programma di esempio. Il primo programma, visibile nel **Listato 1**, permette di agganciare ciascun pin di ingresso del banco GPB dell'integrato, al corrispondente pin di output del banco GPA. I commenti non dovrebbero lasciare dubbi sulla logica del programma, quindi descriviamo unicamente le istruzioni "try: - except:", introdotte per la prima volta nei nostri articoli. Queste istruzioni permettono di "intrappolare" gli errori. Se includiamo un insieme di istruzioni all'interno di un gruppo "try:" (prova), in caso di errore il programma, invece di interrompere l'esecuzione, intercetta l'errore e innesca le istruzioni contenute all'interno del gruppo "except:". In questo gruppo, nel nostro caso, abbiamo inserito le istruzioni necessarie a riportare tutti i pin dell'integrato configurati come input. Questo meccanismo funziona anche quando si interrompe il loop principale del

Listato 1 – segue

programma con CTRL-c.

Per provare il programma possiamo collegare dei pulsanti tra la massa e ciascun pin di ingresso dello shield; in alternativa possiamo collegare un terminale di un cavetto jumper al morsetto di massa e con l'altro terminale toccare i connettori dei morsetti di ingresso. Con lo strumento WINScp, copiamo il programma del **Listato 1** in una cartella di Raspberry Pi all'interno della cartella `/home`: per esempio, `/home/Rele`. Diamo un nome al programma: per esempio "rele-test.py". Posizioniamoci nella cartella `/home/Rele` e lanciamo il programma con il comando:

```
python rele-test.py
```

Ora, portando a massa (con i pulsanti o con il jumper) gli ingressi dello shield, vedremo accendersi i LED e chiudersi i relé delle corrispondenti uscite, per tutto il tempo durante il quale l'ingresso rimarrà collegato a massa. Per terminare il programma, diamo il comando CTRL-c (**Fig. 14**).

Come abbiamo descritto nella parte introduttiva dell'articolo, questo programma, quando funziona, tiene impegnato costantemente il bus I²C, impedendo ad eventuali altri programmi di accedervi. Per superare tale problema è necessario disaccoppiare l'utilizzo del bus rispetto ai programmi.

Un metodo è quello di scrivere programmi che impegnino il bus unicamente per il tempo necessario ad eseguire le azioni richieste, e poi lo liberino per lasciare il campo ad altri richiedenti. Vi forniamo un esempio di questa soluzione nel **Listato 2** e nel **Listato 3**. La soluzione proposta è composta da un programma, da lanciare una sola volta all'accensione di Raspberry Pi, per inizializzare

```
wiringpi.digitalWrite(70, 0)

if not wiringpi.digitalRead(79):
    wiringpi.digitalWrite(71, 1)
else:
    wiringpi.digitalWrite(71, 0)

if not wiringpi.digitalRead(80):
    wiringpi.digitalWrite(72, 1)
else:
    wiringpi.digitalWrite(72, 0)

sleep(0.05)

except:
    wiringpi.digitalWrite(65, 0) # imposta GPA1 a 0 (0V, off)
    wiringpi.pinMode(65, 0)      # reimposta GPA1 come input
    wiringpi.digitalWrite(66, 0)
    wiringpi.pinMode(66, 0)
    wiringpi.digitalWrite(67, 0)
    wiringpi.pinMode(67, 0)
    wiringpi.digitalWrite(68, 0)
    wiringpi.pinMode(68, 0)
    wiringpi.digitalWrite(69, 0)
    wiringpi.pinMode(69, 0)
    wiringpi.digitalWrite(70, 0)
    wiringpi.pinMode(70, 0)
    wiringpi.digitalWrite(71, 0)
    wiringpi.pinMode(71, 0)
    wiringpi.digitalWrite(72, 0)
    wiringpi.pinMode(72, 0)
    print " "
    print "fine"
```

l'integrato MCP23017 e configurare i pin nel modo richiesto dallo shield. Il secondo programma viene lanciato ogni volta che si desidera modificare lo stato di un pin di uscita. Nel nostro esempio, lo faremo lanciare in momenti temporali definiti utilizzando lo schedulatore cron. In questo modo potremo realizzare un impianto temporizzato come un sistema di irrigazione o un sistema di controllo luci giorno/notte. Il programma del **Listato 2** è derivato dalla parte di configurazione iniziale del programma

mostrato nel **Listato 1**. Anche il programma presentato nel **Listato 3** è derivato da frammenti di codice del **Listato 1** con in più la possibilità di ricevere parametri dall'esterno.

Nel nostro caso ne abbiamo previsti due: il nome del pin del quale vogliamo modificare lo stato, nella forma Ax ed il valore dello stato che desideriamo impostare: "ON" oppure "OFF". Il formato con il quale lanceremo il comando sarà quindi:

```
python rele_pin_set.py A1 ON
```

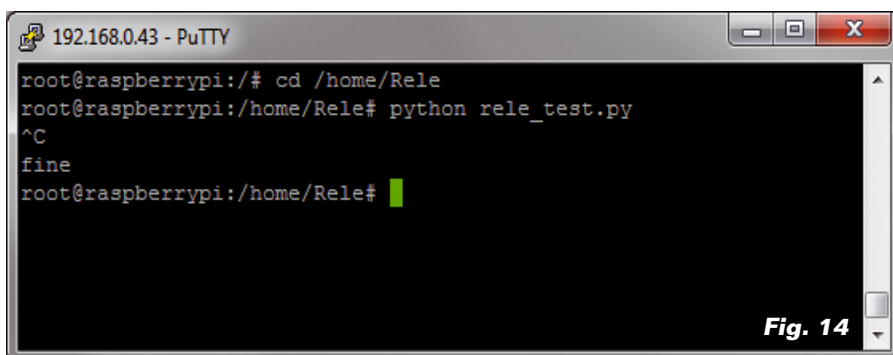


Fig. 14

Listato 2

```
#!/usr/bin/python

# rele_set.py

import wiringpi2 as wiringpi # importa libreria wiringpi2

from time import sleep
import sys

pin_base = 65      # imposta la numerazione dei pin a partire da 65
i2c_addr = 0x20    # indirizzo ottenuto con i pin A0, A1, A2 a GND

try:
    wiringpi.wiringPiSetup()                # inzializza libreria wiringpi2
    wiringpi.mcp23017Setup(pin_base,i2c_addr) # imposta pin e indirizzo i2c
    wiringpi.pinMode(65, 1)                 # imposta GPA0 come output
    wiringpi.digitalWrite(65, 0)            # imposta GPA0 a 0 (0V, off)
    wiringpi.pinMode(66, 1)                 # imposta GPA1 come output
    wiringpi.digitalWrite(66, 0)            # imposta GPA1 a 0 (0V, off)
    wiringpi.pinMode(67, 1)                 # imposta GPA2 come output
    wiringpi.digitalWrite(67, 0)            # imposta GPA2 a 0 (0V, off)
    wiringpi.pinMode(68, 1)                 # imposta GPA3 come output
    wiringpi.digitalWrite(68, 0)            # imposta GPA3 a 0 (0V, off)
    wiringpi.pinMode(69, 1)                 # imposta GPA4 come output
    wiringpi.digitalWrite(69, 0)            # imposta GPA4 a 0 (0V, off)
    wiringpi.pinMode(70, 1)                 # imposta GPA5 come output
    wiringpi.digitalWrite(70, 0)            # imposta GPA5 a 0 (0V, off)
    wiringpi.pinMode(71, 1)                 # imposta GPA6 come output
    wiringpi.digitalWrite(71, 0)            # imposta GPA6 a 0 (0V, off)
    wiringpi.pinMode(72, 1)                 # imposta GPA7 come output
    wiringpi.digitalWrite(72, 0)            # imposta GPA7 a 0 (0V, off)

    wiringpi.pinMode(73, 0)                 # imposta GPB0 come input
    wiringpi.pullUpDnControl(73, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(74, 0)                 # imposta GPB1 come input
    wiringpi.pullUpDnControl(74, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(75, 0)                 # imposta GPB2 come input
    wiringpi.pullUpDnControl(75, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(76, 0)                 # imposta GPB3 come input
    wiringpi.pullUpDnControl(76, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(77, 0)                 # imposta GPB4 come input
    wiringpi.pullUpDnControl(77, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(78, 0)                 # imposta GPB5 come input
    wiringpi.pullUpDnControl(78, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(79, 0)                 # imposta GPB6 come input
    wiringpi.pullUpDnControl(79, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(80, 0)                 # imposta GPB7 come input
    wiringpi.pullUpDnControl(80, 0)          # pull-up e pull-down interni disattivati

except Exception, e:
    print e
```

Per leggere i parametri all'interno del programma si utilizza il metodo "argv" del modulo "sys". In Python, il modulo "sys" rende disponibili una serie di funzioni e di variabili che permettono di interagire con l'ambiente runtime all'interno del quale viene eseguito il programma Python. In particolare, "argv" contiene la lista degli argomenti passati al programma al momento del lancio. I parametri sono resi disponibili in forma di array dove l'elemento argv[0] contiene il nome del programma e i successivi argv[i]

contengono gli argomenti, nell'ordine nel quale sono stati digitati a linea di comando.

Nel nostro caso avremo:

argv[0] = "rele_pin_set"

argv[1] = "A1"

argv[2] = "ON"

I parametri acquisiti vengono utilizzati per personalizzare le istruzioni di configurazione del pin. Dato che abbiamo assegnato al primo pin il valore 65, per ottenere la corrispondenza tra il parametro passato al programma, ed il corrispondente numero gestito dalla libreria, utilizziamo l'istruzione:

```
Pin_num = int(Pin[1:]) + 65
```

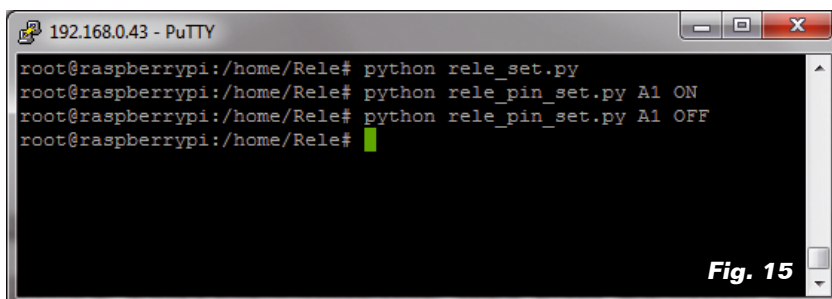
Lo scopo di questa istruzione è di eliminare dal primo parametro la "A" iniziale, e sommare 65 al valore numerico risultante per ottenere il valore corrispondente del pin riconosciuto dalla libreria. Lanciamo manualmente il primo programma per configurare inizialmente l'integrato MCP23017; lanciamo poi il secondo programma, variando i parametri e verifichiamo i risultati che otteniamo (Fig. 15).

Passiamo ora a configurare lo schedatore "cron".

Lo schedatore "cron" è il servizio predefinito, in Linux, per l'esecuzione a tempo di programmi e comandi.

Per funzionare, cron si appoggia a una tabella che contiene le "direttive" sui programmi da eseguire e definisce in quali istanti lanciarli. La tabella utilizzata da cron si gestisce con la funzionalità crontab, la cui sintassi è la seguente:

crontab <opzione>



```
192.168.0.43 - PuTTY
root@raspberrypi:/home/Rele# python rele_set.py
root@raspberrypi:/home/Rele# python rele_pin_set.py A1 ON
root@raspberrypi:/home/Rele# python rele_pin_set.py A1 OFF
root@raspberrypi:/home/Rele#
```

Fig. 15

dove opzione può assumere i valori *-e*, *-l* e *-r*.
Il comando:

crontab -e

ci permette di aprire il file di configurazione predefinito di *crontab*, come si vede in **Fig. 16**. Ogni riga del file di configurazione ha la seguente struttura:

```
***** comando da eseguire
-----
| | | |
| | | |---- giorno della settimana (0-6) (0=Domenica)
| | | +----- mese (1-12)
| | +----- - giorno del mese (1-31)
| +----- - ora (0-23)
+----- - min (0-59)
```

Ciascun parametro può essere configurato come serie di valori (ad esempio 0, 3, 7, 9) come un intervallo (ad esempio 3-7) oppure una combinazione dei due (esempio: 0,3,4-6,9) o, ancora, come frequenza */5 (ogni 5 minuti, ore, ecc.). Nel nostro caso vogliamo che ad ogni minuto pari venga eseguito il programma *rele_pin_set.py*, con i parametri impostati in modo da far chiudere il relé collegato al pin GPBA1 e ad ogni minuto dispari venga rieseguito lo stesso programma con i parametri impostati per far aprire il relé. Nella finestra che ci mostra il contenuto del file, aggiungiamo in fondo le righe di direttiva come in **Fig. 16**:

```
0-58/2 * * * * python /home/Rele/rele_pin_set.py A1 ON > /dev/null
```

che significa far eseguire il programma ad ogni minuto pari di ogni ora, giorno e mese, e:

```
1-59/2 * * * * python /home/Rele/rele_pin_set.py A1 OFF > /dev/null
```

Usciamo dal file con <CTRL>X, rispondiamo "y" alla richiesta di salvataggio e confermiamo il nome del file con "Invio". D'ora in avanti, ogni minuto verrà lanciato il programma che, in

Listato 3

```
#!/usr/bin/python

# rele_pin_set.py

import wiringpi2 as wiringpi
from time import sleep
import sys

Pin = sys.argv[1]
Stato = sys.argv[2]

pin_base = 65      # imposta la numerazione dei pin a partire da 65
i2c_addr = 0x20     # indirizzo ottenuto con i pin A0, A1, A2 a GND

Pin_num = int(Pin[1:]) + 65

try:
    wiringpi.wiringPiSetup()                # inizializza libreria wiringpi
    wiringpi.mcp23017Setup(pin_base,i2c_addr) # imposta pin e indirizzo i2c
    if Stato == "ON":
        wiringpi.digitalWrite(Pin_num, 1) # imposta il pin a 1 (3V3, on)
    else:
        wiringpi.digitalWrite(Pin_num, 0) # simposta il pin a 0 (0V, off)
except Exception, e:
    print e
```

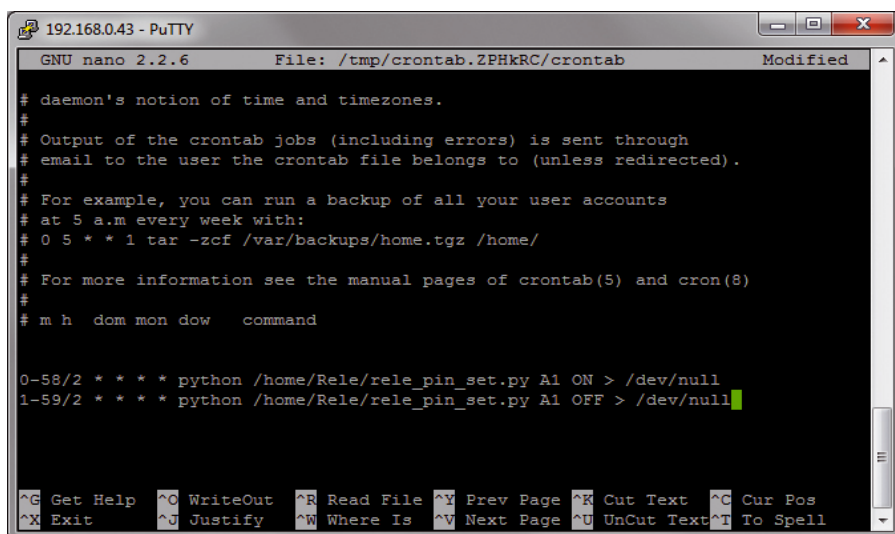


Fig. 16

alternativa, attiverà e disattiverà il relé. Per modificare le impostazioni di cron, richiamiamo la funzionalità:

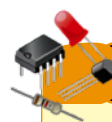
crontab -e

Per vedere le impostazioni utilizziamo l'opzione:

crontab -l

mentre per interrompere l'esecuzione automatica digitiamo

crontab -r



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità e il master del PCB può essere scaricato gratuitamente dal sito della rivista. Il circuito è anche disponibile in kit (cod. FT1093K) al prezzo di 32,00 Euro, 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>

NIDays 2014

di DAVIDE SCULLINO

In marcia verso la quarta rivoluzione industriale: siamo pronti? Il 5 marzo a Roma, National Instruments ci spiegherà come farci trovare preparati.



Quella che sotto molti aspetti viene considerata la quarta rivoluzione industriale sta per iniziare: il mondo interconnesso governerà processi e attività fino ad oggi operanti ognuna localmente e nel proprio mondo si apriranno all'integrazione con il web. Reti di calcolatori controlleranno processi fisici, macchine e sistemi robotici industriali, domestici e interi settori industriali, dall'energia ai trasporti, dalla produzione alla sanità ne potranno beneficiare. Farsi trovare pronti ad affrontare le nuove tecnologie richiede la capacità di sviluppare metodi di progettazione che possano offrire

08:30	Registrazione, welcome coffee e visita all'area espositiva							
10:00	Benvenuto							
10:30	Keynote mattutina - Il dilemma dell'innovazione: soluzioni puntuali o piattaforme programmabili?							
11:00	Premiazione Premio Nicola Chiari							
11:15	Pausa e visita all'area espositiva							
	LabVIEW e Acquisizione Dati	Sistemi Embedded	Test & RF	Academic Summit	LabVIEW Days +	LabVIEWDAQ Zone	LabVIEWRIO Zone	Academic Zone
	Leptis Magna 1-2	Cesarea	Tarragona	Orange 1	Pola	Thugga	Gerasa	Orange 2
11:30	Le novità in LabVIEW 2013	Prognostics and Health Management (PHM), cosa c'è da sapere: metodologie, risultati e trend	Esempi e IP applicative in LabVIEW FPGA per il Vector Signal Transceiver	Come sfruttare al meglio l'Academic Site License nel tuo ateneo (In inglese)	Ottimizza la gestione dei file in LabVIEW (In inglese)	Area Pratica CompactDAQ	Area Pratica CompactRIO	Area Pratica Academic
12:10	Gestire i "Big Data": Cloud, Technical Data Management e Big Analog Data	Scopri i segreti del nuovo modulo LabVIEW Real-Time basato su Linux RT	Uno strumento definito via software: emulazione di protocollo Zigbee su VST	Robotica indossabile e cambiamenti demografici	Suggerimenti per rendere la User Interface un valore aggiunto	Area Pratica CompactDAQ	Area Pratica CompactRIO	Area Pratica Academic
12:50	Lunch a buffet e visita all'area espositiva							
14:00	Keynote pomeridiana - Affrontare le sfide tecnologiche dell'oggi e del domani con la Progettazione Grafica di Sistemi							
	LabVIEW e Acquisizione Dati	Sistemi Embedded	Test & RF	Academic Summit	LabVIEW Days +	LabVIEWDAQ Zone	LabVIEWRIO Zone	Academic Zone
	Leptis Magna 1-2	Cesarea	Tarragona	Orange 1	Pola	Thugga	Gerasa	Orange 2
15:30	Acquisizione dati avanzata: realizzare un sistema di monitoraggio strutturale completo - il progetto dell'Arena di Verona	Comunicare di più in meno tempo: soluzioni di comunicazione cross-piattaforma in LabVIEW	Scegliere gli strumenti software più adatti per la progettazione di sistemi di test e collaudo	NI myRIO: una piattaforma didattica per sviluppare progetti embedded reali in un semestre	Programmazione ad eventi (avanzata) in LabVIEW	Area Pratica CompactDAQ	Area Pratica CompactRIO	Area Pratica Academic
16:10	Dai milliwatt ai megawatt: misure elettriche e di potenza in LabVIEW	Tendenze e nuove tecnologie nell'architettura LabVIEW RIO	Progettare sistemi di test, misura e controllo ad elevate prestazioni	Come trasformare un'idea in prototipo, e lo stesso in prodotto di mercato: il Programma Incubatore per le Imprese Innovative Italiane	Digital Signal Processing per la tua applicazione (In inglese)	Area Pratica CompactDAQ	Area Pratica CompactRIO	Area Pratica Academic
17:00	Conclusione con sorpresa, torta dei 25 anni di NI Italy e visita all'area espositiva							

prestazioni adeguate, integrando sensori, cloud computing e controlli real-time.

La ricetta di National Instruments si basa sulla progettazione grafica di sistemi tramite LabVIEW, l'ambiente di sviluppo integrabile con l'hardware PXI, CompactRIO, FPGA. Come applicarla verrà discusso il 5 marzo prossimo a Roma, a NIDays, l'annuale evento che affianca a keynote e interventi di personaggi chiave di National Instruments, seminari e conferenze tematiche tenuti da personaggi di spicco del mondo accademico e della ricerca, oltre a dimostrazioni pratiche dei prodotti NI.

L'evento si terrà in uno spazio che comprende sale per conferenze, l'area espositiva e le zone allestite per le dimostrazioni pratiche. Ad aprire i lavori, la keynote del mattino il cui tema sarà: "Il dilemma dell'innovazione: soluzioni puntuali o piattaforme programmabili?". La keynote svelerà nuove tecnologie che consentono di semplificare lo sviluppo di soluzioni innovative e spiegherà come la piattaforma NI LabVIEW RIO sia stata creata per rivoluzionare l'insegnamento dell'ingegneria e la progettazione di sistemi

embedded. Testimonierà, inoltre, le più recenti evoluzioni del primo strumento progettato via software e gli ultimi sviluppi nel calcolo eterogeneo e mostrerà la più recente versione di LabVIEW. Alla keynote seguirà l'assegnazione del Premio Nicola Chiari, il prestigioso riconoscimento che dal 2006 ogni anno ricorda la competenza e dedizione di Nicola Chiari (tra i primi in Italia a diffondere e sviluppare LabVIEW) premiando le applicazioni tecniche che maggiormente si distinguono nel mondo dell'industria e della ricerca.

Dopo la premiazione, prenderanno il via le sessioni mattutine e dopo pranzo, si terrà la keynote pomeridiana sul tema: "Affrontare le sfide tecnologiche dell'oggi e del domani con la Progettazione Grafica di Sistemi". Una panoramica sulle soluzioni adottate da tecnici e ingegneri che si affidano alla progettazione grafica di sistemi per affrontare alcune tra le sfide più critiche della nostra società, in settori quali le fonti energetiche di prossima generazione, le telecomunicazioni, i trasporti e la ricerca avanzata. Particolarmente accurata ed arricchita la sezione

delle aree pratiche, che in questa edizione di NIDays conta su tre sale interamente a disposizione del pubblico, che potrà toccare con mano i prodotti National Instruments. I partecipanti potranno scegliere una delle tre zone, equipaggiate con NI LabVIEW e diverse piattaforme hardware National Instruments; le aree sono LabVIEW DAQ Zone (per sperimentare l'acquisizione dati con NI CompactDAQ), LabVIEW RIO Zone (per sperimentare il monitoraggio e il controllo con NI CompactRIO e Academic Zone (per chi desidera provare la tecnologia NI USRP, myRIO e myDAQ). Nelle aree pratiche saranno a disposizione NI Compact DAQ (sistema modulare di acquisizione dati per USB ed ethernet che permette di effettuare misure elettriche e di sensori per ogni tipo di applicazione), NI CompactRIO (sistema embedded riconfigurabile per il controllo e l'acquisizione dati, che può essere utilizzato in un'ampia gamma di applicazioni di monitoraggio e controllo embedded) oltre a NI USRP, MyRIO e MyDAQ, soluzioni appartenenti alla piattaforma NI per la ricerca e la didattica.

L'edizione 2014 di NIDays vedrà avvicinarsi gli interventi di personaggi come Massimo Russo, Nicola Vitiello, Raman Jamal e Jamie Smith, questi ultimi due di National Instruments. Massimo Russo è Direttore Responsabile di Wired Italia.

Nicola Vitiello è Assistant Professor all'Istituto di BioRobotica e attualmente conduce la sua attività di ricerca nell'ambito della neuro-robotica e della robotica indossabile per applicazioni biomedicali presso l'Istituto di BioRobotica della Scuola Sant'Anna. Il suo intervento verterà sul tema "Robotica indossabile e cambiamenti demografici" e affronterà le soluzioni offerte dalla bio-robotica per alleviare le conseguenze sul welfare dell'invecchiamento della popolazione, una delle quali è l'incremento dei cittadini bisognosi di assistenza a causa di indebolimento e disabilità di arti inferiori e superiori. Durante questa sessione verranno presentati i più recenti risultati ottenuti dall'Istituto di BioRobotica della Scuola Superiore Sant'Anna di Pisa nell'ambito della robotica indossabile; in particolare, verranno mostrati il sistema NEUROExos per la riabilitazione del gomito, il sistema HANDEXOS per la riabilitazione della mano ed il progetto CYBERLEGS per l'assistenza agli amputati di arto inferiore.

Rahman Jamal è dal 2011 Technical & Marketing Director Europe di National Instruments (guida l'organizzazione marketing di National Instruments in Europa). Specializzato in ingegneria elettrica, Jamal è autore di numerose pubblicazioni e di applicazioni per conferenze dedicate agli utenti finali, ed è uno stimato leader di molti forum pubblici sulle politiche di innovazione, soprattutto nel campo dei sistemi cyber-fisici

e dell'iniziativa Industry 4.0 promossa dal governo tedesco. Jamie Smith è responsabile del marketing dei prodotti industriali ed embedded, tra cui NI CompactRIO e i prodotti di visione, controllo assi e wireless. Jamie ha fatto parte del Consiglio di Amministrazione di IPSO Alliance e rappresenta National Instruments come membro di Embedded Vision Alliance. Partecipa spesso a conferenze in ambito tecnologico e di business in qualità di relatore. Ha conseguito una laurea in fisica presso la University of California Santa Barbara e una specializzazione in fisica applicata presso la Stanford University.

A NIDays 2014, come anticipato lo scorso anno alla conclusione di LabVIEW Days (il tour di NI che offre la possibilità, agli utenti sia principianti che esperti, di conoscere o approfondire le potenzialità del pacchetto di programmazione grafica LabVIEW), avrà luogo la "sessione speciale per i super esperti", battezzata "NI

LabVIEW Days Plus". La sessione offrirà contenuti avanzati e specifici per gli utenti esperti, ovvero quelli certificati CLAD o CLD e permetterà a questi ultimi di confrontarsi con i tecnici National Instruments.

Durante NIDays 2014, come di consueto gli utenti LabVIEW che lo desidereranno, potranno sostenere l'esame per il conseguimento della certificazione CLAD (Certified LabVIEW Developer) ad un prezzo scontato.

A conclusione della giornata, NI ringrazierà i presenti e taglierà con loro la torta fatta arrivare apposta per festeggiare i 25 anni di presenza in Italia.

Allora, l'appuntamento con NIDays 2014 è per mercoledì 5 marzo 2014 presso Ergife Palace Hotel, Via Aurelia 619, Roma. La partecipazione all'evento è gratuita, ma è obbligatorio registrarsi on-line su www.nidays.it. Affrettatevi perché le iscrizioni saranno accettate fino all'esaurimento dei posti.



HEARTTHROB: IL CUORE CHE PULSA

Basta scuoterlo e questo originale gadget accende i propri LED componendo una sequenza di suggestive animazioni.

Siamo a febbraio e prossimi a San Valentino, quindi guardando la forma del circuito qui descritto è naturale pensare che si tratti del solito gadget luminoso a forma di cuore; in realtà è qualcosa di più, nel senso che oltre a creare suggestivi e complessi giochi di luce quando viene sollecitato meccanicamente, utilizza il più piccolo microcontrollore prodotto da

di VICENZA THUNDERS (R. Ertolupi, D. Sprengiaro)



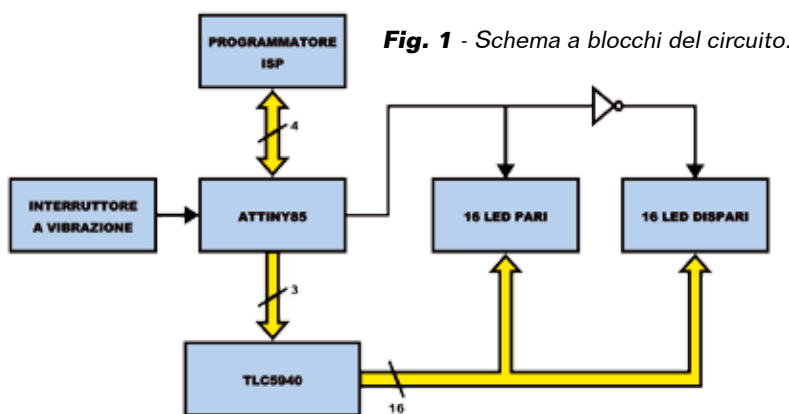


Fig. 1 - Schema a blocchi del circuito.

Atmel: l'ATtiny85. Dunque, una rivisitazione in chiave tecnologia del classico cuore pulsante a LED, che mantiene inalterato il fascino di un'idea sempre d'attualità. La scelta del microcontrollore è stata fatta secondo criteri ben precisi: serviva un controllore poco ingombrante, pratico, facilmente programmabile e dotato di ottime prestazioni in termini di consumo energetico. L'ATtiny è un micro a 8 bit che dispone di 8 kB di memoria, sei linee di I/O (da notare che sono esattamente quante ne servivano per il progetto, dato che è meglio non usare il pin dedicato al reset), quattro canali ADC a 10 bit e può arrivare, con l'utilizzo di un quarzo esterno, a una frequenza di lavoro di 20 MHz, anche se nel nostro progetto ci siamo accontentati degli 8 MHz dell'oscillatore interno. Tutto ciò sta in un case TSSOP a soli otto piedini. Una particolarità del micro è che il firmware si può scrivere utilizzando lo stesso linguaggio di programmazione e lo stesso IDE di Arduino, caricando il programma nell'ATtiny85 mediante

i pratici pin ICSP che si trovano nelle più comuni schede Arduino e che abbiamo predisposto con lo stesso layout anche nella parte posteriore del PCB per facilitare la connessione.

Ultima caratteristica, ma non per importanza, è che il piccolo controllore Atmel può essere alimentato sia dal bus USB o ICSP (ovvero 5V), sia dalla batteria LiPo da 3,7V 500mAh, a seconda della posizione del deviatore a slitta, grazie al suo range di funzionamento che spazia dai 3,5V fino ai 5,5V. Una particolarità è che, sfruttando lo sleep mode, è in grado di assicurare al sistema un'elevata autonomia in termini di durata della batteria; attivando tale modalità, arriva a consumare solo 2 µA.

SCHEMA ELETTRICO

Il circuito, il cui stampato è a forma di cuore, dispone di 32 LED ad alta luminosità posti sul bordo, accesi tramite comandi impartiti dal microcontrollore; siccome quest'ultimo non dispone di tutte le linee di I/O che servono, lo abbiamo interfacciato con un

TLC5940, che è un integrato prodotto da Texas Instruments appositamente per il controllo della luminosità di ogni singolo LED. Il componente dispone di sedici canali forniti di driver PWM a 12bit e correzione di corrente a 6 bit; ogni canale è indipendente dagli altri e l'intensità della luce emessa dal LED collegato al driver dipende dal valore medio del segnale PWM: avendo a disposizione 12 bit per il controllo di tale segnale, si deduce che dallo stato di LED spento (duty cycle = 0%) allo stato di LED totalmente acceso (duty-cycle del 100%) si hanno 4.096 livelli di luminosità.

I sei bit per la correzione della luminosità consentono 64 livelli. È possibile comunque settare la corrente massima assorbibile (infatti questo driver controlla esclusivamente il flusso di corrente verso massa) da tutti i canali dell'integrato tramite il pin 20 (siglato I_{REF}) cui va collegata una resistenza. La I_{MAX} d'uscita si ottiene mediante la formula:

$$I_{MAX} = (V_{IREF} \times 31,5) / R_{IREF}$$

Dove V_{IREF} è una tensione di riferimento di 1,24V e R_{IREF} è la resistenza applicata al piedino 20, che nel nostro caso è $R4=2k\Omega$ e che corrisponde a una I_{MAX} di 20 mA. Il produttore raccomanda di non scendere sotto i 5 mA (altrimenti non è garantita l'illuminazione dei LED) e di non superare i 120 mA, oltre i quali il chip richiede un dissipatore di calore.

Per quanto riguarda la comunicazione con il microcontrollore, il TLC5940 dispone delle seguenti linee: un ingresso seriale (SIN), un pin di segnale (XLAT), un pin multifunzione (VPRG), un piedino per i dati (DCPRG), un clock seriale (SCLK), un pin di clock per il segnale PWM (GSCLK) e una linea di reset (attiva a livello

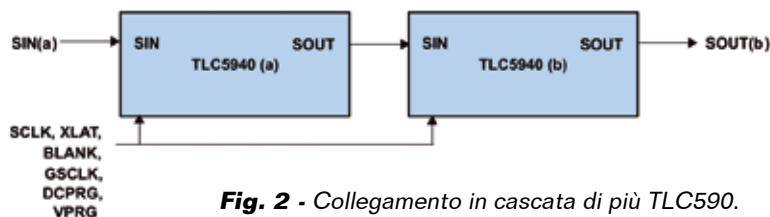
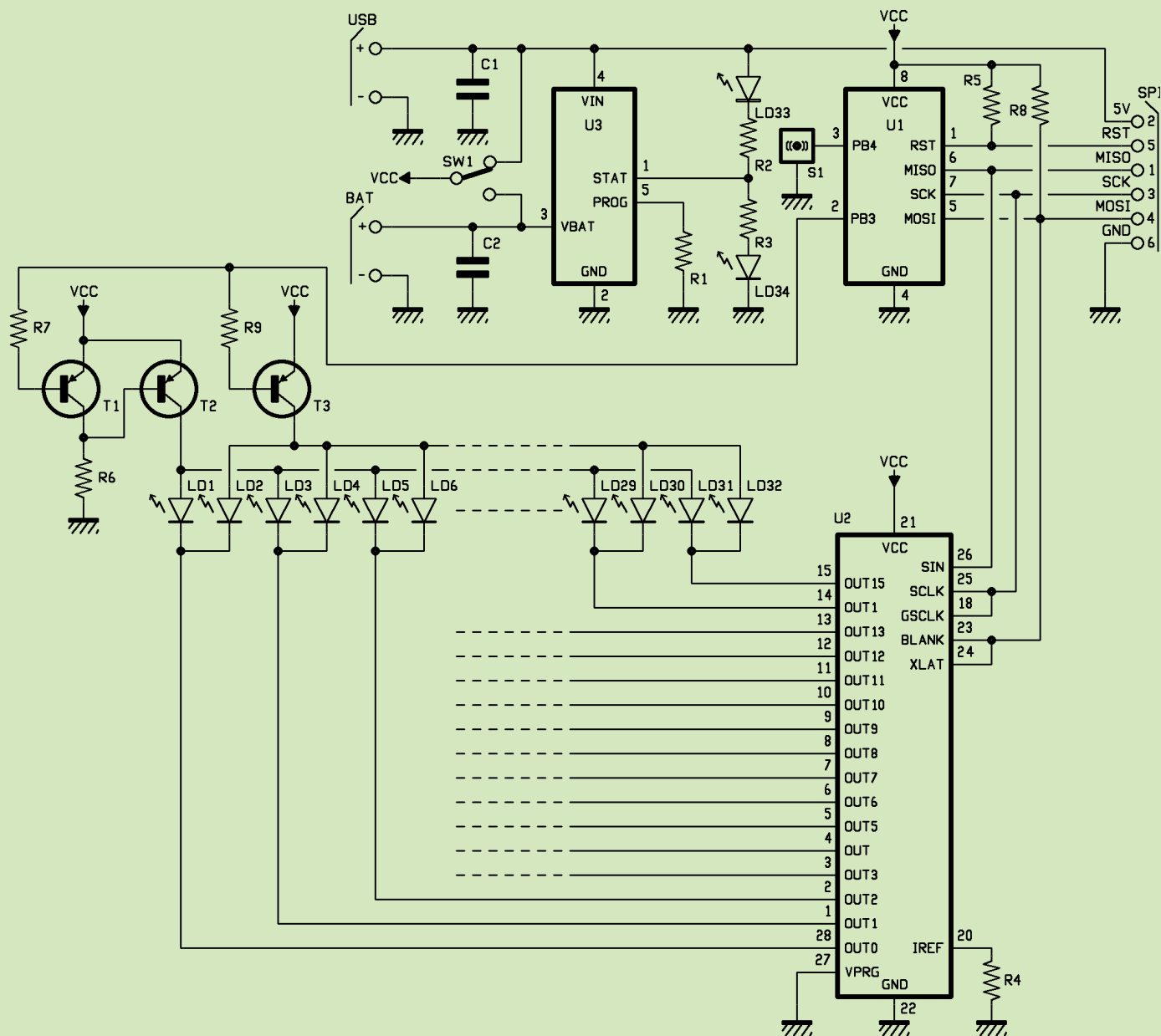


Fig. 2 - Collegamento in cascata di più TLC5940.



basso) di tutte le uscite (BLANK). Una caratteristica molto interessante di questo integrato è la possibilità di collegare in cascata più dispositivi tra loro semplicemente portando il pin di uscita seriale SOUT del dispositivo precedente al pin SIN del dispositivo successivo e collegando in parallelo tutti gli altri pin dei dati in modo da poter gestire più integrati con un solo controllore (Fig. 2).

A questo punto è d'obbligo una domanda: come fa, il TLC5940 (dotato di sole sedici uscite) a controllare 32 LED? Teoricamente

avremmo dovuto utilizzare due TLC5940, ma ciò avrebbe comportato il raddoppio del consumo (a scapito dell'autonomia del dispositivo) e dell'ingombro; ecco quindi che abbiamo trovato un'alternativa, consistente nel multiplexare le uscite. Il multiplex si ottiene tramite un unico pin del controllore che comanda tre transistor PNP di tipo BC327-25. Il pin di uscita del controllore è collegato alle basi dei transistor T1 e T3, tramite le opportune resistenze (rispettivamente R7 e R9). A sua volta, il collettore del transistor T1

è connesso alla base del transistor T2 e a massa tramite la resistenza R6, in modo da fungere da porta logica NOT per pilotare la base di T3. Infine il collettore di T2 è collegato all'anodo dei LED dispari mentre il collettore di T3 all'anodo dei LED pari (la numerazione dei LED parte dal vertice inferiore del cuore e prosegue in senso antiorario, così da seguire gli output del TLC5940).

I catodi dei LED, invece, sono collegati a due a due in modo successivo e, a loro volta, alle uscite del driver.

Adesso riprendiamo quanto detto all'inizio di questo articolo sul fatto che il circuito reagisce a uno stimolo esterno; a permetterlo è l'interruttore a vibrazione S1, al cui interno si trova una piccola sfera metallica che, quando si fa vibrare il componente, mette in collegamento il reoforo centrale con l'involucro esterno determinando la chiusura dei contatti (involucro e pin centrale). Il piedino del microcontrollore che legge il sensore è stato dotato di resistenza di pull-up (interna) in quanto quando il contatto è aperto (a riposo) il piedino stesso si troverebbe esposto ai disturbi e quindi influenzabile e soggetto a falsi allarmi.

Terminata l'analisi del blocco di controllo, possiamo ora alla descrizione della sezione di alimentazione: il circuito consuma in media 55 mA; tale energia può provenire dal connettore USB tipo mini-b che si trova sul retro della scheda, oppure dalla batteria LiPo monocella da 3,7 V – 500 mAh. A decidere da dove proviene l'alimentazione è il deviatore SW1: se il cursore è spostato verso la batteria, il disposi-

tivo è alimentato a batteria; se è spostato verso il connettore USB, il circuito è spento se alimentato a batteria, ovvero acceso e in carica se alimentato tramite USB e, infine, acceso in carica e programmabile se alimentato tramite i pin ICSP.

Per quanto riguarda la carica della batteria, è stato necessario utilizzare un componente costruito appositamente per tale scopo: l'MCP73831. Questo integrato, prodotto da Microchip, si presenta in un case SOT-23 SMD a 5 pin ed è un gestore della carica di batterie LiPo da una cella a 3,7V. Per verificare lo stato di avanzamento del processo di carica della batteria, sono stati predisposti i LED LD33 (rosso, indica che la batteria è in carica) e LD34 (giallo, segnala che la batteria ha terminato la carica). L'MCP73831 ha cinque modi di funzionamento, di seguito elencati.

- **SHUTDOWN MODE:** pur essendo alimentato, il controller non carica la batteria e l'uscita STAT si trova in stato di alta impedenza (LED LD33 e LD34 accesi). Questa modalità si verifica se l'alimentazione V_{DD}

del controller è più bassa della tensione V_{BATT} della batteria o della tensione di soglia V_{UVLO} dell'Under Voltage Lock Out. Il controller si può portare in quiete anche se la resistenza R_{PROG} , che serve a definire la corrente di carica della batteria, ha un valore maggiore di 200k Ω .

- **PRECONDITIONING**

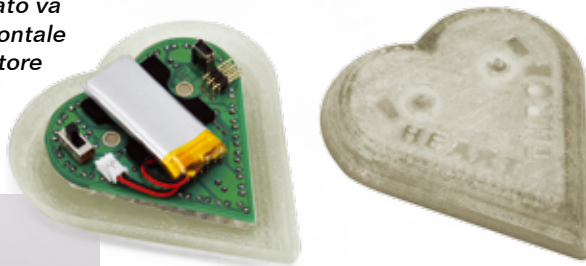
MODE: è la prima fase della carica; l'MCP73831 verifica la tensione della batteria e se la trova minore della V_{PTH} (ossia troppo bassa per avviare la carica normale) inizia a caricare la batteria stessa con la corrente di preconditionamento costante I_{PREG} finché essa non supera la V_{PTH} . Questa modalità serve a preservare la capacità della batteria, in quanto la carica avviene ad un valore di corrente ridotto per non sollecitare eccessivamente la struttura. L'uscita STAT si trova a livello basso ovvero il LED LD33 acceso. Se invece la tensione viene trovata maggiore della V_{PTH} , si passa alla modalità Fast Charge Mode.

- **FAST CHARGE MODE:** il controller carica la batteria con corrente costante I_{REG} finché la tensione non raggiunge il valore di uscita regolata V_{REG} . Il valore di I_{REG} può essere impostato tramite la resistenza R_{PROG} collegata al pin PROG del controller, seguendo la formula $I_{REG} = 1.000V / R_{PROG}$. Nel nostro caso R_{PROG} vale 2k Ω , quindi la corrente è 500 mA (il massimo erogabile dall'integrato). Per questo è stato necessario prevedere, sullo stampato, piazzole la cui area consente di smaltire il calore dell'integrato. L'uscita STAT si mantiene a livello basso.

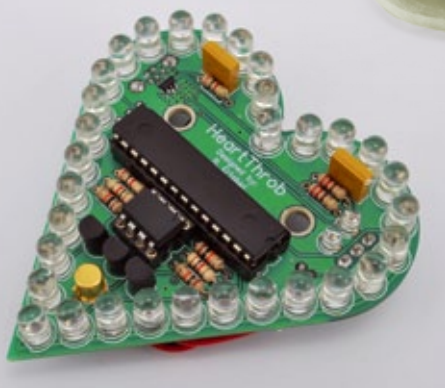
- **CONSTANT VOLTAGE**

MODE: in questa fase il con-

Una volta completato il montaggio, lo stampato va inserito nella metà frontale dell'apposito contenitore



(quella che riporta le scritte) e sul retro va applicata la batteria, come mostrato in fotografia. Il tutto va chiuso con la parte posteriore del contenitore (quella che riporta la cava per il deviatore e per il connettore mini-USB), mediante due viti.



troller carica la batteria con tensione costante V_{REG} finché la corrente di essa non raggiunge il valore di terminata carica ITERM. L'uscita STAT si mantiene a livello basso.

- **CHARGE COMPLETE**
MODE: finita la carica, il controller ritorna in stato di quiete e l'uscita STAT commuta a livello alto in modo da accendere il LED LD34 giallo.

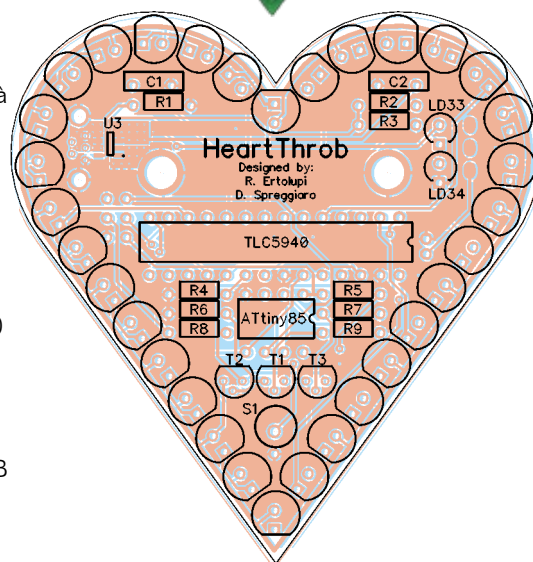
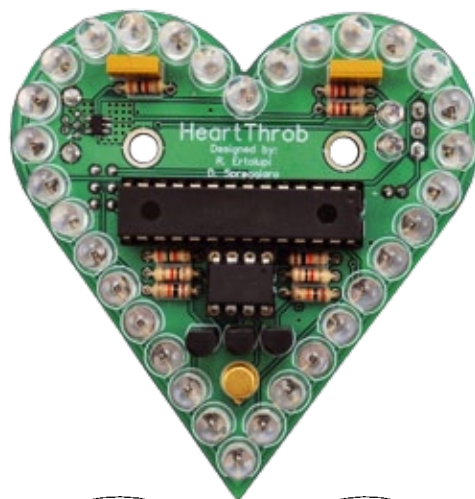
Con la batteria da 500 mAh sono richiesti 90 minuti per la carica completa, che garantirà 9 ore di funzionamento con i LED accesi o 50 ore in sleep mode.

REALIZZAZIONE PRATICA

Veniamo ora ad alcuni consigli sul montaggio del dispositivo. Innanzitutto ricordiamo che il connettore USB, i pin ICSP, il connettore della batteria e il deviatore a slitta SW1 vengono montati sulla faccia posteriore del PCB. Per quanto riguarda l'ordine di montaggio, consigliamo di montare per primo l'MCP73831, procedendo con un saldatore a punta fine (max 30 W) una stazione ad aria calda, pinzette e flussante di bassa densità; il flussante vi aiuterà a non cortocircuitare i contatti adiacenti, che sono molto vicini tra loro. Nell'effettuare la saldatura, togliete il saldatore o il getto d'aria calda appena la lega saldante si è ben sciolta e ha unito i terminali alle rispettive piazzole. Sistemato l'MCP73831, montate le resistenze e i condensatori, procedendo in ordine d'altezza, quindi i due LED da 3 mm, il deviatore a slitta SW1, il connettore ICSP, quello della batteria e, infine, lo zoccolo DIP a 28 pin. Terminate con i LED del contorno della basetta. Ricordiamo di lasciare sollevati sia il connettore della batteria, sia i pin ICSP, allo scopo di evitare cortocircuiti con

Elenco Componenti:

- R1, R4: 2 kohm
R2, R3: 220 ohm
R5: 10 kohm
R6: 1 kohm
R7: 10 kohm
R8: 10 kohm
R9: 1 kohm
C1: 100 nF 100 VL poliestere
C2: 100 nF 100 VL poliestere
U1: ATTINY85 (MF1117)
U2: TLC5940
U3: MCP73831
T1: BC327-25
T2: BC327-25
T3: BC327-25
LD1 ÷ LD32: LED alta luminosità 5 mm rosso
LD33: LED 3 mm rosso
LD34: LED 3 mm giallo
SW1: Deviatore a slitta
S1: Interruttore a vibrazione (cod. 6710-MOVMINI)
- Varie:
- Strip maschio 6 poli 90° (2 pz.)
 - Zoccolo 4+4
 - Zoccolo 14+14
 - Batteria Lipo 3.7V/500mAh
 - Connettore per batteria
 - Connettore verticale Mini-USB
 - Circuito stampato



le piazzole dei LED sottostanti. Terminato il circuito, dovete costruirgli un contenitore adatto; e cosa c'è di meglio che stamparlo (o farselo stampare da un amico) con una stampante 3D? Il nostro è stato realizzato con PLA semitrasparente. Il file STL del contenitore è disponibile sul nostro sito, assieme agli altri file del progetto. Dopo aver posizionato le due parti del case una sopra l'altra con nel mezzo la scheda, prendiamo le due viti autofilettanti di diametro 2 mm e di lunghezza 13 mm e le inseriamo nei fori. Prima di avvitare completamente ci assicuriamo che sia tutto a posto, in

particolare che la batteria e il PCB siano nelle loro sedi e che non causino intralcio all'operazione che stiamo compiendo. Prima di chiudere il contenitore dobbiamo assicurarci che la batteria stia ferma nella posizione in cui la mettiamo, ad evitare che subisca danni. Per fare ciò si può usare un po' di biadesivo spugnoso per attaccarla sul retro PCB, in modo che essa stia in mezzo ai due fori di fissaggio.

IL FIRMWARE

Vediamo ora nel dettaglio la parte focale del progetto, ovvero il firmware da caricare nel micro-

Listato 1 – Funzione utilizzata per aggiornare il TLC5940

```
void heartUpdate (unsigned int time) {
    const unsigned long startTime = millis();

    do {
        digitalWrite(3, 0);           //Abilitazione dei LED "pari"

        for(byte i=0; i<16; i++)
            tlc5940.set(i, values[i*2]); //Impostazione del valore dei LED

        tlc5940.update();             //Invio dati al TLC5940

        digitalWrite(3, 1);           //Abilitazione dei LED "dispari"

        for(byte i=0; i<16; i++)
            tlc5940.set(i, values[i*2+1]); //Impostazione del valore dei LED

        tlc5940.update();             //Invio dati al TLC5940

    } while (millis() - startTime <= time); //Ripeti il processo per un numero di ms
                                           //maggiore o uguale a quello del parametro "time"
}
```

controllare; per comprendere meglio il codice esaminiamo, per prima cosa, il diagramma di flusso dei processi che l'ATtiny deve svolgere (Fig. 3). Sul primo blocco "inizializzazione del sistema" non c'è molto da dire: si tratta delle classiche operazioni che si svolgono all'accensione, quali inizializzare gli ingressi e le uscite, gli spazi di memoria e altre operazioni sulle quali non occorre dilungarsi troppo. Dopo aver fatto ciò, il sistema attende due secondi per evitare che l'interruttore a vibrazione, avvertita la scossa dovuta all'accensione, faccia partire subito una sequenza luminosa; tale funzione è facoltativa e la parte di firmware corrispondente si può cancellare.

Il blocco seguente è molto importante perché determina tutta la struttura del codice che vedremo successivamente, in cui è fondamentale la modalità risparmio energetico, più propriamente detta "sleep". Questo particolare stato serve per fermare la CPU, il clock e altre periferiche integrate; ciò implica che il programma viene messo in pausa fino a quando il microcontrollore non si sveglia. Dunque il problema diventa come svegliare il nostro ATtiny: per risolverlo andiamo a vedere quali periferiche sono ancora

	Active Clock Domains					Oscillators	Wake-up Sources				
	clk_cpu	clk_flash	clk_io	clk_adc	clk_fox	Main Clock Source Enabled	INT0 and Pin Change	SPM/EEPROM Ready	USI Start Condition	ADC	Other I/O Watchdog Interrupt
Sleep Mode											
Idle			X	X	X	X	X	X	X	X	X
ADC Noise Reduction				X		X	X ⁽¹⁾	X	X	X	X
Power-down							X ⁽¹⁾		X		X

Tabella 1 - Modalità di Sleep possibili sull'ATtiny85 e relative caratteristiche.

attive dalla **Tabella 1** presa dal datasheet dell'ATtiny85. Qui vediamo le caratteristiche dei tre possibili livelli di sleep; noi useremo l'ultimo, che è quello che permette di consumare meno corrente (circa 2 μ A). Per il risveglio abbiniamo un Pin Change Interrupt (PCINT) all'ingresso cui è collegato l'interruttore a vibrazione, così da risvegliare il microcontrollore ogni volta che si verifica un cambio di stato logico, ovvero rilevando una vibrazione. Tornando al flow-chart, vediamo proprio questo particolare nell'operazione successiva: il sistema attende in sleep una vibrazione che lo svegli e gli faccia proseguire l'esecuzione del programma. Una volta svegliatosi deve interrompere la modalità risparmio energetico riattivando i moduli e le funzionalità che ser-

vono per lavorare correttamente. Fatto ciò deve valutare se le vibrazioni che l'hanno svegliato sono durature e non sporadiche; in questo caso farà accendere i LED secondo una sequenza a caso tra quelle programmate e poi ricomincerà il ciclo tornando in modalità risparmio energetico. Detto ciò, andiamo a vedere come questi discorsi si traducono nel codice vero e proprio, compilabile usando l'IDE di Arduino. All'inizio si nota la definizione di alcuni parametri che vedremo

come impostare successivamente in fase di configurazione, poiché sono valori empirici. Dopodiché si passa ad includere

Listato 2 – Esempio di effetto

```
void throb() {
    const unsigned long startTime = millis();

    while (millis() - startTime <= DURATION) {
        for(char v=0; v<=117; v++) {
            for(byte j=0; j<32; j++)
                values[j] = v*35;

            heartUpdate(1);
        }
        heartUpdate(50);

        for(char v=117; v>=0; v--) {
            for(byte j=0; j<32; j++)
                values[j] = v*35;

            heartUpdate(10-v/13);
        }
        heartUpdate(50);
    }
}
```

le necessarie librerie, di seguito elencate.

- La libreria **Sleep**, che mette a disposizione dei metodi per gestire la modalità di sleep e il relativo risveglio. Questa è una libreria già presente nel pacchetto delle librerie Arduino, più precisamente nel pacchetto delle librerie AVR, quindi non c'è alcunché da scaricare.
- La libreria **Power**, che permette di accendere e spegnere alcuni moduli del nostro microcontrollore con metodi molto semplici e pratici. Come la Sleep, anche questa libreria fa parte del pacchetto AVR e dunque vale quanto già detto. In particolare, notiamo come queste due librerie vadano a braccetto dato che si occupano di tematiche simili. La sostanziale differenza fra le due sta nel fatto che Power agisce sui moduli periferici alla CPU, mentre Sleep agisce direttamente sulla CPU.
- La libreria **PinChangeInterrupt**, che è una libreria specifica per gli ATtiny e semplifica molto la gestione dei PCINT. Questa libreria non è presente nativamente nell'IDE di Arduino e vedremo successivamente, in fase di programmazione, dove poterla trovare e come inserirla fra le nostre librerie.
- La libreria **Tiny85_TLC5940**, che è una libreria fatta su misura per l'ATtiny85 e che permette di controllare uno o più TLC5940 utilizzando solo 3 pin del microcontrollore. Chiaramente questo comporta una minore capacità di sfruttare le potenzialità del nostro driver LED, ma per quello che dobbiamo fare in questo progetto ci possiamo accontentare. Anche questa libreria non è presente nel pacchetto originale delle librerie di Arduino,

quindi spiegheremo in seguito, descrivendo il processo di programmazione, da dove scaricarla e come poterla utilizzare.

Detto ciò, entriamo nel vivo del programma e andiamo ad analizzare le seguenti parti del codice, tralasciando, però, quelle operazioni in cui i commenti posti affianco sono sufficienti per spiegare la loro funzione. Andiamo quindi a spiegare il significato del metodo "set_sleep_mode(SLEEP_MODE_PWR_DOWN)": questo è un metodo della libreria Sleep che serve per impostare quale, tra le modalità di sleep disponibili nella **Tabella 1**, si intende usare. Si deve fare attenzione che così facendo si effettua semplicemente una scelta, senza far partire tale modalità.

Poi arriviamo ad un altro processo un po' particolare, ovvero la disattivazione del modulo ADC e il suo successivo spegnimento. Infatti queste due operazioni non sono equivalenti e questo modulo necessita prima di essere disattivato e poi di essere spento. Il modulo ADC, come molti probabilmente già sanno, è il convertitore analogico-digitale che serve per effettuare le letture analogiche. Visto che, nell'ATtiny85, è la periferica che consuma più corrente, dopo il TIMER1, è opportuno spegnerlo quando non lo si usa. Entrando nel loop, troviamo la routine **sleepNow** nella quale il nostro microcontrollore fa una serie di operazioni per entrare in modalità sleep:

- imposta il pin 3, usato per la polarizzazione dei transistor, come ingresso e abilita la relativa resistenza di pull-up; questo piccolo stratagemma consente di erogare meno corrente dal pin e di polarizzare il solo transistor T2;
- resetta la comunicazione con il



Fig. 3
Diagramma di flusso del firmware.

TLC5940 e spegne tutte le sue uscite utilizzando il metodo "tlc5940.init()";

- disabilita i moduli TIMER0, TIMER1 e USI con "power_timer0_disable(); power_timer1_disable(); power_usi_disable()"; questo ci consente di ridurre ulteriormente il consumo del micro;
- imposta un Pin Change Interrupt sul pin 4 (quello con cui viene letto l'interruttore a vibrazione) utilizzando "attachPcInterrupt(4, wakeUp, CHANGE)"; in particolare, con questo metodo, facciamo in modo che il microcontrollore

esegua la ISR "wakeUp" ogni volta che ci sarà un cambio di stato sul pin 4.

Noi useremo quest'ultima funzionalità per risvegliare l'ATtiny85 dal suo sonno. La routine wakeUp potrebbe anche essere vuota, ma nel nostro caso abbiamo inserito una funzione che incrementa la variabile beatCounter: questo ci permetterà di valutare empiricamente l'intensità della vibrazione.

Ora il nostro piccolo microcontrollore è pronto per entrare in Sleep, grazie al metodo **sleep_mode()**: la CPU rimarrà bloccata su questo punto del programma fino all'arrivo di una vibrazione che la risvegli.

Supponendo ora che il PCINT faccia risvegliare la CPU, proseguiamo con la descrizione del codice seguendo l'ordine logico, trovando "sleep_disable()" e "power_timer1_enable()", due metodi che servono per far tornare operativo il nostro microcontrollore. Più precisamente, il primo serve per far ripartire la CPU e per uscire definitivamente dallo sleep mode, mentre il secondo è per riabilitare il TIMER1 che ci serve per poter usare le funzioni *millis* e *micros*.

Il nostro microcontrollore, dopo essersi ripreso, valuta se la vibrazione che l'ha svegliato è sufficientemente duratura in base ai parametri impostati all'inizio del codice. Se ciò non risulta, l'ATtiny85 ricomincerà il loop e tornerà in sleep mode. Se invece la vibrazione risulta abbastanza forte, il micro eseguirà la routine **lightEffect** che ha lo scopo di far partire un effetto luminoso sui LED, disabilitando prima l'interrupt per evitare che interrompa e disturbi i processi successivi. Bisogna poi riabilitare i moduli TIMER0 e USI che servono per la comunicazione con il TLC5940 e che prima erano stati disabilitati. È necessario, inoltre, reimpostare il pin 3 come output.

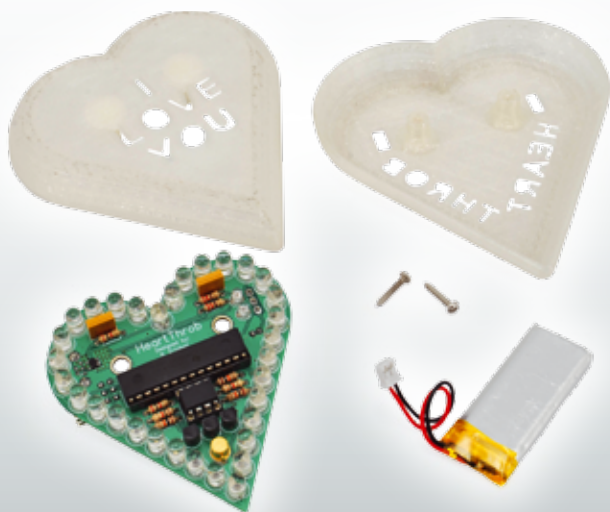
Fatto ciò il cuore è pronto per accendersi; manca solo da scegliere una funzione a caso tra quelle programmate e per farlo il micro fa un calcolo pseudo-casuale, utilizzando le funzioni random e micros. Il risultato di tale operazione viene usata in uno switch che determina quale routine eseguire. Si noti che con questo metodo una funzione non può mai ripetersi due volte di seguito; ciò non è, chiaramente, una cosa indispensabile, ma è semplicemente una scelta di stile.

Andiamo ora a esaminare una delle possibili routine per capire il principio di funzionamento usato: per esempio prendiamo in considerazione la funzione **throb** che è programmata per emulare il battito cardiaco.

Innanzitutto vediamo che c'è un ciclo while che viene eseguito ripetutamente per un tempo almeno uguale al parametro DURATION impostato all'inizio del codice. In secondo luogo si può vedere che si vanno a modificare i dati contenuti nell'array **values**; questi rappresentano il livello di luminosità di ogni singolo LED da 0 a 4095, dove 0 sta per spento e 4095 per acceso. Vediamo poi che si entra in un'altra routine chiamata **heartUpdate**, che ha lo scopo effettivo di inviare i dati al driver dei LED e di gestire la loro accensione polarizzando i transistor a seconda delle necessità. Osservando il codice presente in questo metodo, notiamo che il parametro in ingresso time viene usato per quantificare i millisecondi nei quali il processo contenuto in un ciclo do-while verrà eseguito ripetutamente. Dentro questo ciclo troviamo due parti di codice quasi uguali nelle quali si abilitano i LED pari o dispari ad intermittenza e, nel frattempo, si attivano le uscite del TLC5940 utilizzando i metodi della relativa libreria:

- **"tlc5940.set(i, values[i*2])"** usato per settare il valore di ogni singola uscita prendendo tali valori dall'array values, senza però inviarli;
- **"tlc5940.update()"** usato per inviare i dati al nostro LED driver.

Quest'ultimo metodo è molto particolare perché richiede circa un millisecondo per essere eseguito e solo durante questo periodo le uscite del TLC5940 vengono



Insieme delle parti per realizzare il cuore a LED: il guscio è stato stampato in 3D usando del PLA trasparente.

attivate; ciò comporta la necessità di eseguire questo metodo il più frequentemente possibile, senza usare alcun delay perché vorrebbe dire tenere spenti i LED per quel lasso di tempo. Tutti questi inconvenienti sono causati dal fatto che l'ATtiny85 non dispone di potenzialità sufficienti per comunicare al meglio con questo driver LED.

Infine, la routine termina usando gli stessi principi già spiegati e dunque il loop riprende facendo tornare il microcontrollore in modalità sleep.

PROGRAMMAZIONE

A questo punto dobbiamo programmare il nostro microcontrollore e lo faremo usando una scheda Arduino come programmatore ISP.

Per prima cosa dobbiamo procurarci il core ATtiny e il file sorgente, disponibili insieme agli altri file del progetto sul nostro sito www.elettronica.in.it; fatto ciò, scompattiamo la cartella compressa, andiamo nella nostra cartella degli sketch di Arduino (per verificare la locazione di questa cartella è sufficiente aprire l'IDE e andare su *File->Preferenze*: lì vedremo la posizione della cartella degli sketch). Se non sono presenti creiamo due cartelle e le chiamiamo *hardware* e *libraries*, e spostiamo il contenuto della cartella che abbiamo decompresso nella cartella *hardware*. Ora dobbiamo scaricare le due librerie che ci servono e che non sono presenti nel pacchetto delle librerie di Arduino: *PinChangeInterrupt* e *Tiny85_TLC5940*, entrambe scaricabili da elettronica.in.it. Dopo averle scaricate entrambe, scompattiamo i relativi ZIP e spostiamo il contenuto nella cartella *libraries* che abbiamo visto in precedenza.

Teoricamente siete a posto così,

tuttavia va detto che se nel vostro computer avete Windows o Mac OS X, dovete superare un problema: un bug nel compilatore AVR presente nell'IDE di Arduino non permette la compilazione di sketch destinati ad essere caricati nel microcontrollore ATtiny, che occupano in memoria più di 4,8 kB. Di seguito spieghiamo come risolvere questo problema.

- Per Windows, scaricate la cartella compressa dal sito elettronica.in.it ed estraete il contenuto. All'interno dovreste trovare una cartella chiamata *hardware* che copierete. Ora andate all'interno della cartella in cui avete installato Arduino (la potete riconoscere perché troverete il file *arduino.exe*) e incollate la cartella che avevate copiato. Sicuramente Windows vi metterà in guardia dicendovi che esiste già questa cartella; voi dategli di unire le cartelle e fate così anche per le sottocartelle fino a quando apparirà una finestra in cui dovete acconsentire a sostituire il file *ld.exe* presente con quello nuovo che state inserendo.
- Per Mac OS X scaricate la cartella compressa dal sito elettronica.in.it ed estraete il contenuto. All'intero dovreste trovare un file senza estensione con il nome *ld*, che copierete. Ora posizionatevi con il cursore del mouse sopra l'icona Arduino all'interno della cartella applicazioni e cliccate tasto destro,

scegliete *mostra contenuto pacchetto* e navigate nelle cartelle fino ad arrivare a */Contents/Resources/Java/hardware/tools/avr/avr/bin/* e qui sostituite il file *ld* presente con quello che avete copiato in precedenza.

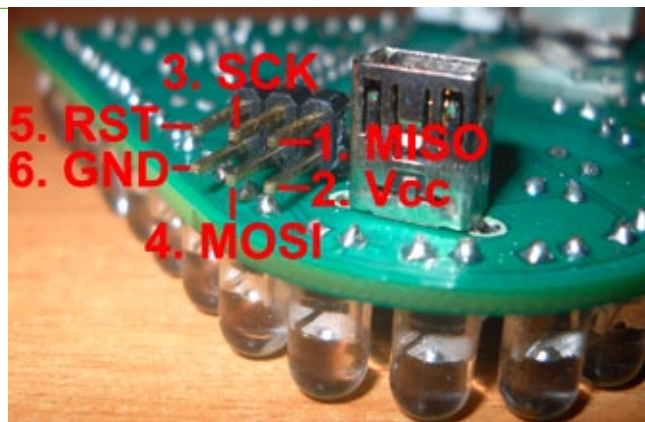
Fatto ciò dovreste essere in grado di compilare il programma per la vostra HeartThrob. Per farlo andate su *Strumenti->Tipo di Arduino* e selezionate "ATtiny85 @ 8 MHz (internal oscillator; BOD disabled)".

Per collegare il circuito all'Arduino usiamo il connettore ICSP presente sul retro: la piedinatura di tale connettore, che possiamo vedere in Fig. 4, è uguale a quella del connettore presente nella vostra scheda Arduino.

Per prima cosa assicuriamoci che il cursore di SW1 sia spostato verso l'alto, ovvero che il micro sia alimentato con i 5V. Dunque effettuiamo i seguenti collegamenti:

- pin 1 dell'ICSP sul pin 1 di ICSP dell'Arduino (MISO);
- pin 2 dell'ICSP sul pin 2 di ICSP dell'Arduino (Vcc);
- pin 3 dell'ICSP sul pin 3 di ICSP dell'Arduino (SCK);
- pin 4 dell'ICSP sul pin 4 di ICSP dell'Arduino (MOSI);
- pin 5 dell'ICSP sul pin dell'Arduino usato per gestire il reset del microcontrollore da programmare, che nell'Arduino Mega è il pin 53 mentre negli altri è il pin 10;

Fig 4
Piedinatura del connettore ICSP: notare che il connettore si monta dal lato saldature del c.s. facendo attenzione che non tocchi le piazzole sottostanti i pin.



Guarda HeartThrob in azione su YouTube!



Vuoi vedere in azione il nostro Cuore palpitante? Scansiona il QR Code oppure collegati alla pagina <http://goo.gl/Q0rb3l>



- pin 6 dell'ICSP della HeartThrob al pin 6 di ICSP dell'Arduino (GND).

Tornando al firmware, iniziamo caricando sul nostro Arduino lo sketch "ArduinoISP" che si trova tra gli esempi (assicuratevi di aver scelto il giusto tipo di Arduino nelle opzioni, se prima avevate cambiato), poi andiamo su *Strumenti->Programmatore* e scegliamo "Arduino as ISP". Ora dobbiamo impostare i fuse del nostro ATtiny85 affinché lavori a 8 Mhz (l'impostazione di fabbrica è 1 MHz). Per fare ciò andremo a caricare un finto bootloader il quale, in realtà, ha il solo scopo di modificare le impostazioni dei fuse. Dunque andiamo su *Strumenti->Tipo di Arduino* e selezioniamo "ATtiny85 @ 8 MHz (internal oscillator; BOD disabled)" e poi andiamo su *Strumenti->Scrivi Bootloader*. Se tutto è stato fatto come descritto, dovrete vedere alcuni LED del cuore lampeggiare e, dopo un po', la conferma dell'operazione avvenuta sull'IDE. Sicuramente vedrete anche quattro errori "avrdude: please define PAGEL and BS2 signals in the configuration file for part ATtiny85", ma non allarmatevi: questo tipo di errori

viene segnalato ogni volta che programmerete l'ATtiny e potete considerarli una conferma che l'operazione è avvenuta con successo. Se invece vi viene segnalato qualche altro tipo di errore, probabilmente avete sbagliato qualcosa nei passaggi precedenti, quindi togliete l'alimentazione al circuito e verificate che i collegamenti e le impostazioni sull'IDE siano corrette.

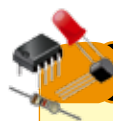
Ora non ci resta che caricare il programma sul microcontrollore e, per farlo, è sufficiente aprirlo nell'IDE e, senza modificare niente dal processo precedente, andare su *File->Carica con un programmatore*. Se tutto va a buon fine, deve verificarsi quello che abbiamo descritto prima per il caricamento del bootloader ma ottenendo, questa volta, solo due di quegli errori.

CONFIGURAZIONE

Passiamo ora a vedere come poter modificare il programma a seconda delle nostre esigenze. All'inizio ci sono dei parametri facilmente modificabili: il primo è DURATION, che stabilisce la durata di ogni effetto luminoso, espresso in millisecondi. Il secondo e il terzo sono L_VIBRATION e D_VIBRATION. Questi due parametri servono per impostare la sensibilità alle vibrazioni in base al tipo di interruttore a vibrazione di cui si dispone: per ottenere la massima sensibilità impostate il primo a 0 e il secondo a 2; ciò farà partire i LED alla prima vibrazione avvertita. Se invece desiderate ridurre la sensibilità, dovete aumentare il valore dei due parametri, per esempio mettendo il primo a 180 e il secondo a 90. Chiaramente questi sono valori empirici e variabili a seconda della sensibilità di ogni singolo interruttore a vibrazione. È quasi impossibile

misurare con precisione l'intensità di una vibrazione utilizzando questo interruttore a vibrazione e questo metodo. Per valutare se disponete un modello molto sensibile o meno, potete scuotere la HeartThrob e porla vicino al vostro orecchio: dovrete sentire lo spostamento della pallina interna all'interruttore sulle pareti. Se questo rumore continua per molto tempo, circa più di un secondo dopo la fine della vibrazione, allora il vostro interruttore a vibrazione è molto sensibile, altrimenti no.

Il quarto parametro è NUMBER_OF_FUNCTIONS che serve nel caso vogliate aggiungere o togliere altri effetti luminosi. In tal caso dovrete anche fare delle modifiche nella routine lightEffect e in particolare nello switch, facendo in modo che siano presenti tutti i case da 0 a NUMBER_OF_FUNCTIONS-1. ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità; il master del circuito stampato, il software col quale programmare il microcontrollore e il file STL del contenitore possono essere scaricati gratuitamente dal sito della rivista www.elettronica.in. Gli altri componenti sono reperibili presso Futura Elettronica: ATTINY85 Euro 4,50; TLC5940 Euro 8,00; MCP73831 Euro 1,50; Batteria Lipo 3.7V/500mAh Euro 4,90; Interruttore a vibrazione Cod. 6710-MOVMINI Euro 14,90. 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>



FORM1

ALLA PROVA

Abbiamo messo alla prova la stampante stereolitografica Form1; ecco le nostre impressioni a caldo.



di LUCA PERENCIN

Dopo quasi un anno di attesa, anche noi abbiamo finalmente ricevuto il nostro esemplare di Form1, la stampante in stereolitografia che abbiamo contribuito a finanziare tramite Kickstarter. Anche se avevamo già avuto modo di vedere

diverse foto e filmati della stampante, trovarselo davanti fa decisamente un certo effetto. L'aspetto estetico è curato in ogni dettaglio, e, seguendo una logica molto "americana", anche l'imballaggio non fa eccezione. Aprendo la voluminosa scatola ci

CARATTERISTICHE TECNICHE

- **Tecnologia:** Stereolithography (SLA)
- **Area di stampa:** 125 x 125 x 165 mm
- **Dimensioni del pezzo più piccolo stampabile:** 300 micron
- **Risoluzione minimo strato stampabile:** 25 micron (altre risoluzioni, 50, e 100 micron)
- **Velocità di stampa:** 1,5 cm/ora
- **Dimensioni:** 30 x 28 x 45 cm
- **Peso:** 8 kg
- **Alimentazione e consumi:** 100-240V 1,5A 50/60Hz 60W
- **Materiale di stampa:** Acrylate Photopolymer Resin

si trova subito davanti un foglio illustrato con le prime indicazioni di assemblaggio, utili anche se è tutto piuttosto intuitivo; la stampante presenta un piccolo display frontale e sul retro ha un pulsante, un ingresso per il connettore di alimentazione e una porta USB.

Il cabinet è ben rifinito e con pochi elementi essenziali; il voluminoso coperchio arancione ha uno scopo non solo estetico, ma serve anche per proteggere l'utente dalle eventuali rifrazioni del laser interno. Sollevandolo, troviamo il supporto rimovibile per la stampa del materiale (quello che potrebbe essere il piatto di una stampa ad estrusione) e la vaschetta per la resina. La parte inferiore, un contenitore in alluminio satinato e arrotondato, cela il cuore della stampante, ovvero l'elettronica di comando, oltre, ovviamente, al gruppo laser, ai motori galvanometrici e allo specchio rifrangente, che sono i veri artefici della stampa con questa tecnologia.



Il meccanismo che proietta il raggio laser.

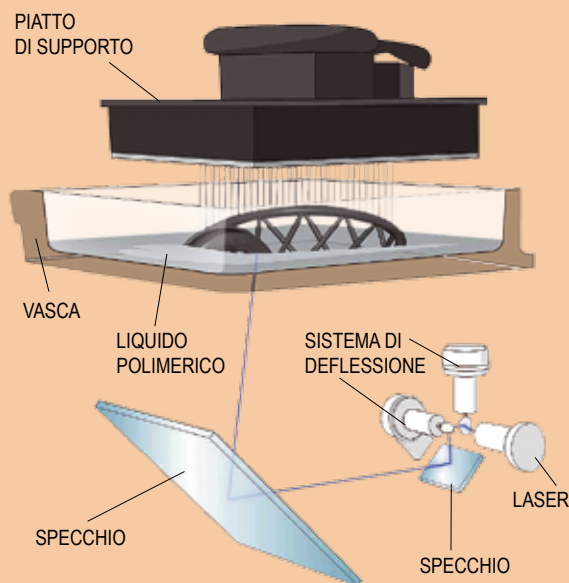
Ultimo elemento è il sistema di oscillazione della vaschetta, studiato per distribuire uniformemente la resina prima di ogni passaggio; questo sistema, come vedremo più avanti, ci ha dato anche qualche problema. Il secondo scatolone, meno voluminoso del primo, ma altrettanto importante, contiene il "finishing kit", ovvero tutto l'occorrente per rifinire il pezzo dopo la stampa; infatti la stampa stereolitografica, utilizzando della resina come componente principale, ha

bisogno di una fase di finitura finale per rimuovere i residui di materiale non stampato e i supporti generati in fase di stampa. Il kit è composto da un supporto quadrato con diverse sezioni: il supporto per il piatto di stampa, uno scomparto per un contenitore ermetico, contenente un'ingegnosa griglia, per il lavaggio del pezzo con alcool isopropilico, oltre ad uno spazio con un supporto spugnoso per appoggiare i pezzi in lavorazione, e ad un ultimo vano dove appoggiare i



La stampante Form1 completamente montata: il coperchio arancione protegge la nostra vista durante il funzionamento.

La tecnica di stampa SLA



Stereolitografia (SLA o SL) è un termine coniato nel 1986 da Chuck Hull, considerato uno degli inventori della stampa 3D; è lui, infatti, uno dei fondatori della 3D System inc, una famosa azienda produttrice di stampanti 3D per l'uso prettamente professionale. Il processo consiste nel ricostruire oggetti solidi mediante l'irradiazione di una resina fotopolimerica (ovvero sensibile alla luce) in sottili strati successivi, sovrapposti l'uno all'altro. Questa tecnica permette di avere ottime risoluzioni e una relativa velocità di esecuzione.

Peraltro tale soluzione è quella che per diversi anni è stata utilizzata nelle stampanti industriali e professionali, solo che ora è stata adattata, grazie anche a progetti come la Form1 o la ilios HD (che hanno permesso di beneficiarne a basso costo), al mercato consumer e hobbistico. Diversamente da altri tipi di stampa, come quella ad estrusione, l'uso dei supporti ausiliari è pressoché obbligatoria, per evitare flessioni, a causa della gravità, durante la stampa del pezzo. I supporti vanno rimossi in una successiva fase di finitura.

vari accessori, compresi nel kit, come una spatola, una bottiglia con cannuccia, pinzette e alcuni panni in microfibra. Occorre inoltre dotarsi (non obbligatori ma fortemente consigliati) di guanti in plastica e sacchetti in plastica per i residui della resina, che vanno smaltiti con qualche accortezza.

LA PRIMA STAMPA

Attrezzati di tutto punto come "il piccolo chimico" e dopo aver letto le varie istruzioni, ci apprestiamo a testare la stampante facendole effettuare una stampa di prova partendo da un file di esempio.

Condizione necessaria per stampare è utilizzare l'apposito programma "PreForm" scaricabile dal sito di formlabs; il software è attualmente disponibile per Windows (XP o superiore) e Mac OSX (10.6.8 o superiore), escludendo quindi, per il momento, il mondo Linux (in Internet abbiamo comunque trovato "tracce" di esperimenti fatti con i vari emulatori).

Il primo esperimento di installazione su una macchina XP fallisce, forse per la scarsità di prestazioni della scheda video

integrata, mentre una seconda installazione con Windows 7 su una macchina meglio equipaggiata ci dà finalmente accesso al programma.

Il software, che è proprietario e completamente "chiuso", permette di impostare pochi parametri, come la risoluzione di stampa (100, 50 o 25 micron) ma anche di ruotare, scalare e duplicare il pezzo; lavora con i classici file STL provenienti dai vari repository/editors online, quali ad esempio Thingiverse o Tinkercad. Scegliamo di stampare, come primo pezzo, una piccola copia del "vica illusion sculpture" (<http://www.thingiverse.com/thing:5700>) che permette di saggiare, a nostro avviso, il comportamento di una stampante con i pieni e i vuoti, oltre al fatto che questa figura rappresenta un nodo intrecciato dall'aspetto davvero difficile da riprodurre.

Importiamo il nostro STL, impostiamo il colore della resina (clear, nel nostro caso) e posizioniamo il pezzo al centro dell'area di stampa; il software calcola automaticamente la quantità di resina necessaria e il tempo stimato per la stampa, che ovviamente varia in



Il kit per maneggiare i materiali necessari alla stampa.



Flacone contenente il liquido polimerico da versare nella macchina per costruire gli oggetti; il liquido è fotosensibile e va quindi protetto dalla luce.

Il fenomeno Kickstarter

Chi ha una buona idea ma non può realizzarla per mancanza di fondi propri o per l'impossibilità di trovare un finanziatore tra banche, finanziarie, Angel Investors, ecc., da qualche tempo può contare sul Crowdfunding. Il meccanismo è abbastanza semplice: il progetto viene presentato sul sito, completo di specifiche, costi, benefici e requisiti. Si pone un obiettivo temporale per la raccolta fondi e si fissa la cifra desiderata (o meglio, calcolata per le necessità del progetto). Sarà poi la promozione e la "bontà" del progetto a decidere se il progetto ha successo o meno. I contributori (bakers) hanno diverse possibilità di partecipare alla raccolta fondi. Di solito ci sono donazioni di vari livelli, che vanno dai pochi dollari, in cambio di un adesivo o menzioni fino a diverse migliaia di dollari per avere il prodotto in anteprima o altro. Per fare diventare realtà Form1, i suoi ideatori si sono affidati a Kickstarter,

la piattaforma web di Crowdfunding per eccellenza. Il progetto è stato presentato il 26 Settembre 2012, con un mese di tempo per raccogliere la cifra di 100.000 Dollari. Il successo della campagna è stato tale che il 26 Ottobre la campagna si è conclusa con la raccolta di ben 2.945.885 dollari, ben al di sopra di ogni aspettativa (dati forniti dalla pagina web www.kickstarter.com/projects/formlabs/form-1-affordable-professional-3d-printer). Va inoltre notato che parecchie donazioni sono state fatte per i tagli più alti, ovvero quelli che prevedevano in cambio un esemplare della stampante. In passato, altre startup tecnologiche si sono avvalse con successo (o insuccesso) del Crowdfunding (tramite portali come Indiegogo, Kickstarter e altri) e non solo per progetti tecnologici: questi portali hanno finanziato anche libri, film, dischi e iniziative a carattere umanitario.

base alla grandezza e alla risoluzione richiesta (da 25 a 100 micron il tempo può variare anche di quattro volte). In questo caso la complessità dell'oggetto influisce poco con i tempi di stampa. Se avete avuto modo di lavorare con altre stampanti 3D e altri strumenti di slicing (ad esempio Slic3r) avrete notato che i parametri su cui giocare sono parecchi e permettono di fare una stampa con riempimenti di tipo diverso, oltre che di poter giocare con velocità per gli strati bassi, alti, ecc. In questo caso il pezzo viene "affettato" in tanti layer, che verranno poi sinterizzati dal laser, in modo pressoché uguale dall'uno all'altro, senza la possibilità di variare i parametri. Non obbligatorio, ma fortemente consigliato, è l'uso del supporto di stampa, ovvero uno strato aggiuntivo che fa da zoccolo al pezzo vero e proprio; il supporto viene generato direttamente dal software, mediante un'apposita sezione, configurabile, e consiste in una serie di piedini terminanti con una sottilissima punta, che

fanno da supporto, appunto, al pezzo. Nel caso voleste stampare un pezzo "aereo", come una figura rappresentante un nastro, o figure complesse con parti sospese o prominenti, potete avvalervi di questo strumento, che permette di generare, appunto, i vari piedini di supporto, rimovibili poi in fase di finitura. Formlabs suggerisce, inoltre, di posizionare i pezzi in modo da non intrappolare la resina e farla defluire nella vaschetta. La sezione di generazione permette di impostare densità e spessore dei supporti e della base, la grandezza della punta che terminerà sull'oggetto e altri piccoli dettagli sicuramente adatti a chi ne fa un uso molto avanzato. Anche nel caso della nostra stampa di prova abbiamo ritenuto opportuna la generazione del supporto, perché altrimenti, in fase di finitura, per rimuovere l'oggetto avremmo dovuto letteralmente "scalpellarne" la base con l'apposita spatola in dotazione. Una volta generato, possiamo esportare il pezzo completo di

supporto in un file dal formato proprietario (.FORM) in modo da poterlo replicare in altre occasioni in cui servisse. Il file è "chiuso" in tutti i sensi: è criptato e non si conoscono, al momento, altre stampanti o programmi in grado di rielaborarlo o riaprirlo. Sarà quindi possibile solo averne un backup o eventualmente scambiarlo con un altro proprietario di Form1.

Veniamo alla fase di stampa vera e propria: una volta ridimensionato il pezzo, ma solo per guadagnare tempo (siamo davvero curiosi di vedere cosa ne verrà fuori) passiamo alla generazione dei supporti, usando i parametri standard; l'unica opzione che andiamo a modificare è "use internal support", ovvero togliamo l'opzione di generazione per le parti che il software considera interne (immaginate di voler stampare una scatola aperta rovesciata: il "tetto" avrà bisogno di qualche supporto per essere stampato correttamente). Nel nostro caso sembrava veramente un supporto inutile, dato che avevamo già provato a stampare lo stesso pezzo con la 3Drag senza alcun problema.

Una volta generati i supporti, si passa alla fase di stampa vera e propria: il software calcola la quantità di resina necessaria, in millilitri, per la completa realizzazione del pezzo, ma, purtroppo, non abbiamo indicazione della quantità né sulla vaschetta (che presenta una linea di minimo/massimo) né sulla bottiglia di resina. Per la prima stampa andiamo quindi "ad occhio", ripromettendoci di analizzare meglio il tutto per le successive. Una volta riempita la vaschetta possiamo finalmente lanciare la stampa: il programma, in questo caso, "compila" il codice necessario alla stampa e procede



La macchina prima della stampa: la vaschetta contiene il liquido polimerico.



Il pezzo in "crescita" all'interno della stampante.

all'upload verso la stampante; contemporaneamente il display della Form1 ci chiede se abbiamo effettuato tutte le operazioni preliminari (supporti, vaschetta, ecc) e chiede conferma della stampa con la pressione del pulsante fisico presente sulla stampante. A seconda della dimensione del file generato, l'upload può avere tempi diversi, ma mai lunghissimi: si parla di un paio di minuti al massimo. Nel frattempo, comunque, la Form1 inizierà a stampare, partendo dai layer più bassi. Una volta completato l'upload, e questo è sicuramente uno dei vantaggi della nostra Form1, è possibile avviare la stampa anche se stacciamo il cavo USB (o spegniamo il PC), dato che la Form1 dispone di uno spooler di stampa interno; quindi non è necessario che ci sia una macchina in funzione attaccata in modo permanente alla stampante. Dal tasto multifunzione presente sul frontale è possibile, mediante un singolo tocco, mettere in pausa la stampa, o, dopo una pressione lunga, annullarla del tutto.

Una volta avviata, la stampante compie un paio di operazioni di allineamento: il supporto calerà dall'alto, mediante la sua guida di precisione, fino ad immergersi nella vaschetta. A questo punto il

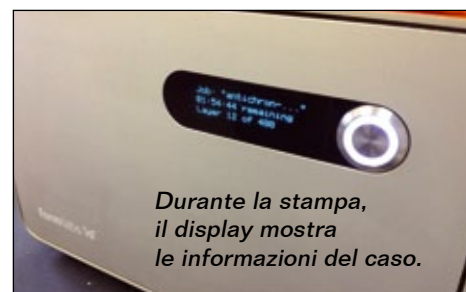
sistema laser all'interno comincerà ad "asciugare" strati successivi di resina, alternando una scansione del raggio ad un movimento oscillatorio della vaschetta. Il processo, in sé, al di là dei primi attimi di curiosità è anche abbastanza noioso e non permette di intervenire in alcun modo (il coperchio arancione prevede una sicurezza che interrompe la stampa in caso di apertura). Durante le prime fasi, inoltre, il supporto è totalmente immerso nella vaschetta e il pezzo è praticamente invisibile.

Le tempistiche previste dalla stampante vengono comunque rispettate, sul display abbiamo l'indicazione del tempo residuo e del numero di layer stampati sul totale.

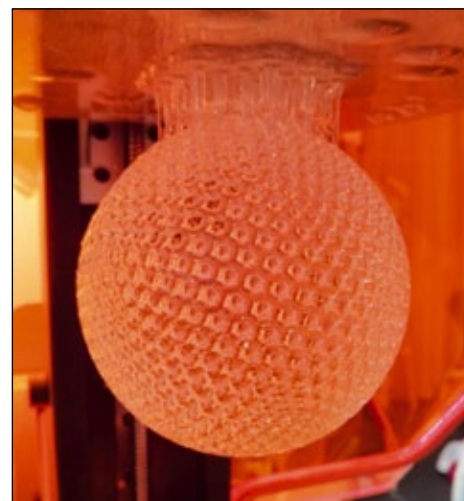
Non ci resta che aspettare, dunque, la fine. Durante il processo, in ogni caso, abbiamo avuto modo di controllare lo stato della resina e di effettuare un rabbocco, interrompendo e riprendendo il lavoro, senza influire sul risultato finale.

Al termine della stampa, il piano di appoggio si posizionerà all'estremità superiore, in attesa di essere rimosso, per la successiva fase di finitura; il supporto, infatti, è dotato di sistema meccanico di blocco/sblocco a leva, che permette di rimuovere

agevolmente il tutto, una volta stampato, per posizionarlo poi sul banco di finishing. Una volta posizionato il blocco sul banchetto, occorre agire con la spatola, in modo piuttosto energico, per rimuovere la stampa e i supporti; dopo l'iniziale titubanza, abbiamo scoperto che il metodo più rapido consiste nel fare leva alla base del supporto, su vari angoli, in modo da permettere un distacco parziale dalla base, e poi infilare la spatola e fare leva per staccare il resto in un sol colpo. Durante questa operazione è sempre consigliato un paio di guanti, anche quelli di tipo usa e getta, per poter manipolare pezzo e spatola senza sporcarsi di resina appiccicosa. Una volta staccato, è necessario lavare la stampa in alcol isopropilico; a questo proposito, FormLabs ha predisposto una



Durante la stampa, il display mostra le informazioni del caso.



I supporti creati dal programma per sostenere il pezzo durante la stampa.



vaschetta rettangolare, in plastica trasparente, alta e stretta, che dovrebbe contenere il pezzo più grande stampabile. All'interno della vaschetta abbiamo una piccola griglia, sempre in plastica, estraibile tramite due comode maniglie; una volta inserita, la griglia rimane perfettamente allineata con il fondo del contenitore, permettendo quindi di estrarre il pezzo dopo il lavaggio, con un semplice movimento, senza rovesciare una goccia di alcol. Formlabs suggerisce di lasciare in ammollo il tutto per circa 15 minuti (un tempo da mettere in conto insieme a quello della

stampa) e di rimuovere i supporti dopo questo lavaggio. Dopo aver seguito scrupolosamente la procedura, ci apprestiamo a rimuovere i supporti, sempre con la spatola, e constatiamo che, effettivamente, si rimuovono agevolmente senza lasciare tracce visibili sul pezzo stampato; dopodiché, forse anche perché abbiamo poca esperienza e forse abbiamo lasciato il pezzo a bagno per poco tempo e risulta abbastanza "appiccicoso", lo sottoponiamo ad un secondo lavaggio necessario a rimuovere la resina in eccesso. Ora il pezzo è stampato!

La risoluzione è decisamente apprezzabile, anche per una stampa "di prova" come quella che abbiamo fatto; la resina semitrasparente (Clear) non lascia trasparire tracce di sovrapposizioni, o scalini, come nella stampa con la tecnica ad estrusione. In compenso, la consistenza del pezzo lascia un po' a desiderare e si avverte una certa "gommosità" e la superficie è scalfibile con facilità. Però il pezzo ottenuto con la resina trasparente è molto bello: difficile ottenere la stessa qualità di finitura con la tecnica ad estrusione. La prima prova è conclusa e siamo abbastanza soddisfatti, ma

Cosa va e cosa non va

Proviamo a mettere sui piatti della bilancia pregi e difetti della stampante Form1; cosa ne uscirà?

PREGI

- 👍 Alta risoluzione, selezionabile a livello software in fase di pre stampa.
- 👍 Semplicità d'uso: una volta impostata, la stampante lavora in perfetta autonomia senza tempi di attesa per riscaldamento o azzeramenti.
- 👍 Server di stampa integrato: gestione facilitata da un solo pulsante multifunzione e da un display su cui compaiono tutte le informazioni necessarie.
- 👍 Design innovativo e rifiniture ben curate.

DIFETTI

- 👎 Software di stampa completamente proprietario e chiuso. A differenza di altre stampanti, non è possibile utilizzare altro software se non quello della FormLabs, pertanto i file generati sono chiusi e non interpretabili o importabili da altri sistemi di stampa (a differenza, ad esempio, dei G-Code).
- 👎 La meccanica si è rivelata fragile, almeno per quanto riguarda la gestione della vaschetta per la resina.
- 👎 L'impiego della resina comporta l'utilizzo pressoché obbligato di guanti, e precauzioni, il rischio di sporcare parti vitali della stampante (come, ad esempio, lo specchio interno) è sempre in agguato. Una apposita sezione delle FAQ è dedicata a questi casi.
- 👎 La fase di finitura è pressoché obbligatoria. Il pezzo stampato è sempre ricoperto di resina residua, e necessita di essere lavato e i supporti rimossi.
- 👎 Elevato costo unitario del pezzo, dovuto al costo della resina.

FORM1 VS 3Drag

Sebbene, visto il differente tipo di tecnologia, non sia possibile fare un confronto diretto con la 3Drag, possiamo comunque abbozzare una comparazione sostanziale sui risultati. Ebbene, La Form1 ci ha sorpreso in quanto a risoluzione e facilità d'uso, non ha tempi di attesa per riscaldamento o altri tempi morti. Di contro è ancora parecchio costosa (quasi il prezzo di cinque stampanti "3Drag") e la resina ha un prezzo di circa 150 dollari al

litro: decisamente tanto, rispetto ai pochi euro/kg di PLA e ABS. Una volta stampato, il pezzo va obbligatoriamente rifinito, tramite bagni di alcool, per poter rimuovere i supporti; questo obbliga a procurarsi un minimo di attrezzatura e spazi di lavoro. La stampante ha anche qualche difettuccio, come già accennato, che probabilmente verrà tolto nelle prossime versioni.

C'è da considerare anche il fattore "chiusura" della strategia di FormLabs, ovvero il fatto che sia il software che l'hardware sono totalmente chiusi e non modificabili (almeno dal punto di vista legale).

È una scelta che si può condividere o meno e che se da un lato può garantire stabilità e uniformità, dall'altro preclude a sviluppatori, maker e "smanettoni" in genere la possibilità di apportare migliorie ed evoluzioni. Il software, tra l'altro, può solo importare file STL (leggibile da programmi sia commerciali che Open Source) e salvare in formato .FORM proprietario e leggibile solo da FormLabs. Progetti come la 3Drag, al contrario, si basano su logiche diverse, aperte, che permettono un'ampia possibilità di evoluzioni da parte degli esperti o dei semplici maker. Proprio la 3Drag, può essere trasformata in CNC e quindi consente, con una minima modifica (basta smontare la testa di stampa e sostituirla con una staf-



fa che porterà un elettromandrino), di avere due macchine con un minimo esborso in più, mentre la Form1 può fare solo quello. Il costo del materiale, come detto, è in discesa libera, vista la sua crescente diffusione; inoltre ogni giorno vengono presentati nuovi colori, nuovi materiali, non da ultimo un filo color legno, che danno spazio alle idee e ai progetti più nuovi e interessanti. Dal lato software, la 3Drag può essere pilotata dai vari software Open Source presenti in rete, da RepetierHost, Pronterface/Printrun e perfino da Cura, un programma nato per una stampante "concorrente", in piena logica OpenSource, appunto. Questi software non solo importano i file STL, ma possono anche esportare in formati compatibili e liberi, in modo che possano essere scambiati tra dispositivi diversi.



pensiamo subito a come testare la Form1 con pezzi più complessi, per valutarne le prestazioni. Molte luci e qualche ombra. Durante le nostre prove abbiamo provato a lanciare diverse stampe, con complessità e durate differenti, ed è emerso un difetto "congenito" della stampante: come già accennato, la stampante effettua dei cicli di lavorazione abbastanza precisi (vengono alternati laser e movimenti della vaschetta) che servono a distribuire in modo omogeneo la resina. Il movimento della vaschetta è affidato ad un meccanismo piuttosto semplice, ma dalla logica contorta, ovvero, la vaschetta è ancorata ad un supporto metallico, che la sostiene, incernierato per un lato al telaio principale, e dall'altro lato sostenuto da un perno, collegato ad una vite senza fine, guidata da uno stepper motor. Quest'ultimo compie dei cicli di avanti e indie-

tro, che fanno alzare e abbassare la vaschetta; non esistendo alcun sensore o finecorsa, il movimento è affidato alla precisione del motore e della sua elettronica. Durante uno dei nostri test, la vite senza fine azionata dal motore è andata a crollare verso il fondo, svitandosi dal supporto della vaschetta; ciò ha causato il disallineamento della vaschetta, necessario per la corretta stampa, di fatto impedendo che il laser sinterizzasse correttamente la resina. Purtroppo, non essendoci alcun sensore, la stampante ha continuato a "credere" di essere allineata e non ha interrotto la stampa; di conseguenza il pezzo in stampa è risultato interrotto, la resina si è asciugata sulla superficie della vaschetta e il motore si è danneggiato. Durante la riparazione abbiamo colto l'occasione per smontare e analizzare la stampante al suo interno, così

da poterla osservare e analizzare nella sua meccanica. Dopo un breve consulto abbiamo dedotto che un finecorsa, o meglio, una modifica del meccanismo di movimento basato su un sistema a camme, potrebbe decisamente aumentare l'affidabilità della stampante.

CONCLUSIONI

A parte il difetto che abbiamo riscontrato, la Form1 è una buona stampante, adatta ad un pubblico esigente e professionale, disposto a spendere qualcosa in più per avere stampe di qualità superiore a quelle ottenibili con le stampanti FDU (ad estrusione). Punti forti sono la risoluzione e la qualità; di contro il costo del materiale è ancora abbastanza alto, la velocità è accettabile, ma i tempi di realizzazione aumentano in proporzione lineare alla risoluzione scelta.

L'utilizzo di tecnologie avanzate allo stato solido per i fari anteriori migliora la sicurezza dei veicoli.



PIXEL LIGHT SISTEMA DI CONTROLLO SMART PER FARI A LED

di PAUL DECLOEDT

S secondo una ricerca condotta per conto della Commissione Europea, la probabilità che si verifichino incidenti gravi nelle ore serali e notturne aumenta del 60% rispetto alle ore diurne. Per contrastare queste cifre allarmanti, le Case automobilistiche stanno prendendo in considerazione dispositivi di illuminazione molto più sofisticati e intelligenti. Questo articolo analizza nel dettaglio la tecnologia emergente *Pixel Light* ed esamina le relative tecniche di controllo oggi disponibili, prima di tentare di prevedere quale di queste tipologie di

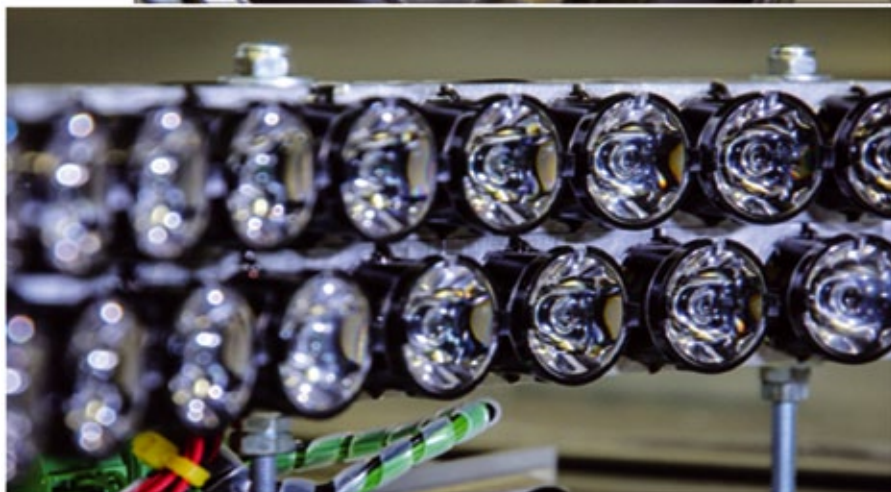
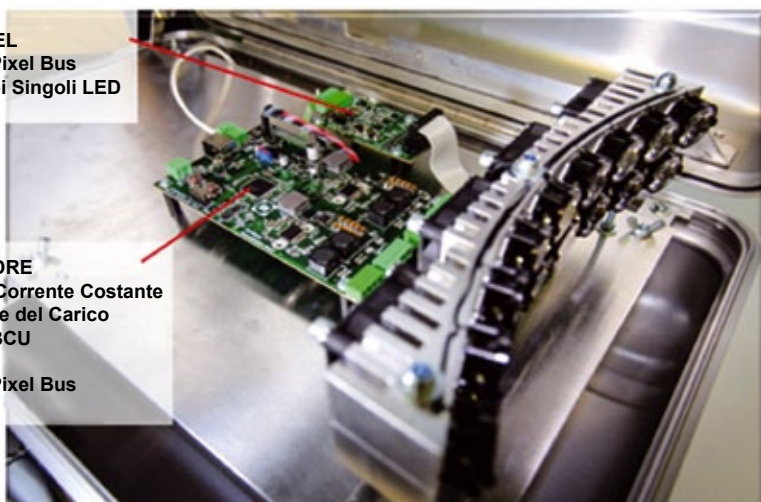
controllo avrà la maggiore diffusione nella prossima generazione di automobili. Nell'ultimo decennio, abbiamo assistito a una graduale evoluzione dell'impianto di illuminazione dell'autoveicolo, passando dalle lampade ad incandescenza alle soluzioni alternative allo stato solido. Inizialmente, ciò riguardava solo le funzioni meno essenziali, come l'illuminazione dell'abitacolo, le luci di posizione e il quadro strumenti, ma da un po' di tempo i LED vengono utilizzati, in particolare, anche nei fari anteriori, sia per le auto di lusso che per quelle di fascia media. Inoltre, i fari anteriori a

SCHEDA PIXEL

- Interfaccia Pixel Bus
- Controllo dei Singoli LED

SCHEDA MADRE

- Sorgenti di Corrente Costante
- Distribuzione del Carico
- Interfaccia BCU
- Diagnostica
- Interfaccia Pixel Bus

**Fig. 1***Un sistema Pixel Light*

Grazie all'impiego di Pixel Light (un esempio è illustrato in **Fig. 1**), è oggi possibile generare il fascio di luce della forma richiesta per migliorare la visibilità di guida, con emettitori LED discreti che possono essere accesi, spenti o attenuati per creare un profilo di illuminazione che si adatti appieno alla specifiche necessità in ogni momento del viaggio. Rispetto ai sistemi AFS realizzati con le tradizionali lampade a incandescenza (basate sulla regolazione del fascio di luce tramite motore), il controllo di più emettitori LED con *Pixel Light* offre una strategia molto più efficace. Grazie all'accesso al sistema GPS del veicolo, è possibile regolare la luce emessa dagli emettitori per produrre un fascio che meglio illumini le curve della strada, inclinandolo opportunamente in presenza di cunette e dossi (come illustrato in **Fig. 2**). Inoltre, combinando questa tecnologia con sistemi di riconoscimento delle immagini, si possono apportare importanti miglioramenti alla visibilità di segnali stradali, pedoni e altri oggetti presenti nel campo visivo del conducente, mentre viene ridotto notevolmente l'abbagliamento a cui sono sottoposti i veicoli che sorraggiungono dalla direzione opposta. La **Fig. 3** illustra come un veicolo in avvicinamento può essere escluso dal fascio abbagliante, così da eliminare l'effetto "glare".

LED vengono ora anche proposti come optional nelle vetture economiche. Tra i fattori chiave di questa migrazione vi è il fatto che i dispositivi LED offrono una maggiore affidabilità e consumi inferiori. Ma è la versatilità dell'illuminazione a stato solido, con la possibilità di regolare le potenze dei singoli emettitori, il fattore che permetterà più di

ogni altra cosa di migliorare la sicurezza del veicolo. La nuova e innovativa tecnologia optoelettronica, altamente versatile, sembra destinata a rivoluzionare il modo in cui i sistemi di illuminazione frontale adattiva (AFS) vengono realizzati nelle autovetture, consentendo di ottenere importanti miglioramenti nella sicurezza stradale.

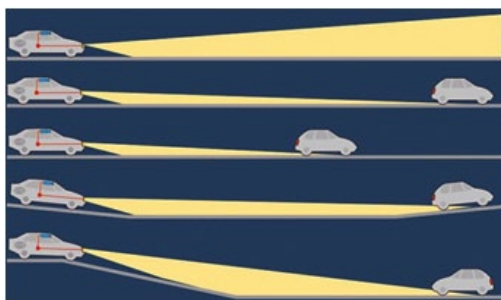
A.**B.****Fig. 2** - Inclinazione regolabile del fascio luminoso con Pixel Light: a) senza sistema AFS, b) con sistema AFS Pixel Light.



Fig. 3 - Confronto di sistemi AFS: a) con sistema anti-glare Pixel Light, b) senza sistema anti-glare Pixel Light.

Poiché gli AFS a LED con *Pixel Light* permettono l'inclinazione e la rotazione del fascio di illuminazione a livello dei singoli emettitori, essi, fondamentalmente, eliminano la necessità di un motore per la rotazione dell'impianto di illuminazione. *Pixel Light* consente anche di modellare il fascio di luce e, pertanto, anche per questa funzione non occorre più alcun motore. Analogamente, il sistema ACOL (Adaptive Cut Off Line), sebbene richieda un maggior numero di pixel nei fari anteriori, elimina anche la necessità di un motore che regoli l'inclinazione delle luci.

TOPOLOGIE DI PILOTAGGIO DEI SISTEMI AFS A PIXEL

Ci sono due configurazioni di base per il pilotaggio dei sistemi AFS *Pixel Light*.

A) Nei sistemi con topologia di pilotaggio parallela (PDT) occorre utilizzare, come alimentatore di corrente, un convertitore buck ad alta potenza. Poiché la topologia PDT utilizza correnti di alimentazione elevate, esiste una connessione per ciascun emettitore LED. Le variazioni nelle caratteristiche elettriche dei diversi emettitori incidono significativamente

sulle prestazioni complessive dell'impianto di illuminazione. Dato che potrebbero esserci notevoli differenze nelle tensioni tra anodo e catodo dei diversi emettitori, potrebbe essere necessario dissipare il calore prodotto in eccesso. Ciò aumenta il consumo di potenza e richiede sistemi di gestione termica più complessi (con un aggravio in termini di costi e di spazio occupato). È improbabile che questa configurazione venga utilizzata diffusamente nei sistemi AFS, a meno che non si riesca a migliorare l'adattamento tra i vari LED.

B) A differenza della topologia PDT in cui la corrente di alimentazione è costante per ciascun emettitore LED, nei sistemi con topologia di pilotaggio seriale (SDT) occorrono due connessioni per ogni stringa di LED e si utilizza un circuito per il pilotaggio delle stringhe al fine di garantire l'erogazione di una corrente costante a ciascuna stringa. I singoli emettitori nell'array di pixel vengono disattivati/attenuati tramite interruttori per modificare la forma del fascio. Se un nuovo chip viene

inserito nel circuito è possibile evitare, grazie al partizionamento del sistema, i problemi di consumo energetico e di surriscaldamento tipici delle topologie in parallelo. Poiché la topologia seriale prevede una corrente di alimentazione costante per ciascuna stringa, non è necessario l'adattamento delle correnti (a differenza di quanto avviene per la PDT), con chiari vantaggi in termini di costi.

A quanto pare, il pilotaggio SDT offre una topologia migliore con cui gestire i sistemi AFS con tecnologia *Pixel Light*. Ma se si vuole che questi sistemi diventino di uso comune nelle vetture che entreranno sul mercato nei prossimi anni, essi dovranno essere realizzati con un'elettronica modulare altamente efficace in termini di costi. Un esempio di circuito di pilotaggio che utilizza la topologia seriale con partizionamento del sistema è descritto in **Fig. 5**. In questo esempio, il chip NCV78763 di ON Semiconductor, un circuito buck-boost per il pilotaggio dei LED, fornisce la corrente di alimentazione al sistema. Questo dispositivo altamente integrato è supportato dalla combinazione di un con-

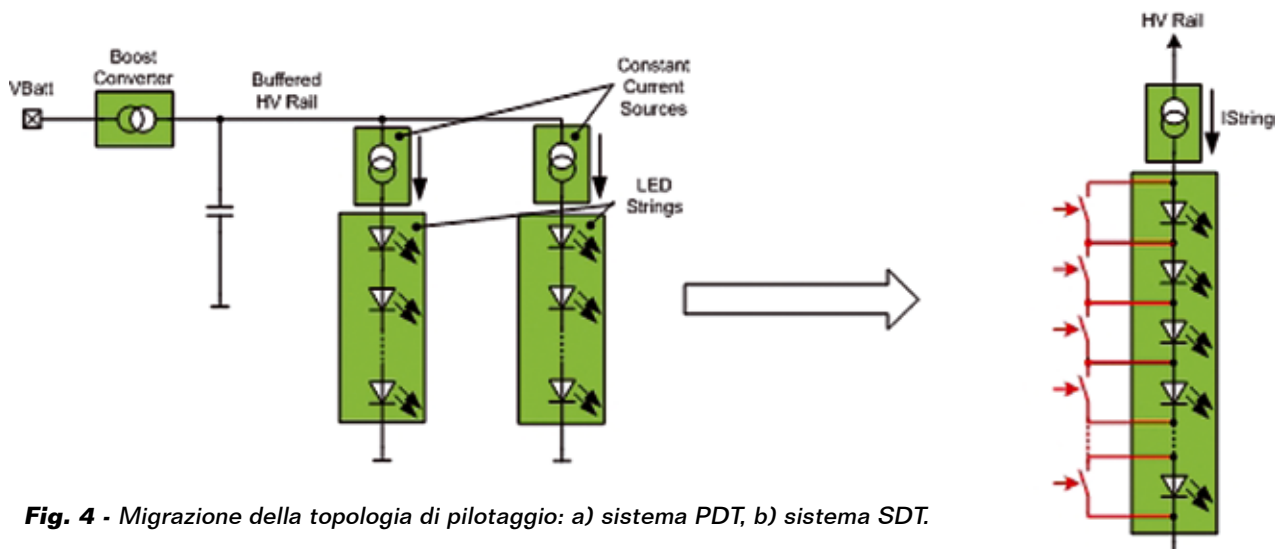


Fig. 4 - Migrazione della topologia di pilotaggio: a) sistema PDT, b) sistema SDT.

troller dei pixel con relativo chip associato. La natura modulare di questa soluzione si sposa perfettamente con la strategia di sviluppo dei prodotti generalmente attuata dalle Case automobilistiche e permette di abbassare i costi dei componenti, semplificare il processo di implementazione e sveltire la progettazione e la realizzazione del sistema.

In conclusione, le funzioni di alto livello e la commutazione veloce di *Pixel Light*, dovute alle migliori caratteristiche prestazionali di

questa tecnologia, offrono un potente strumento per migliorare la sicurezza stradale, anche nelle condizioni di guida più difficili. I fasci di luce possono assumere la forma più adatta al contesto e inclinati come richiesto, con un elevato grado di precisione e reattività. *Pixel Light* consente, inoltre, di creare più zone di ombra all'interno del fascio, nei punti dove serve. Tutto ciò può essere ottenuto senza l'uso di motori (nonché dei relativi circuiti di pilotaggio, ecc.). Di conseguenza, l'elettronica

dell'impianto di illuminazione anteriore può essere molto meno complessa e la spesa dei materiali associati sostanzialmente più bassa. La topologia SDT, grazie al suo intrinseco adattamento delle correnti, permetterà di ottenere un controllo semplice ed efficiente dei sistemi AFS con tecnologia *Pixel Light*. Inoltre, grazie anche all'impiego di semiconduttori ad alte prestazioni, può aver luogo quell'approccio modulare che le Case automobilistiche richiedono insistentemente. ■

a cura di Paul Decloedt, ON Semiconductor

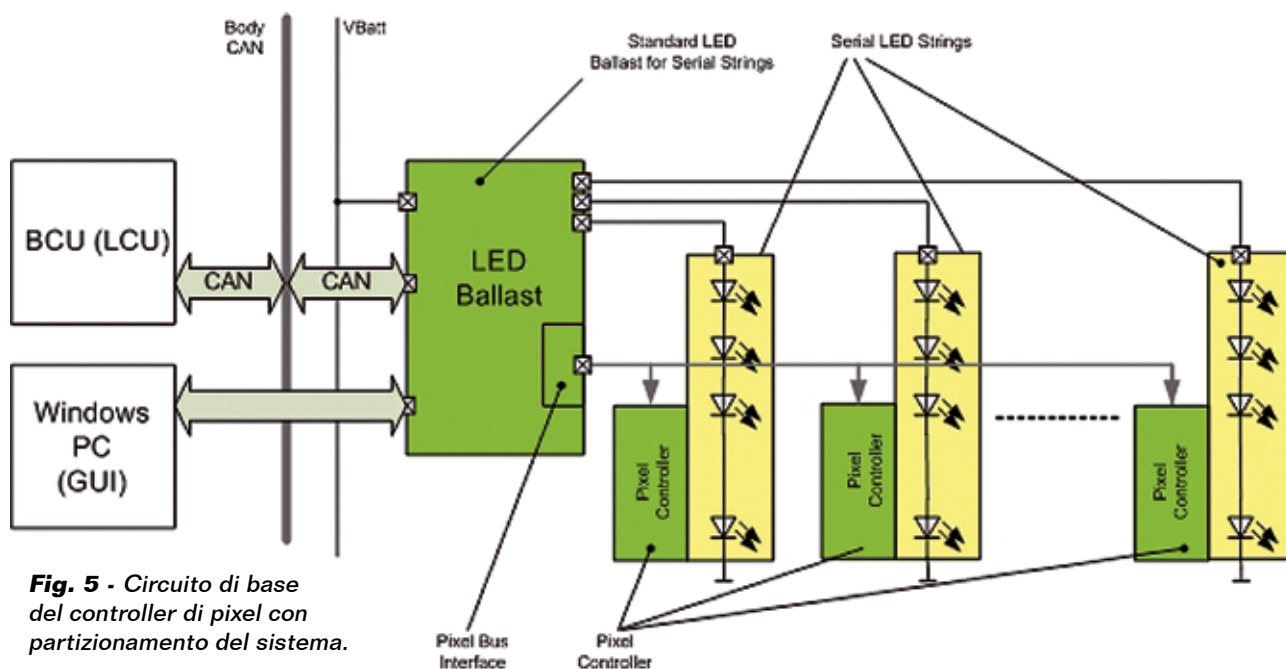
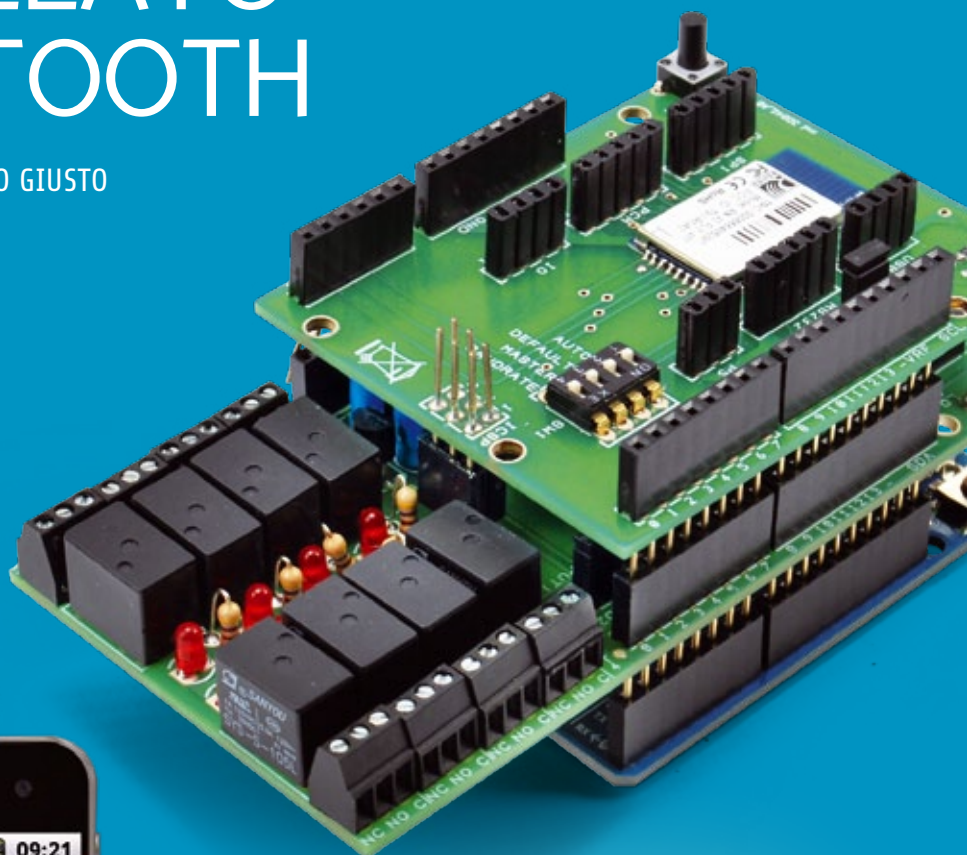


Fig. 5 - Circuito di base del controller di pixel con partizionamento del sistema.

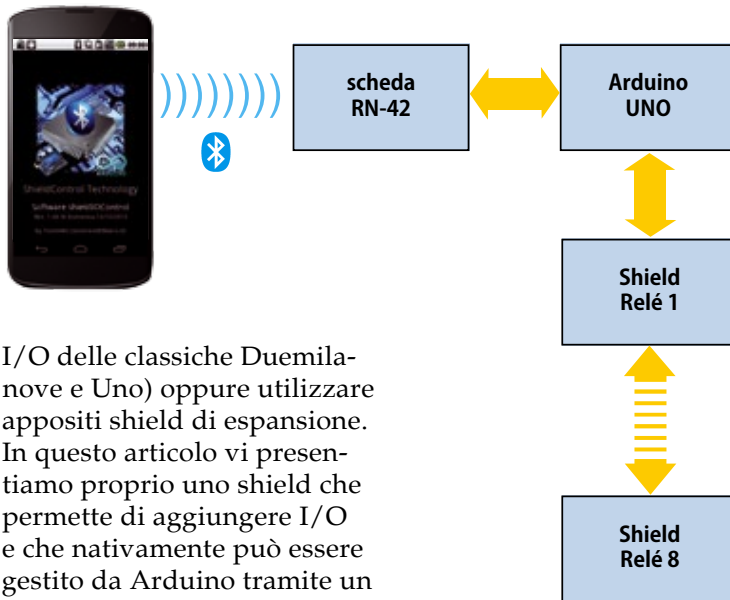
I/O EXPANDER I²C CONTROLLATO DA BLUETOOTH

degli ingg. TOMMASO E ALESSANDRO GIUSTO

Espandiamo le risorse digitali di Arduino con uno shield I²C bus e controlliamole tramite collegamento bluetooth attraverso uno smartphone Android.



Tutte le schede Arduino dispongono di un certo numero di contatti di ingresso e uscita digitali. Per i progetti più semplici, tali risorse sono più che sufficienti, però in quelli dove servono parecchie linee di controllo, la dotazione "di serie" può non bastare, anche perché alcuni pin sono condivisi con risorse interne di Arduino oppure vengono dedicati alla gestione di shield esterni. In questi casi una possibile soluzione è passare a versioni più potenti di Arduino (per esempio la Mega, che dispone di più



I/O delle classiche Duemila-nove e Uno) oppure utilizzare appositi shield di espansione. In questo articolo vi presentiamo proprio uno shield che permette di aggiungere I/O e che nativamente può essere gestito da Arduino tramite un canale dati a bus I²C; lo shield è un I/O expander generico in grado di aggiungere otto uscite a relé ed altrettanti ingressi a livello TTL.

Ognuno di voi può farne l'uso che ritiene più opportuno, tuttavia l'applicazione di queste pagine si propone non solo di aggiungere I/O ad Arduino, ma anche di permetterne il controllo in wireless mediante un collegamento Bluetooth, quindi tramite uno smartphone Android nel quale verrà installata un'apposita App che descriveremo più avanti.

Cosa ancor più interessante, è che la struttura proposta è modulare, nel senso che su Arduino si possono montare più shield, fino ad ottenere un massimo di 64 input e 64 uscite a relé.

LO SHIELD I/O EXPANDER

A basso livello, la gestione del componente avviene tramite bus di comunicazione I²C, implementata, nello shield, dall'integrato MCP23017 di Microchip.

Come molti di voi sanno, il bus I²C è un canale di comunicazione standard utilizzato per l'interconnessione a breve distanza tra un'unità Master e più Slave;

tipicamente il Master è una MCU, mentre le unità Slave sono periferiche di vario genere che non richiedono microprocessori o microcontrollori, ma normali interfacce I²C-Bus come il componente Microchip da noi usato o il PCF8574 Philips/NXP.

Il bus è composto da tre fili, di cui uno è la massa comune e gli altri due sono SCL (clock che scandisce la comunicazione) e SDA (linea dati seriale bidirezionale); il protocollo I²C-Bus prevede l'inizio di ogni sessione di comunicazione da parte della sola unità Master (le unità Slave si limitano a rispondere all'interrogazione della Master), la quale invia comandi composti da 8 byte, uno dei quali contiene l'indirizzo della Slave cui è diretto. Essendo, l'address, espresso mediante un byte, si può dire che l'I²C-Bus supporti 128 periferiche (in realtà solo 112, perché 16 address sono riservati).

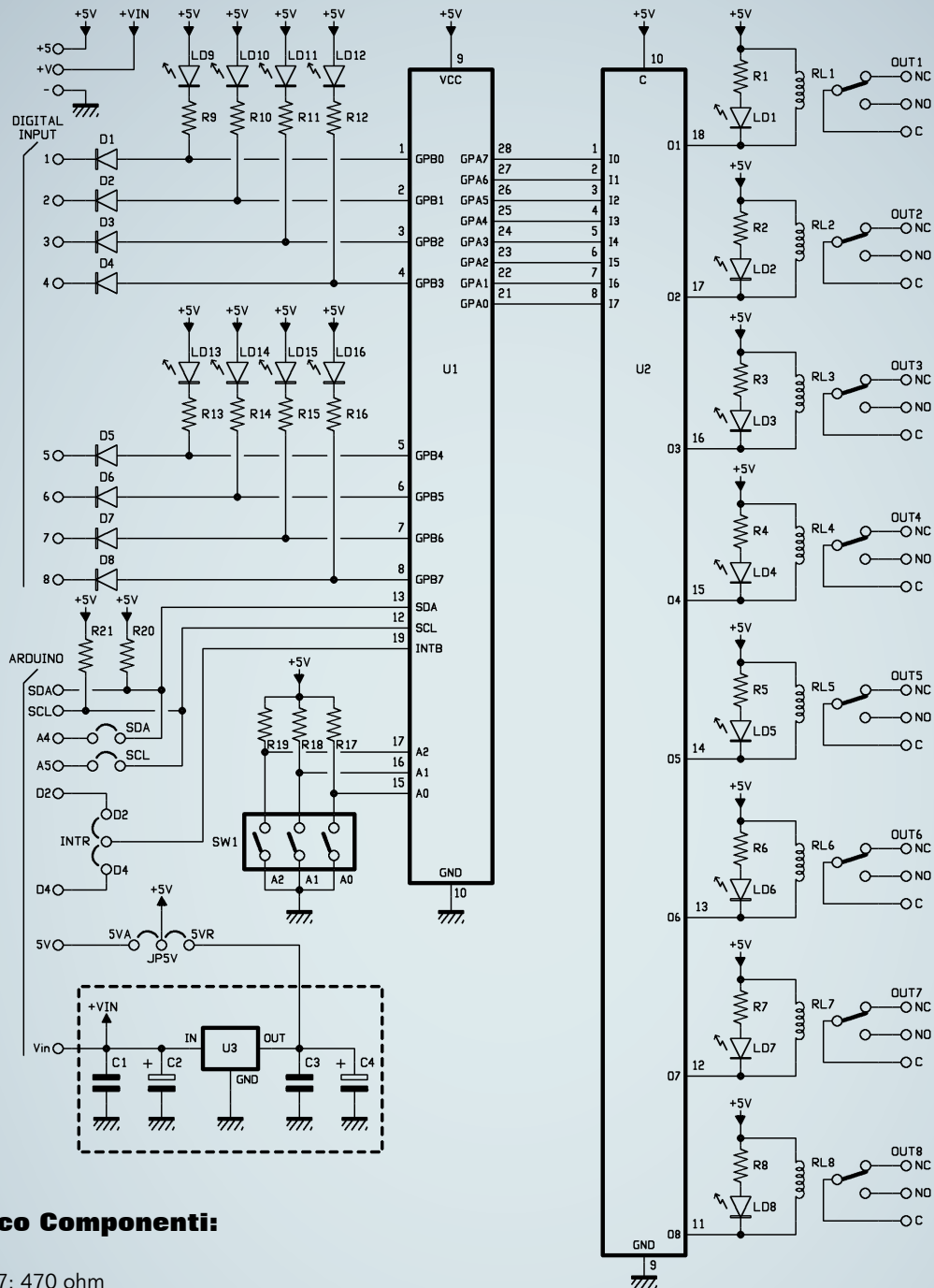
Solitamente l'identificazione delle singole unità Slave si realizza impostando l'opportuna combinazione logica sulle apposite linee di cui ogni integrato è provvisto; la gran parte

degli integrati ha solo tre linee, che consentono 8 combinazioni. Nel nostro caso, la possibilità di definire la destinazione dei comandi su I²C-Bus ci permette di collegare contemporaneamente (uno sopra l'altro) più shield I²C I/O Expander; siccome ogni scheda di espansione dispone di 8 uscite a relé ed altrettanti ingressi digitali, è possibile ottenere un totale di 64 uscite a relé e 64 input digitali.

Al sistema, realizzabile dotando Arduino di un certo numero di shield I/O Expander, abbiamo deciso di abbinare il controllo da Bluetooth, in modo da gestire da smartphone, per esempio, un certo numero di utilizzatori e di ingressi digitali. Per la comunicazione wireless abbiamo aggiunto uno shield Bluetooth RN-42, mentre lato Android abbiamo sviluppato un'apposita App che permette di visualizzare a display lo stato di tutte le risorse digitali ed inviare dei comandi di attivazione delle uscite.

Essendo il sistema modulare (cioè possono essere aggiunti o rimossi shield I²C I/O Expander a piacimento) e versatile, sia il firmware Arduino che la App sono stati pensati e realizzati in modo che si auto-adattino alla situazione applicativa. In particolare, il firmware Arduino è in grado di rilevare quanti e quali shield Expander sono presenti; la App riceverà tali informazioni da Arduino e di conseguenza adatterà la propria interfaccia grafica.

La App dispone di una pagina di configurazione in cui è possibile impostare le etichette di testo da utilizzare per la descrizione delle risorse I/O, permettendo così di adattarla alla configurazione hardware scelta. Infine, è presente una pagina



Elenco Componenti:

R1÷R7: 470 ohm
R8÷R16: 470 ohm
R17÷R21: 4,7 kohm
C1, C2: 100 nF multistrato
C3: 100 µF 16 VL elettrolitico
C4: 100 µF 16 VL elettrolitico
D1÷D8: 1N4148
LD1÷LD8: LED 3 mm rosso
LD9÷LD16: LED 3 mm verde
SW1: Dip-Switch 3 vie
RL1÷RL8: Relé miniatura 5V

U1: MCP23017-E/SP
U2: ULN2803
U3: 78L05

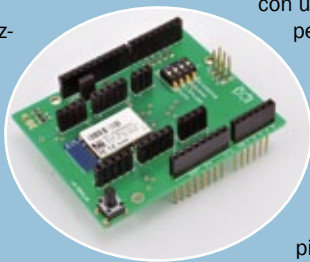
Varie:

- Zoccolo 14+14
- Zoccolo 9+9
- Strip maschio/femmina 3 poli (2 pz.)

- Strip maschio/femmina 6 poli (1 pz.)
- Strip maschio/femmina 8 poli (2 pz.)
- Strip maschio/femmina 10 poli (1 pz.)
- Strip maschio 3 poli (2 pz.)
- Jumper (2 pz.)
- Morsetto 2 poli passo 2,54 mm (6 pz.)
- Morsetto 3 poli passo 2,54 mm (8 pz.)
- Circuito stampato

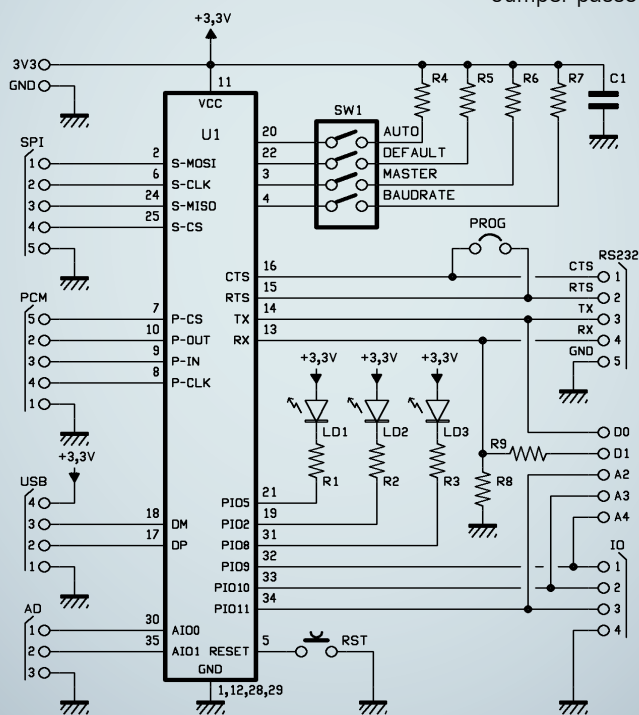
Il link Bluetooth

Arduino realizza la connessione Bluetooth con lo smartphone grazie ad un apposito shield già descritto nella 4^a puntata del Corso Android, pubblicata nel fascicolo n° 172 di Elettronica In (cui rimandiamo) e commercializzato dalla Futura Elettronica (www.futurashop.it). Questo shield si basa sul modulo RN-42 della Roving Networks (www.rovingnetworks.com) il cui funzionamento può essere visto dall'utente come un collegamento seriale; ciò, grazie al fatto che lo stack Bluetooth è completamente incapsulato, quindi utilizzando, ad esempio, un programma di emulazione di terminale (il classico Hyper Terminal o PuTTY), l'unità ricevente restituisce ogni volta il carattere corrispondente al tasto premuto sulla tastiera. Il modulo è compatibile Bluetooth 2.0 ed è di Classe 2 per quanto riguarda la potenza in trasmissione (garantisce una copertura di circa 20 metri). Il componente supporta i profili bluetooth SPP e HID e dialoga



con l'esterno mediante un'interfaccia UART con un data-rate fino a 3 Mbps; sul link Bluetooth, la velocità massima è di 721÷2.000 kbps. Il modulo funziona con una tensione di 3,3 V e perciò può interfacciarsi con le linee di I/O di Arduino Uno, che funzionano con detti livelli di tensione. Nel riquadro sottostante vedete lo schema elettrico e la lista componenti dello shield. I piedini dell'RN-42 sono tutti collegati ai pin-strip dello shield eccetto il 5 (RESET; reset hardware del modulo, qui gestito da un pulsante posto sullo shield stesso) e il 23 (SPI_CS; pin CSB dell'interfaccia SPI, utilizzato solo per la programmazione dell'RN-42). Nella nostra applicazione, Arduino Uno gestisce la comunicazione Bluetooth interfacciandosi al modulo RN-42 mediante l'UART; la connessione SPI di cui il modulo RN-42 dispone, è riservata alla programmazione dello stesso e quindi non è gestita da Arduino.

- | | | |
|----------------------------|------------------------|-------------------------------|
| R1 ÷ R3: 220 ohm (0805) | LD1: LED rosso (0805) | - Strip M/F 8 vie (2 pz.) |
| R4 ÷ R7: 1 kohm (0805) | LD2: LED verde (0805) | - Strip M/F 10 vie |
| R8: 20 kohm (0805) | LD3: LED giallo (0805) | - Strip Maschio 2 vie |
| R9: 10 kohm (0805) | SW1: Dip-Switch 4 vie | - Strip Femmina 3 vie |
| C1: 100 nF ceramico (0805) | | - Strip Femmina 4 vie (2 pz.) |
| U1: RN-42 | Varie: | - Strip M/F 3 vie (2 pz.) |
| RST: Microswitch | | - Strip M/F 6 vie (3 pz.) |
| | | - Jumper passo 2,54 |



di configurazione Bluetooth in cui è possibile programmare il nome e il pin code da abbinare allo shield RN-42.

SCHEMA ELETTRICO SHIELD I/O EXPANDER

Lo shield è già stato utilizzato e descritto nel fascicolo n° 181, perciò ne diamo una descrizione abbastanza sommaria, rimandando, per gli approfondimenti, all'articolo MusicXmas pubblicato nel predetto fascicolo n° 181.

Lo shield è sviluppato intorno al componente U1, ossia l'I/O Expander MCP23017 prodotto da Microchip; questo dispone in totale di 16 pin (suddivisi a livello logico in 2 diverse porte A e B) configurabili singolarmente come ingresso o uscita. Nel nostro schema abbiamo deciso di dedicare tutti gli 8 pin della porta A come uscite e gli 8 pin della porta B come ingressi. Le uscite finiscono sull'integrato U2 (ULN2803) che è un componente progettato appositamente per pilotare carichi induttivi come le bobine dei relé (infatti al suo interno dispone di diversi stadi a transistor dotati di diodi di protezione della giunzione base-collettore). Gli ingressi (porta B) sono invece collegati ad apposite sezioni circuitali che realizzano gli input digitali. Ogni ingresso dispone di un diodo di protezione, una resistenza di pull-up e un LED che ne segnala lo stato. Quando l'ingresso è logicamente basso (a massa) il relativo LED risulta acceso, mentre l'I/O Expander segnala che l'ingresso è a "0"; viceversa, quando l'ingresso è a livello alto, il relativo LED è spento e l'I/O Expander indicherà che l'ingresso è "1".

La gestione dell'MCP23017 av-

viene da I²C Bus, realizzato con i piedini 12 (SCL) e 13 (SDA). i quali sono collegati ai relativi pin della scheda Arduino. L'MCP23017 offre la possibilità di generare un interrupt a fronte della variazione di almeno un pin programmato come input; nel nostro caso abbiamo sfruttato questa funzione per gestire da Arduino gli input in modalità interrupt e non polling (risultando quindi più veloce e senza appesantire il bus I²C).

L'interrupt permette di avviare una connessione, cioè l'interrogazione da parte dell'unità Master, che nel nostro caso è Arduino. Chiaramente, come vedremo più avanti, per sfruttare questa caratteristica è necessario che anche il software venga appositamente realizzato. Abbiamo lasciato (tramite jumper) la possibilità di portare il pin di interrupt dell'MCP23017 (INTB, pin 19) in ingresso su 2 differenti pin di Arduino (D2 o D4) in modo da adattarlo il più possibile al reale utilizzo. L'indirizzo I²C dell'MCP23017 viene selezionato tramite il dip-switch SW1.

Lo shield si alimenta tramite il Vin di Arduino, grazie ad un regolatore interno, basato sull'integrato U3 (L78L05) che ricava 5 volt stabilizzati; per attivare questa opzione, il jumper JP5V va chiuso su 5VR. Chiudendolo su 5VA, lo shield riceve i 5 V direttamente dalla linea +5V di Arduino; ciò potete farlo solo se il numero di schede è limitato a due, perché oltre, l'assorbimento sul 5V di Arduino sarebbe eccessivo.

LIBRERIA ARDUINO

I²C I/O EXPANDER SHIELD

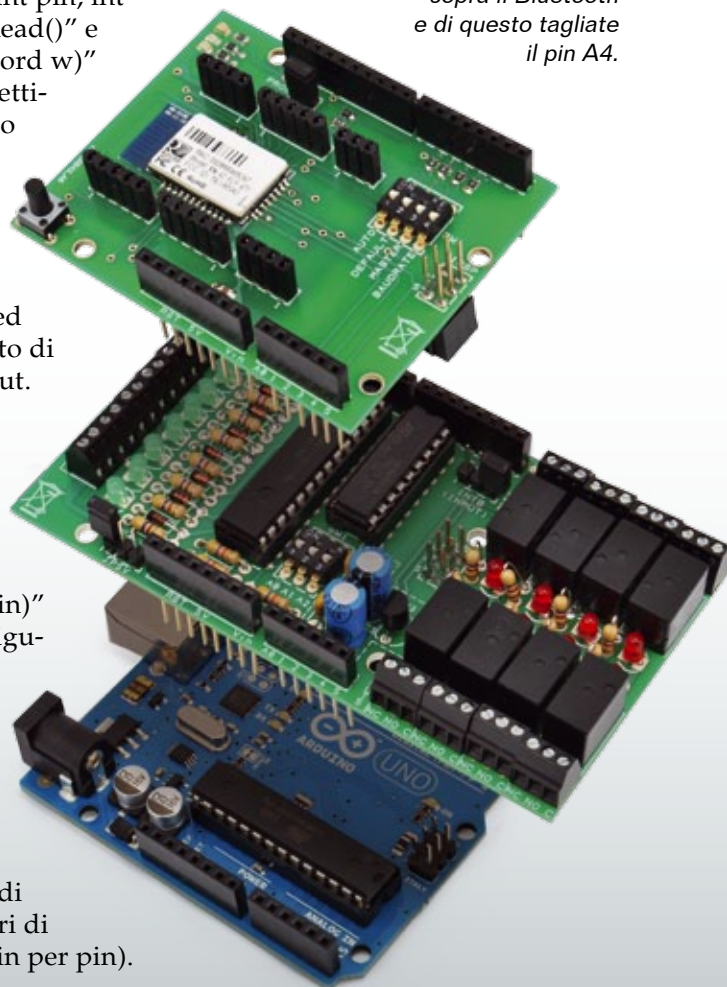
Per il progetto abbiamo sviluppato una libreria software

Arduino che fornisce tutte le routine necessarie per la rilevazione degli shield collegati e la relativa gestione degli I/O. La comunicazione con l'MCP23017 è basata sull'I²C Bus, pertanto la libreria utilizza "Wire.h" di Arduino; in essa sono presenti una funzione "begin(int i2cAddress)" per inizializzare il singolo shield identificato tramite indirizzo I²C, una "init()" per programmare correttamente i registri interni del chip secondo le nostre necessità, ed infine una "pinMode(int pin, int mode)" per indicare se i singoli pin di I/O sono input o output. Oltre alle funzioni di inizializzazione, le istruzioni che ci interessano sono le "digitalRead(int pin)", "digitalWrite(int pin, int val)", "digitalWordRead()" e "digitalWordWrite(word w)" che permettono, rispettivamente, di leggere lo stato di un singolo pin di input, scrivere un singolo pin di output, leggere lo stato di tutti gli otto pin di input ed infine, scrivere lo stato di tutti e 8 i pin di output. Siccome la gestione degli input avviene tramite interrupt, abbiamo previsto un'apposita funzione "pinDisableINT(int pin)" che permette di configurare il singolo pin come generatore di interrupt; in particolari applicazioni, potrebbe essere utile avere la possibilità di non impostare tutti i pin di input come generatori di interrupt, ma farlo pin per pin).

SHIELD RN-42 E RELATIVA LIBRERIA ARDUINO

Lo shield che realizza la connessione Bluetooth è quello basato sul modulo RN-42 della Roving Network, la cui descrizione è già stata fatta in articoli passati, quale ad esempio il MusicXmas del fascicolo di novembre 2013. Per questo, rimandiamo chi volesse conoscerlo nei dettagli ai precedenti articoli. Lo shield viene gestito da Arduino tramite un'apposita libreria. La comunicazione con il modulo RN-42 avviene tramite

Disposizione degli shield su Arduino; se usate Arduino Uno prima versione, mettete sopra il Bluetooth e di questo tagliate il pin A4.



Master e slave Bluetooth

Lo standard Bluetooth prevede due classi di dispositivi: master e slave. Della prima fanno parte tutti quei dispositivi che, di propria iniziativa, possono avviare una nuova connessione, mentre alla seconda appartengono quelli che attendono la richiesta di connessione. Inoltre, il Bluetooth è basato su request/reply, ossia ad ogni richiesta di comandi proveniente dal master, lo slave deve rispondere con pacchetti di conferma o risposta. Nell'applicazione proposta, lo smartphone è il master e lo shield Bluetooth lo slave. Il protocollo utilizzato prevede stringhe di quattro byte strutturate come di seguito.

- 1° byte: tipo di operazione (lettura o impostazione stato I/O scheda); GETINFO_CMD (byte 0x41) per lettura; SETINFO_CMD (byte 0x42) per impostazione.
- 2° byte: identificativo della risorsa. Sono previste 5 possibilità: stato completo scheda (STATUS_CMD; byte 0x40); uscite a relé singolo shield (OUTRLE_CMD; byte 0x41); ingressi digitali singolo shield (INPDIG_CMD; byte 0x42); nome Bluetooth del sistema (NMEBTH_CMD; byte 0x43) ed infine PIC code bluetooth (PNCBTH_CMD; byte 0x44).
- 3° byte: identificativo risorsa (numero dello shield per i comandi OUTRLE_CMD e INPDIG_CMD) oppure indice carattere nome o PIN (per i comandi NMEBTH_CMD e PNCBTH_CMD).
- 4° byte: stato di impostazione risorsa (solo per comando uscite) oppure singolo carattere nome o pin code.

Per i comandi OUTRLE_CMD e INPDIG_CMD, il terzo byte identifica il numero dello shield, mentre il quarto specifica lo stato di tutte le uscite o ingressi (ogni singolo bit del byte corrisponde alle uscite/ingressi da 1 a 8).

porta seriale hardware pertanto viene inclusa la "Serial.h" di Arduino.

La libreria implementa tutte le funzioni di basso livello necessarie per la comunicazione con il modulo RN-42; in essa sono



Fig. 1

presenti le funzioni:

- "begin()" che inizializza la porta seriale con baud-rate pari a 115.200 e attende che questa sia pronta per l'utilizzo;
- "Kill_Connection_RN_42()" per forzare la chiusura della comunicazione Bluetooth;
- "Reboot_RN_42()" per eseguire il boot software dell'RN-42;
- "Set_Factory_Default_RN_42()" per impostare la configurazione dell'RN-42 ai valori di fabbrica;



Fig. 6



Fig. 2

- "Set_Operation_Mode_Slave_RN_42()" e "Set_Operation_Mode_Master_RN_42()" che servono ad impostare le modalità Bluetooth, rispettivamente, *slave* o *master*;
- "Set_Extended_Status_String_RN_42()" per configurare la stringa "Extended Status String"; - "Set_Name_RN_42(String Name)" e "Set_PIN_Code_RN_42(String Pin_Code)" per programmare, rispettivamente, il nome e il pin code Bluetooth;
- "Enter_Command_Mode_



Fig. 7



Fig. 3



Fig. 4



Fig. 5

RN_42()) ed "Enter_Data_Mode_RN_42()" per entrare nelle modalità, rispettivamente, *comandi* e *dati*.

Oltre a queste routine di basso livello, quelle che più ci interessano per il software Arduino sono:

- "New_Connection_Request_RN_42()" e "Close_Connection_Request_RN_42()", le quali, rispettivamente, testano se dall'esterno è arrivata una richiesta di nuova o chiusura connessione;

- "Write_Buffer_Bluetooth_RN_42(char *Buffer_Bluetooth, int Buffer_Length)" e "Read_Buffer_Bluetooth_RN_42(char *Buffer_Bluetooth)" che ci servono per scrivere o leggere un buffer di dati tramite link Bluetooth;
- "Available_RN_42()", che ci permette di testare quanti byte di dati sono disponibili in ricezione.

FIRMWARE ARDUINO

Avendo definito in precedenza le librerie software per la ge-

stione degli shield I/O Expander e Bluetooth, la realizzazione del software Arduino risulta più semplice e veloce.

La CPU di Arduino si fa carico di gestire sia la comunicazione I²C con gli input/output (configurazione lettura/scrittura stati logici) che quella seriale con il modulo RN-42 (richieste di nuove connessioni o sconnessioni e protocollo di invio dati). Come risorse hardware, oltre al bus I²C e alla porta seriale richiesti dagli shield, vengono utilizzati il pin 13 per gestire



Fig. 8

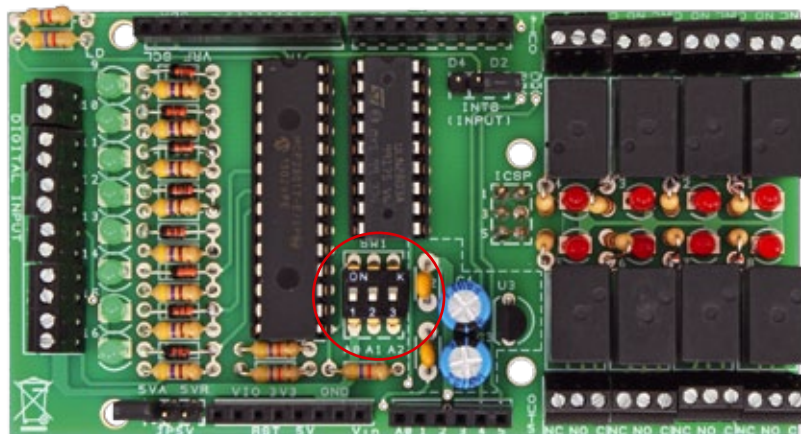


Fig. 9



Fig. 10

il LED presente sulla board Arduino e il pin 2 per leggere il segnale di interrupt degli shield I/O Expander. Sono utilizzati 36 byte della EEPROM per memorizzare il nome e pin code Bluetooth da associare, all'accensione, al link wireless. Nella funzione "setup()" vengono configurati i due pin digitali, viene inizializzata la EEPROM, si avvia il bus I2C e, infine, viene rilevato quanti e quali shield I/O Expander sono collegati. Allo scopo, si utilizza il bus I2C ed in particolare l'ACK che gli slave rispondono ai diversi comandi. Per ogni shield trovato, vengono impostati i versi di input e output dell'MCP23017. Successivamente, attraverso gli opportuni comandi seriali, viene configurato il modulo RN-42 dello shield bluetooth; in particolare, viene configurato come slave (comando "SM,0") e programmato il nome e pin code predefinito ("1234").



Con SW1 si imposta l'address I²C dello shield I/O Expander: notate la corrispondenza tra A0, A1, A2 e i singoli dip.

Caricare il firmware

Nel nostro progetto, la porta seriale di Arduino necessaria a caricare il firmware è la stessa usata per gestire lo shield Bluetooth. Per questa ragione, quando dovete programmare Arduino vi conviene rimuovere lo shield Bluetooth. Per non avere quest'onere, potete lasciare montato lo shield e usare il pulsante di reset presente su di esso: premendolo, il modulo RN-42 si porta nello stato di reset, quindi è disabilitato e risulta sconnesso dalla porta UART. Fintanto che il pulsante risulta premuto, Arduino ha la porta seriale disponibile, che pertanto può essere utilizzata per il download del firmware. A programmazione finita basta rilasciare il pulsante.

In seguito si entra nel ciclo principale di esecuzione: come prima operazione viene letto lo stato digitale del pin 2 (interrupt I/O Expander) e, se trovato a livello basso, si procede ad aggiornare lo stato degli input di tutti gli shield rilevati. Successivamente si passa a gestire la comunicazione Bluetooth: si testa la presenza di nuovi byte nel buffer di ricezione seriale ed in caso affermativo, i byte stessi vengono analizzati. Viene quindi verificata la condizione di richiesta di nuova connessione o disconnessione, altrimenti i byte vengono processati secondo le regole del protocollo di comunicazione. In particolare, i comandi sono suddivisi in operazioni di lettura e scrittura: le prime riguardano la lettura dello stato generale, nonché delle uscite e degli ingressi del singolo shield, mentre quelle di scrittura riguardano lo stato delle uscite e la configurazione Bluetooth.

SOFTWARE ANDROID

Il software sviluppato per smartphone Android ("ShieldIOControl") permette di verificare lo stato di tutti gli ingressi digitali e di leggere o modificare lo stato delle uscite.

La visualizzazione degli shield rilevati è gestita a pagine ed è possibile "sfogliare" le diverse pagine attraverso due appositi pulsanti.

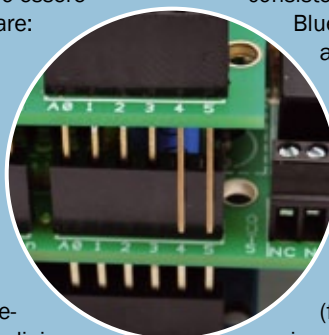
Le etichette grafiche che identificano i diversi input/output e relative descrizioni dello stato possono essere configurate da software e salvate in modo che vengano ripristinate alle successive esecuzioni. Segnaliamo, però, che tali impostazioni vengono memorizzate nello smartphone e non nella scheda elettronica (sarebbero necessari troppi byte di EEPROM); pertanto se si utilizzano 2 o più smartphone per la gestione dell'elettronica, è necessario eseguire la configurazione su entrambi i terminali.

All'avvio, dopo una schermata iniziale di presentazione del software (Fig. 1), viene avviata la scansione per il rilevamento di tutti i dispositivi Bluetooth presenti all'interno della zona di copertura dello smartphone. Nella lista visualizzata (Fig. 2) vengono mostrati sia tutti i sistemi già accoppiati, sia i nuovi rilevati; cliccando la voce della scheda ("BTIOControl" nel nostro esempio) la si seleziona e si richiede l'avvio del collegamento. È inoltre presente un pul-

sante per richiedere una nuova scansione ed il conseguente aggiornamento della lista. Quando viene rilevato un dispositivo, se è la prima volta lo smartphone lo trova (**Fig. 3**) viene richiesta la digitazione del PIN code ("1234" è il predefinito) e in seguito si passa nella schermata principale del programma (**Fig. 4**), nella quale, partendo dall'alto troviamo il numero di shield, lo stato delle uscite a relé e degli otto ingressi digitali corrispondenti. Di fianco all'etichetta che identifica lo shield, sono presenti due pulsanti per passare alla visualizzazione dello shield precedente e successivo.

Per ogni singola uscita viene utilizzata la seguente convenzione: sull'estrema sinistra è presente l'etichetta di testo che identifica la risorsa; un'altra etichetta di testo ne indica lo stato corrente. Inoltre, un pulsante più a destra permette di modificare lo stato corrente. Che l'uscita sia attiva oppure disattiva, l'etichetta di testo viene aggiornata di conseguenza, con ON (attiva) oppure OFF (disattiva) ed il pulsante è abilitato. Per ogni ingresso digitale è invece presente una sola etichetta di testo riportante lo stato corrente: ON nel caso in cui l'ingresso digitale sia a livello alto; OFF in caso contrario. Prendendo ancora come riferimento la **Fig. 4**, si vede che i primi quattro ingressi sono ON. Tornando ancora alle uscite, per richiedere la modifica dello stato corrente è sufficiente fare clic sul relativo pulsante; a questo punto il comando selezionato viene inviato alla scheda e viene mostrata una finestra indicante l'invio del comando (**Fig. 5**). Terminato l'invio, l'interfaccia grafica è automaticamente

Nel caso in cui decidiate di realizzare il progetto presentato in queste pagine utilizzando una scheda Arduino Uno (la versione di Arduino più semplice, ma anche più diffusa) vi può essere utile notare un particolare: le prime versioni della Uno utilizzavano i pin A4 e A5 per il bus I²C (segnale SDA e SCL); dalla Rev. 3 sono stati aggiunti due pin per separare questo canale dati, ma è stata fatta la scelta di lasciare comunque collegato il bus anche sui piedini A4 ed A5. Nel nostro sistema, la CPU di Arduino comunica con lo shield I/O Expander mediante il bus I²C, mentre il modulo RN-42 dello shield Bluetooth dispone di alcuni GPIO; tre di questi vengono portati verso Arduino (in particolare il GPIO9 fa riferimento al pin A4). Quindi la linea A4 risulterebbe collegata sia al bus I²C dell'MCP23017 che al GPIO9 dell'RN-42; questo problema pregiudica il corretto funzionamento della scheda, tanto che all'avvio



il bus I²C risulta bloccato e pertanto il software non viene eseguito correttamente.

La soluzione a questo inconveniente consiste nell'inserire lo shield Bluetooth sopra a tutti gli altri (la cosa è utile anche per evitare che il modulo RN-42 sia chiuso sopra e sotto e quindi la comunicazione RF venga ostacolata); inoltre, bisogna lasciare flottante il pin che corrisponde ad A4 di Arduino (flottante significa non inserito nel corrispondente strip femmina, ovvero piegato all'esterno per fare in modo che non entri in esso). In questo modo la Arduino Uno è in grado di comunicare con gli integrati MCP23017 tramite bus I²C e con il componente Bluetooth RN-42 tramite porta seriale (chiaramente il GPIO9 non potrà essere utilizzato o per lo meno collegato con Arduino). L'accorgimento qui descritto non serve, invece, se usate Arduino Uno Rev. 3.

aggiornata. Durante il normale funzionamento è presente un timer di aggiornamento sia dello stato corrente che della grafica. Ciclicamente il software si collega alla scheda, ne legge lo stato corrente ed aggiorna l'interfaccia grafica di tutti gli shield; se lo stato degli ingressi è variato, anche la relativa sezione grafica viene modificata. Agendo sul tasto "menu" di Android, in basso viene visualizzato il menu di scelta configurazione (**Fig. 6**) composto da due pulsanti: "BT Config" per selezionare la configurazione Bluetooth della scheda e "Text Config" per la configurazione delle etichette di testo. Nella prima pagina di configurazione (**Fig. 7**) sono presenti due sezioni di configurazione: "Nome sistema" e "Codice PIN"; abbinato ad ogni sezione

è presente un campo di testo per impostare la configurazione ed un pulsante per richiedere l'invio e modifica della singola configurazione. Nella seconda pagina di configurazione ("Text Config", **Fig. 8**) è possibile specificare otto etichette di testo descrittive di ciascuna uscita, oltre a due etichette che identificano lo stato delle uscite ed ulteriori due etichette abbinate ai due stati che gli input possono assumere. Infine, sulla destra è presente un pulsante che riporta la configurazione ai valori predefiniti (nell'immagine di **Fig. 8** l'esempio è relativo ai valori preconfigurati) mentre sulla sinistra c'è un pulsante "OK" che conferma i valori inseriti. Nell'immagine successiva (**Fig. 9**) vedete una configurazione di esempio dove abbiamo specifi-

cato che le ultime tre uscite gestiscono un allarme, le luci del salotto e quelle esterne e che le uscite attive vogliamo siano indicate con "alto" e quelle disattive con "basso". Premendo "OK", la configurazione viene aggiornata e salvata e in seguito anche la grafica della finestra principale viene aggiornata (Fig. 10).

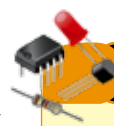
REALIZZAZIONE PRATICA

L'hardware del sistema è costituito da una scheda Arduino (supponiamo la Uno Rev. 3) da uno o più shield I/O Expander e da uno shield Bluetooth RN-42.

Dato che la Arduino si compra già pronta ed anche lo shield Bluetooth viene venduto da Futura Elettronica (www.futurashop.it) praticamente già montato (rimangono da montare solo il pulsante di reset, gli strip maschio/femmina di montaggio su Arduino e il dip-switch a 4 poli), l'unico circuito da realizzare è quello dello shield I/O Expander; quest'ultimo viene comunque venduto in kit di montaggio (sempre da Futura Elettronica) per facilitarvi il compito. Lo shield fa uso di componenti

tradizionali, quindi nessun problema: procedete in ordine di altezza (prima resistenze, diodi al silicio, LED, condensatori ceramici ed elettrolitici, dip-switch, ponticelli, il regolatore integrato L78L05, gli integrati DIP, i relé e le morsettiere) e seguite, per l'orientamento degli elementi polarizzati, il verso indicato nel disegno di montaggio che trovate in queste pagine. Completate il montaggio con gli strip maschio/femmina di collegamento degli shield. Nel caso in cui utilizzate un solo shield, la posizione dei pin del dip-switch a 3 poli non è significativa in quanto qualunque sia l'indirizzo PC corrispondente, non si pongono problemi di conflitto. Viceversa, se usate più shield di espansione, fate attenzione a configurare ciascuno con una diversa combinazione del dip-switch, altrimenti si verificheranno collisioni nella comunicazione PC. A riguardo, rammentiamo che dip a ON significa che il relativo pin dell'MCP23017 è collegato a massa e si trova a 0 logico.

Per la corretta impostazione ricordate che A0, A1, A2 corrispondono agli switch rispet-



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità; il master del circuito stampato dello shield I/O Expander per Arduino può essere scaricato dal sito della rivista. Questo shield (cod. FT1079K) è anche disponibile in kit al prezzo di 34,00 Euro; la scatola di montaggio comprende tutti i componenti, la bassetta forata e serigrafata e tutte le minuterie. Lo shield Bluetooth con RN-42 è disponibile già montato e collaudato al prezzo di 34,00 Euro (cod. FT1032M) mentre la board Arduino UNO Rev.3 costa 24,50 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>

tivamente verso l'MCP23017, centrale, verso C1/C2. Prima di mettere in funzione il sistema, ricordate di porre tutti i dip del dip-switch dello shield Bluetooth su OFF.

Per l'alimentazione dell'insieme, serve un alimentatore in grado di fornire 9 volt e 200 mA, più 180 mA per ciascuno shield I/O Expander montato. ■

RM ELETTRONICA

Forniture per hobbisti, scuole e industria

Distributore autorizzato:

**FUTURA
ELETTRONICA**

**ELSE
Kit**

**NUOVA
ELETTRONICA**

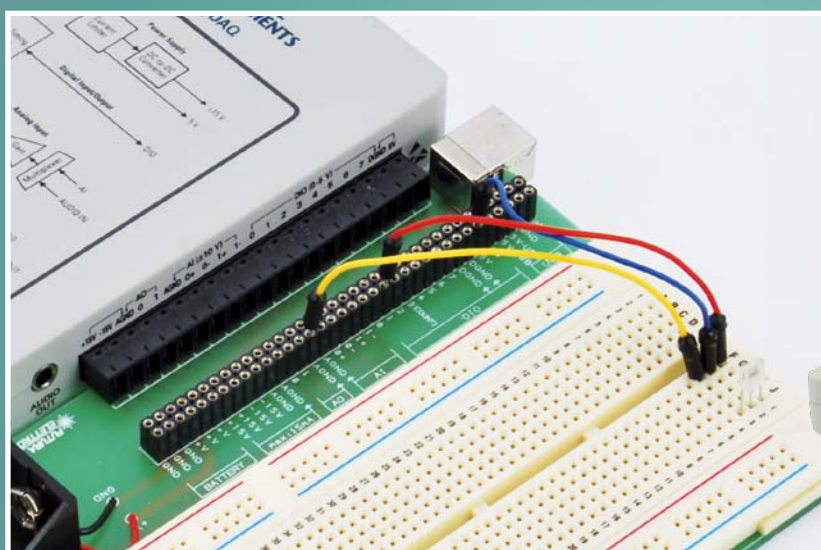
Il tuo punto
di riferimento
per l'elettronica
a ROMA

Via Val Sillaro 38 • 00141 Roma • Tel: 06-8104753

myKIT

MISURA DI IRRADIANZA CON CONVERTITORE LUCE-FREQUENZA

di MARCO MAGAGNIN



L'obiettivo di questa esercitazione è di realizzare un convertitore luce-frequenza TSL235R per effettuare una misurazione di irradianza utilizzando il contatore della scheda myDAQ e il software LabVIEW. L'irradianza è definita come la potenza radiante incidente su una superficie di area

unitaria; per potenza radiante s'intende la potenza emessa, trasmessa o ricevuta sotto forma di radiazione, espressa in watt (W). L'irradianza viene indicata con il simbolo E_e e la sua unità di misura nel Sistema Internazionale è il W/m^2 . Il convertitore luce-frequenza è composto

Questa esercitazione illustra come eseguire una misura di irradianza utilizzando un convertitore luce-frequenza collegato alla scheda myDAQ. La frequenza del convertitore sarà acquisita tramite DAQ Assistant, che viene installato in LabVIEW con il driver NI DAQmx e sarà convertita nel valore di irradianza corrispondente utilizzando la programmazione in LabVIEW.

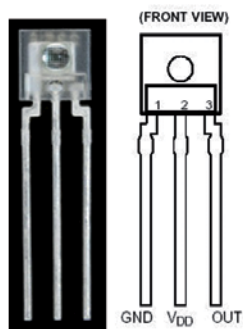


Fig. 1
Convertitore
luce-frequenza.

da un fotodiodo al silicio e da un convertitore corrente-frequenza contenuti in un singolo circuito integrato CMOS.

Il segnale in uscita dal sensore è un'onda quadra (duty cycle del 50%) avente una frequenza direttamente proporzionale all'irradianza. L'uscita digitale consente di interfacciarsi direttamente con una scheda di acquisizione dati: nel nostro caso, il dispositivo myDAQ. Il sensore è stato progettato per rilevare radiazioni elettromagnetiche con lunghezza d'onda che vanno da 320 a 1.050 nm, di conseguenza rileva anche la radiazione ultravioletta/visibile (UV/visibile). Questo convertitore luce-frequenza, inoltre, è caratterizzato per operare in un range di temperatura che va da -25 a 70°C.

Per comprendere meglio il funzionamento del convertitore luce-frequenza può essere utile osservare il segnale di uscita (onda quadra) su un oscilloscopio. Potete utilizzare quello incluso nel NI ELVISmx (Fig. 2) e collegare il sensore al dispositivo myDAQ. Il convertitore luce-frequenza utilizza tre connettori (Fig. 1). Il connettore VDD va collegato all'alimentazione +5V del dispositivo myDAQ, il connettore GND va collegato all'ingresso AGND e l'uscita OUT va collegata all'ingresso AI 0+ del dispositivo myDAQ. In-

fine collegate l'ingresso AI 0- al ground AGND. Una volta aperto l'Oscilloscope, selezionate AI0 alla voce Source su Channel 0 Settings e fate clic sul pulsante Run per iniziare a visualizzare il segnale.

Come potete osservare, sotto il grafico dell'oscilloscopio sono riportati alcuni valori tra cui la frequenza (Freq: 5,493 kHz). Se non doveste visualizzare bene il segnale a onda quadra, agite sulla manopola Time/Div per tarare l'asse orizzontale e sulla manopola Scale Volts/Div per tarare l'asse verticale.

Il costruttore del sensore fornisce il grafico frequenza-irradianza in scala logaritmica (Fig. 3) per convertire la frequenza del segnale di uscita dal sensore espressa in kHz in un valore di irradianza espresso in $\mu\text{W}/\text{cm}^2$. Prestate attenzione al fatto che il grafico è in scala logaritmica e quindi l'equazione che lega luminosità e resistenza è l'equazione di una "retta" in forma logaritmica. Come si vede dal grafico, la "retta" passa circa per l'origine e ha una pendenza unitaria. Si ricava che:

$$\log fO [\text{kHz}] = \log Ee [\mu\text{W}/\text{cm}^2] \rightarrow fO [\text{kHz}] = Ee [\mu\text{W}/\text{cm}^2]$$

questo mostra che, come detto prima, la frequenza è direttamente proporzionale all'irradianza.

REALIZZARE IL CIRCUITO DI MISURA

Il convertitore luce-frequenza utilizza tre connettori (Fig. 1). Il connettore VDD va collegato all'alimentazione +5V del dispositivo myDAQ, il connettore GND va collegato all'ingresso AGND e l'uscita OUT va collegata all'ingresso digita-

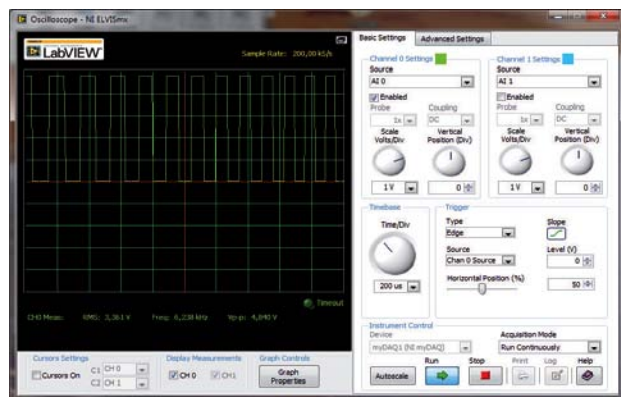


Fig. 2 - Utilizzo dell'Oscilloscope per la visualizzazione del segnale di output del sensore.

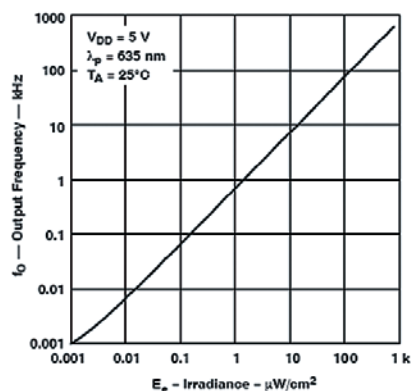
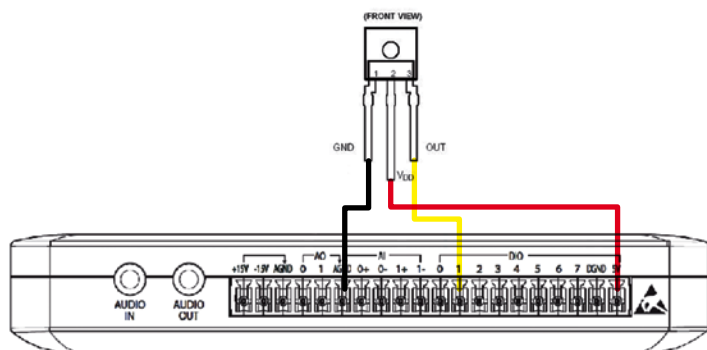


Fig. 3 - Grafico irradianza-frequenza fornito dal costruttore del sensore.



le DIO1 che è il counter GATE del dispositivo myDAQ.
Per ulteriori approfondimenti su contatori e temporizzatori del dispositivo myDAQ consultate il capitolo dedicato ad essi che trovate nella guida del dispositivo myDAQ.

PANNELLO FRONTALE LABVIEW

L'interfaccia utente che creerete mostra i valori di irradianza rilevata riportati su un pannello grafico e due indicatori.

CODIFICA DELLA STRATEGIA

In LabVIEW avete bisogno di acquisire la frequenza del segnale proveniente dal convertitore luce-frequenza. Questo valore è espresso in Hz e basterà convertirlo in kHz per ottenere direttamente il valore dell'irradianza rilevata del sensore.

$$fO [kHz] = Ee [\mu W/cm^2]$$

Infine, si fornirà in uscita il risultato a due indicatori grafici e a un grafico sul pannello frontale. Il diagramma a blocchi di LabVIEW è molto simile allo schema a blocchi di codifica.

COME FUNZIONA

All'interno del ciclo While, sulla sinistra è situato DAQ Assistant: esso è configurato per acquisire dal counter della scheda myDAQ un singolo valore di frequenza ogni volta che viene eseguito. Il dato acquisito viene poi diviso per mille in modo da ottenere la frequenza espressa in kHz, che sarà uguale al valore di irradianza rilevata dal sensore espressa in $[\mu W/cm^2]$. Successivamente questo valore viene inviato ai due indicatori grafici e al grafico sul Pannello Frontale per essere visualizzato. Tutto il codice presente nel ciclo While 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 100 ms, pertanto la frequenza di campionamento è di 10 campioni al secondo, ossia di 10 Hz.

COSTRUZIONE DEL PANNELLO FRONTALE

Ora implementerete il Pannello Frontale: aprite un nuovo file LabVIEW e, sul Pannello Frontale, cliccate con il pulsante destro del mouse fino a veder comparire la palette Controls, dove sono elencati tutti i tipi di controllo e di indicatore, ma anche decorazioni, tabelle, ecc. Per comodità, potete fermare la palette sullo schermo in modo da non dover rifare clic col pulsante destro del mouse ogni volta che ne avete bisogno; allo scopo cliccate sulla "puntina" in alto a sinistra della palette. Sulla palette Controls selezionate Modern (la prima voce), poi Numeric e inserite sul Pannello Frontale un indicatore Thermometer e un indicatore Meter. Con lo strumento Positioning ridimensionate questi due indicatori grafici (potete ridimensionare anche la parte colorata dell'indicatore Meter) e con lo strumento Labeling rinominateli entrambi assegnando il nome Irradianza $[\mu W/cm^2]$ (Fig. 4). Per far comparire un display con il corrispondente valore numerico associato a questi due indicatori grafici, cliccate con il tasto destro del mouse sopra tali indicatori e dal menu a tendina che comparirà selezionate Visible Items»Digital Display. Se preferite cambiare le impostazioni di scrittura (font, dimensione, colore, ecc.), nella barra del menu in alto cliccate su Text Settings (è il riquadro che riporta la dimensione e il font che state utilizzando: 15pt Application Font). Queste impostazioni possono essere variate anche per scrivere sul diagramma a blocchi. Adesso inserite anche un pulsante Stop che trovate nella palette Controls alla voce Modern»Boolean; inserite anche un Waveform Chart, situato alla voce Modern»Graph. Disponeteli e ingranditeli a

Tabella 1 - NI myDAQ assegnazioni dei segnali contatori\temporizzatori.

NI myDAQ Signal	Programmable Function Interface (PFI)	Counter/Timer Signal	Quadrature Encoder Signal
DIO 0	PFI 0	CTR 0 SOURCE	A
DIO 1	PFI 1	CTR 0 GATE	Z
DIO 2	PFI 2	CTR 0 AUX	B
DIO 3*	PFI 3	CTR 0 OUT	—
DIO 4	PFI 4	FREQ OUT	—

*Pulse-width modulation (PWM) pulse train measurements are generated through DIO 3

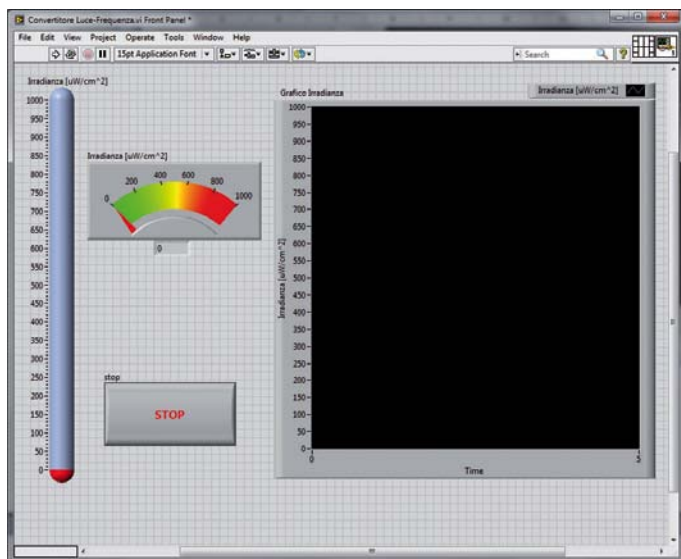


Fig. 4 - Pannello Frontale LabVIEW.



Fig. 5 - Schema a blocchi di codifica.

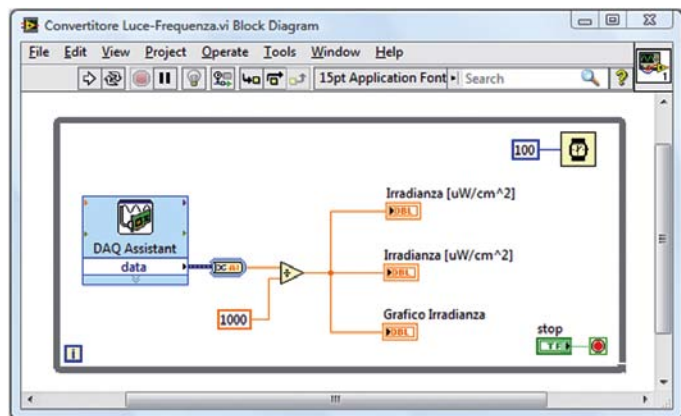


Fig. 6 - Diagramma a blocchi LabVIEW.

vostro piacimento con lo strumento Positioning e rinominateli con lo strumento Labeling, come vedete in Fig. 4.

Utilizzando il medesimo strumento rinominate anche l'etichetta dell'asse Y e la legenda in alto a destra del grafico con Irradianza [$\mu\text{W}/\text{cm}^2$], poi deselezionate la voce Autoscale Y dal menu a tendina che comparirà facendo clic col tasto destro del mouse sull'etichetta dell'asse Y. Ora, in corrispondenza dei valori estremi dell'asse y e degli altri due indicatori grafici inserite, usando lo strumento Labeling, i valori 0 e +1000 (se durante le misure che effettuate il va-

lore di irradianza supera i $1000 \mu\text{W}/\text{cm}^2$ inserite una soglia più alta così potrete visualizzarlo).

COSTRUZIONE DEL DIAGRAMMA A BLOCCHI

Ora vi occuperete della realizzazione del diagramma a blocchi. Se non lo visualizzate, potete farlo selezionando Window>Show Block Diagram o digitando <Ctrl+E>. Nel diagramma a blocchi, gli oggetti del Pannello Frontale appaiono 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 di stop con le stesse etichette che avevate messo sul pannello frontale. Inoltre dovreste visualizzare i terminali, ad esempio quello del pulsante di stop, in questo modo:

. Però è preferibile poterlo visualizzare con questo simbolo: . 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 selezionate 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) che come controlli (cioè producono dati). I controlli si distinguono dagli indicatori per due caratteristiche: la prima 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 gli indicatori hanno una freccia che mostra il dato entrare nel terminale); la seconda è il bordo intorno al terminale (i controlli hanno un bordo più spesso e gli indicatori più sottile).

Per trasformare un elemento da indicatore a controllo, dovete cliccare col pulsante destro del mouse sopra il rettangolino che lo rappresenta e impartire il comando Change to Control; questa operazione può essere effettuata anche dal Pannello Frontale. Notate che il pulsante Stop è già

Tabella 2

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

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 seguente 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 dovete inserire il blocco DAQ Assistant per acquisire la frequenza del segnale proveniente dal convertitore luce-frequenza. Le seguenti istruzioni guidano passo-passo nella configurazione del DAQ Assistant.

1. Assicuratevi che myDAQ sia collegato al PC;

2. Premete Ctrl-Spazio per aprire la finestra Quick Drop (al primo utilizzo ciò richiederà circa un minuto);
3. Cercate la voce DAQ Assistant e fate doppio clic su di essa 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. Counter Input;
 - c. Frequency;
 - d. Dev 1 (NI myDAQ) * Nota: Se si dispone di un altro hardware NI installato, myDAQ non sarà Dev1;
 - e. ctr0;
 - f. Finish;
6. Impostate il corretto range di frequenza di ingresso su Signal Input Range:
 - a. Max: 1MHz;
 - b. Min: 2Hz;
7. Lasciate le altre impostazioni come sono;
8. Fate clic su OK.

Una volta terminata la configurazione andate sulla palette Functions e inserite il blocco Convert from Dynamic Data situato alla voce Express»Signal Manipulation. Appena inserirete il blocco Convert from Dynamic Data si aprirà una finestra per la configurazione, alla voce Conversion selezionate Single Scalar e fate clic su OK. Ora alla voce Programming»Numeric selezionate un blocco Divide e inseritelo nel diagramma a blocchi. Per aggiungere il divisore 1000 a quest'ultimo blocco cliccate col tasto destro del mouse in corrispondenza del secondo ingresso e selezionate Create»Constant. Comparirà un riquadrino dove con lo strumento Labeling dovreste inserire il valore 1000. Adesso, con lo strumento Positioning sistemate i vari blocchi e gli indicatori come in Fig. 8 e utilizzando lo strumento Wiring collegateli aiutandovi con la figura. Per finire, inglobate tutto il codice grafico fatto finora dentro un While Loop che troverete sulla palette Functions alla voce Programming»Structures e collegate con lo strumento Wiring il terminale del pulsante Stop al conditional terminal nell'angolo in basso a destra del ciclo While. Come ultima cosa inserite il blocco Wait(ms) situato nella palette Functions alla voce Programming»Timing e collegatelo a una costante con valore 100 cliccando il tasto destro del mouse sul terminale d'ingresso del blocco Wait(ms), selezionando Create»Constant,

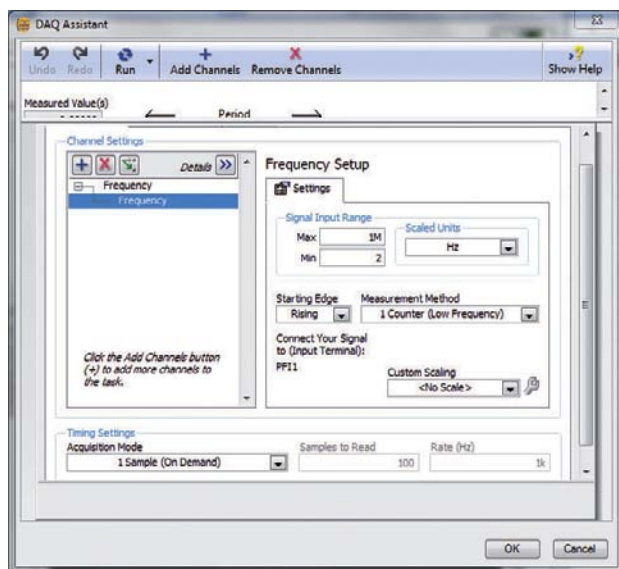


Fig. 7 - Configurazione DAQ Assistant.

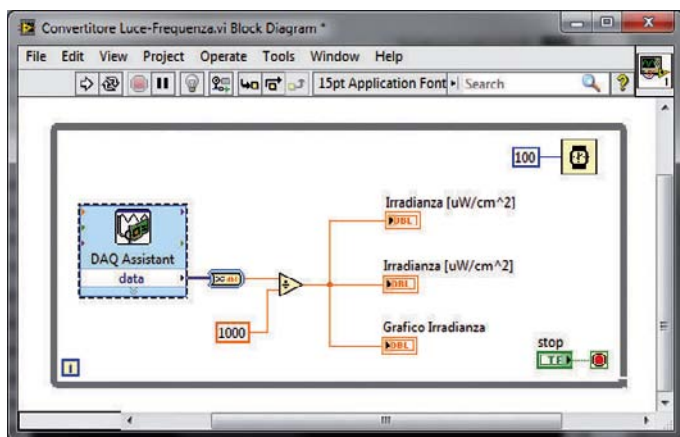


Fig. 8 - Diagramma a blocchi LabVIEW.

e inserendovi il valore 100. Questo per fare in modo che il ciclo vada in esecuzione ogni 100 ms e perciò il DAQ Assistant acquisisca 10 campioni al secondo.

Quando avrete finito di costruire il vostro VI, andate su File»Save as e scegliete un nome e una destinazione per salvarlo.

Ora cliccate sul pulsante Run  nella barra degli strumenti per mettere il VI in esecuzione (Fig. 9).

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

va usato soltanto se non riuscite ad arrestare il VI in altro modo. Notate che 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.

NOTA AGGIUNTIVA

Questa esperienza didattica può essere utilizzata per effettuare il test di una piccola cella fotovoltaica. La parte mancante può essere realizzata sfruttando gli ingressi analogici della scheda. Collegando la cella fotovoltaica agli ingressi AI0 e AI1 si può effettuare la misura della tensione e della corrente e ricavare così la caratteristica V-I della cella. Salvando i dati acquisiti nell'arco di una giornata o in intervalli di tempo prestabiliti su un foglio di calcolo, si potranno poi confrontare con i valori di irradianza rilevati nello stesso arco di tempo con il convertitore luce-frequenza. Questo permetterà di avere una stima dell'efficienza energetica della cella solare. ■

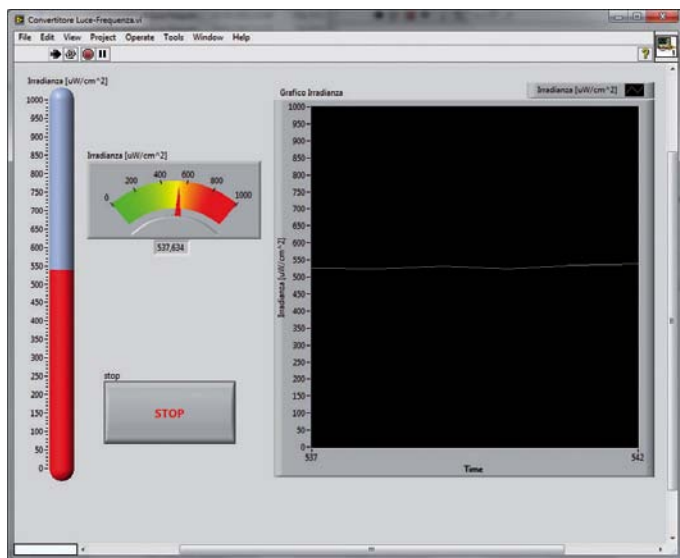
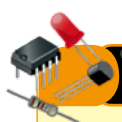


Fig. 9 - VI in esecuzione.



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

Il nuovo standard di rete PAN sviluppato da Aurel si basa su moduli che possiamo anche gestire con Arduino: ecco come.



USIAMO RFTide CON ARDUINO

dell'Ing. MIRCO SEGATELLO

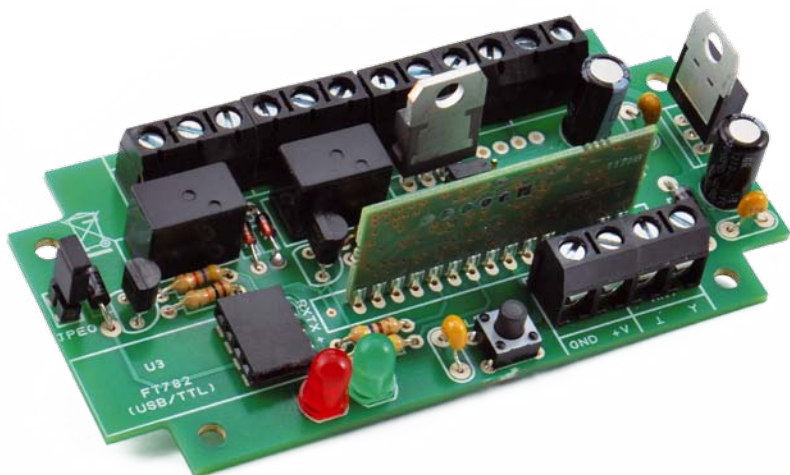
Le comodità, si sa, fanno piacere a tutti è per questo che non appena possibile ci attrezziamo per automatizzare alcune funzioni che altrimenti dovremmo svolgere manualmente, con inevitabile impegno di tempo ed energie: dall'impianto di irrigazione, alla tenda

che rientra automaticamente se si alza troppo vento, al controllo del riscaldamento in remoto della casa di villeggiatura, alle luci del vialetto che si accendono da sole appena arrivati. Per tutto questo possiamo utilizzare il termine domotica, che indica, appunto, tutti gli auto-

matismi presenti in un'abitazione o, più in generale, in uno stabile. In commercio sono disponibili diversi sistemi per realizzare una casa domotica e non è difficile trovare anche degli

impianti utilizzando la tecnologia a bus, la quale permette di connettere e di controllare ogni singolo elemento del sistema domotico.

La direzione che la domotica sta



prendendo negli ultimi tempi è quella che porta alla sostituzione del bus con una rete wireless, in cui ogni elemento intelligente scambia informazioni via radio. Il principale beneficio è la riduzione del cablaggio e conseguentemente dei tempi di installazione, cui si aggiunge la possibilità di realizzare punti mobili da riposizionare a piacimento all'interno dell'abitazione.

Uno dei prodotti che si stanno facendo strada nel settore della domotica wireless è l'RFTide dell'italiana Aurel.

LA RETE RFTide

I prodotti RFTide sono nati per lo sviluppo di piccole reti mesh facili da installare e da mantenere, destinate principalmente alla domotica all'interno di abitazioni, ma anche alla telemetria e alla smart-energy. L'impiego principale risiede nel gestire aperture e chiusure di tende, controllo di impianti di illuminazione, lettura di sensori, rilevamento dello stato di attuatori. Sebbene il sistema sia nato essenzialmente per essere integrato nella domotica, si può applicare alla sicurezza, giacché consente di realizzare sistemi di monitoraggio dei locali, integrandosi perfettamente con gli appositi sensori Aurel, permettendo



di realizzare sofisticati impianti d'allarme wireless, eventualmente affacciati sul web per effettuare il monitoraggio a distanza e per gestire da remoto le varie funzionalità. Mediante l'interconnessione ad una rete Ethernet e al web, il Gateway RFTide permette di rea-

lizzare una rete wireless PAN, che monitora l'attività ed il consumo dei vari utilizzatori elettrici e può interagire con essi, attivandoli o disattivandoli in modo intelligente, tenendo conto dei consumi e delle fasce orarie che garantiscono il maggior risparmio.

La rete RFTide è costituita da due classi di dispositivi: una o più unità Master (che possono essere telecomandi o Gateway) e una serie di Slave interfacciati con attuatori o sensori in generale. I dispositivi Master inviano i comandi (i quali possono essere comandi di acquisizione o per l'esecuzione di operazioni) mentre gli Slave sono quelli che formano i nodi della rete. In realtà i comandi possono essere inviati, oltre che dall'apposito telecomando Aurel, da altri sistemi interfacciati alla rete, come smartphone

RFTide è ideale per il controllo domotico.



868 MHz Mesh Network



Il sistema Aurel consente la creazione di reti mesh.

o PC, mediante il Gateway, che rappresenta un punto di accesso alla rete. La RFTide prevede due tipi di trasmissione: quella dove un Master (tipicamente un telecomando) invia nella rete comandi che devono essere eseguiti dagli Slave; quella originata da uno Slave e diretta al Master, che contiene dati diretti a quest'ultimo e relativi, ad esempio, all'acquisizione di una lettura. Se il Master è collegato ad un computer, il comando può contenere un'interrogazione dello stato dello Slave o di un sensore ad esso collegato, oppure il valore di una grandezza misurata come ad esempio una temperatura; in quest'ultimo caso il Master non trasmette comandi di attivazione agli utilizzatori, ma diviene veicolo di trasporto di dati da e verso l'eventuale computer o altro

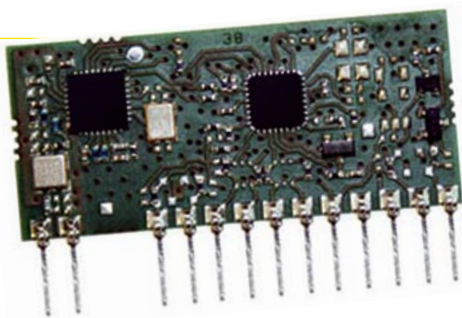
sistema collegato ad esso. Gli Slave sono unità di vario genere che possono attivare utilizzatori elettrici mediante relé, oppure acquisire dati riguardanti la misura di temperatura o altre informazioni da sensori. La rete RFTide opera ad 868 MHz, banda di frequenze in cui Aurel ha una consolidata esperienza e nella quale al momento operano meno sistemi che nella classica ISM a 2,4 GHz; ciò permette una comunicazione meno disturbata. Inoltre, operando ad una frequenza più bassa, le perdite sono minori che a 2,4 GHz, quindi il consumo di elettricità da parte dei dispositivi è ridotto rispetto, ad esempio, a quelli ZigBee. Come nello ZigBee, nell'RFTide si può realizzare la comunicazione tra nodi che non si "vedono" direttamente, sfruttando il mesh

networking: in pratica i dati trasmessi dai Master verso le unità Slave possono essere ripetuti da altri Slave o da appositi dispositivi chiamati Repeater. Pertanto, quando uno Slave intercetta un segnale radio contenente dati secondo il protocollo RFTide, se questi dati contengono un comando diretto ad esso lo Slave lo esegue, altrimenti lo ritrasmette. Analogamente avviene se uno Slave intercetta dati trasmessi da un'unità analoga e diretti ad un Master.

Proprio per questa modalità di funzionamento, la rete RFTide ammette l'esistenza di nodi mobili, essenziale se si vuole utilizzare un telecomando di controllo da qualsiasi punto della casa. L'elemento principale di una rete RFTide è un dispositivo Master, cui possono essere associati tanti nodi (Slave); la procedura di abbinamento è semplicissima e simile a quella di apprendimento dei dispositivi basati sull'encoder rolling-code HCS creato da Microchip per l'impiego nel comando a distanza via radio. Ripetendo questa procedura per tutti i nodi, si crea la rete; questo modo di funzionamento semplifica il ruolo dell'installatore e risolve il problema dell'installazione iniziale. Inoltre, per modificare la rete o aggiungere un nodo basta semplicemente ripetere il processo di abbinamento.

IL MODULO BASE RFTide

Le unità Master e Slave sono basate su uno stesso modulo base RFTide; la distinzione tra Master e Slave si fa per come esso viene impiegato. Il modulo base RFTide è formato da un ricetrasmittitore RF ad 868 MHz interfacciato con un microcontrollore, il quale rende disponibili all'esterno due linee di uscita GPOUT (General Purpose Out) utilizzabili per



Il modulo base RFTide.

comandare carichi mediante transistor o line-driver, e due linee digitali GPIN (General Purpose In) per acquisire livelli logici. Lo stadio RF è un ricetrasmittitore ad 868 Mhz, capace di una sensibilità in ricezione di ben -100 dB e di una potenza in trasmissione di +7 dB.

Aurel produce i moduli RFTide in due versioni, distinte per tensione di alimentazione: quelli a 5V sono destinati ad applicazioni alimentate dalla rete a 220 Vca, mentre quelli a 3V sono progettati per l'alimentazione a batteria. Gli assorbimenti sono molto contenuti: appena 33 mA in trasmissione e 10 mA in ricezione; è prevista una modalità di Standby, nella quale l'assorbimento scende a 4 µA. Nella versione a 3V, le due linee di ingresso possono essere configurate per leggere tensioni analogiche.

Per realizzare un Master è necessario utilizzare un modulo base RFTide provvisto di un numero seriale identificatore univoco (NetworkID di 4byte) che viene assegnato durante la fabbricazione e non può essere modificato; in questo caso si parla di "moduli serializzati". Ciascuno Slave ha invece come indirizzo predefinito 0x0000 e, durante la fase di associazione, questo assume come NetworkID il valore del Master, copiando il nuovo valore nella memoria non volatile interna. Solo moduli con lo stesso NetworkID possono comunicare tra loro, tuttavia è possibile copiare il numero seriale da un Master Remote Control ad un altro in modo che i due possano far parte della stessa

rete. Per completezza diciamo che qualsiasi modulo può essere configurato come Master, ma se non è stato serializzato in fabbrica esso avrà come Network-ID il valore 0x00 non modificabile e qualsiasi nodo associato ad esso avrà sempre Network-ID pari a 0x00. La rete, in questo caso, pur funzionando correttamente, non sarà univoca, perciò due reti così configurate e operanti nello stesso campo di copertura interferiranno tra loro.

L'assegnazione di un codice univoco a ciascun Master e quindi alla rete che gli è abbinata, permette di far operare più reti nello stesso raggio d'azione e perciò di evitare l'interferenza, ad esempio, tra appartamenti vicini.

Non è possibile clonare reti esistenti o comunque intromettersi in reti esistenti, se non copiando il numero seriale del Master, che ha configurato la rete, in un altro Master, operazione eseguibile solo disponendo fisicamente del Master da clonare.

Una cosa interessante della rete RFTide è che ciascun nodo può comunicare direttamente con qualsiasi altro; infatti ciascun modulo della rete ha un indirizzo univoco assegnato durante la fase di associazione, un po' come avviene con l'indirizzo IP dei computer in una rete. Questo indirizzo (da non confondere con il Network-ID) consente di distinguere i vari moduli connessi a una rete e di dirigere le comunicazioni da modulo a modulo (Point-To-Point) secondo necessità; pertanto gli Slave potranno essere programmati per comunicare tra loro senza interessare il Master. È questo, ad esempio, il caso di un sensore di vento che, superata la soglia di allarme, invia il comando di chiusura al modulo che gestisce la tenda motorizzata. Un'altra interessante modalità

di comunicazione è chiamata Broadcast: in essa il Master (o un nodo) può comunicare con tutti gli altri contemporaneamente inviando loro un comando o un messaggio specifico.

I tempi di trasmissione e ricezione nelle reti RFTide sono molto contenuti e indipendenti dal numero di nodi presenti: il tempo di latenza è infatti di soli 200 ms tra l'invio di un comando e la ricezione di conferma, sia con un nodo che con 256 (che è il numero massimo di moduli ammessi). Per rendere la rete più affidabile, è previsto che ad ogni comando inviato dal Master, lo Slave destinatario risponda con un messaggio di avvenuta ricezione; inoltre lo stesso Slave può essere interrogato sullo stato di esecuzione del comando.

UNITÀ MASTER

L'unità Master è un modulo base RFTide programmato per funzionare come tale; la più semplice è il telecomando Aurel (Remote Control) RFTide, progettato per impartire comandi di attivazione agli utilizzatori gestiti ad unità Slave come ad esempio la gestione di tapparelle motorizzate, illuminazione, elettroserrature, ecc. L'unità Master Remote Control è un vero e proprio telecomando dotato di pulsanti per inviare comandi e per avviare la procedura di abbinamento delle unità Slave: sul pannello frontale riporta un tasto per selezionare l'attuatore e tre per impartire i comandi, oltre ad una fila di LED che segnalano lo stato degli attuatori; sul retro ha il pulsante con cui si avvia la procedura di abbinamento agli Slave. Quanto ai LED, la fila segnala qual è il canale selezionato cui si trasmette il comando, mentre il LED multicolore è usato per segnalare lo stato assunto dall'attuatore collegato allo Slave

cui è stato inviato un comando. In una rete RFTide ci possono essere più dispositivi Remote Control; ogni Remote Control può entrare in una rete RFTide semplicemente apprendendo il codice del telecomando esistente: basta avviare l'apposita procedura su entrambi i Remote Control.

Il Master Remote Control può trasmettere in modo Broadcast: ciò significa che il comando è diretto a tutti gli Slave contemporaneamente; tale funzione è utile, ad esempio, per chiudere insieme tutte le tapparelle motorizzate di una casa.

Quando un modulo serializzato dispone di un convertitore USB/seriale, diventa un USB-Dongle che una volta connesso ad un PC lo fa diventare un Master e quindi in grado di gestire la rete e comunicare con tutti i moduli. L'Aurel fornisce gratuitamente un software per Windows, denominato RFTide-v1.0, che permette la creazione e la gestione di un'intera rete via PC.

Se, ad esempio, connettiamo l'USB-Dongle ad una Raspberry Pi opportunamente programmata per funzionare da interfaccia di rete Ethernet, realizziamo un Master che funziona da Gateway con specifiche capacità di interfacciamento. Il Gateway Aurel RFTide può interfacciarsi con dispositivi esterni alla rete, trasferendo eventualmente i dati su reti Ethernet o WiFi, perciò consente

di interagire con la rete RFTide usando PC remoti connessi ad una rete Ethernet oppure, tablet e smartphone connessi in WiFi. Il Gateway trova applicazione in sistemi decisamente più articolati e complessi del semplice comando a distanza: ad esempio nella telemetria e nel monitoraggio di ambienti; nel controllo dei consumi energetici (in questo caso permette di acquisire i consumi e inserire o disinserire utilizzatori collegati alla rete elettrica); nella gestione del condizionamento e del riscaldamento (permette di acquisire i dati su temperatura e umidità nell'ambiente e governare l'attività di caldaie, pompe di calore, ecc.); nella sicurezza (consente di verificare l'attività di sensori PIR, radar, contatti ecc. e gestire gli attuatori di allarme); nella domotica (permette di governare illuminazione, tapparelle e tende da esterno motorizzate sulla base dei dati forniti da sensori crepuscolari e di pioggia). Insomma, il Gateway va usato quando si desidera interfacciare l'installazione RFTide con una rete tipo LAN e consentire il controllo da computer, smartphone o da remoto tramite il web.

UNITÀ SLAVE

Qualsiasi modulo RFTide non serializzato può svolgere la funzione di Slave ed il suo compito diventa quello di ricevere i comandi dal Master e di eseguirli; può altresì essere interrogato sullo stato dei suoi ingressi o venire usato per leggere la condizione di sensori ad esso collegati.

In questo caso lo Slave è costantemente nello stato di standby (TX spento) e si risveglia solo quando deve trasmettere dei dati verso la rete. Il risveglio può essere programmato ad intervalli

regolari, ma può anche avvenire alla commutazione dell'ingresso GPIN0. Queste caratteristiche consentono di alimentare a batteria gli Slave, garantendo lunghissima autonomia d'esercizio.

Gli Slave, oltre ad essere dei ricetrasmittitori, fanno da ponte e quindi permettono di realizzare reti ben più estese di ciò che permetterebbe la massima distanza coperta dal segnale radio trasmesso dai Master o dagli Slave stessi; gli Slave formano quindi una rete mesh con funzione di ripetitori.

Quando sulla rete viaggia un comando, solo il nodo Slave destinatario risponderà, eseguendo l'operazione richiesta, ed eventualmente trasmettendo i dati richiesti; ciascuno Slave, infatti, dispone di un indirizzo univoco assegnato dal Master durante l'associazione alla rete.

Per funzionare in una rete e ricevere i comandi in arrivo da un Gateway o da un Remote Control, ciascuno Slave va abbinato mediante una semplice procedura: basta premere l'apposito pulsante di apprendimento collegato al suo piedino 7 ed il tasto Program sul Remote Control (dal Gateway occorre impartire il comando di Learning).

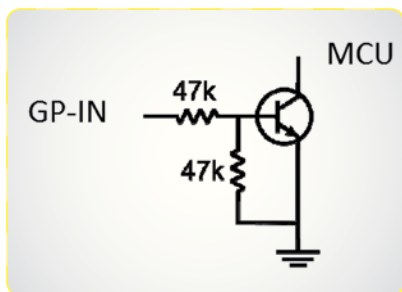
PROFILO DEI MODULI

Con il termine *profile* (profilo) l'Aurel identifica la specifica funzionalità del modulo nella rete; nello standard RFTide sono previsti sei profili che vanno impostati durante la fase di associazione.

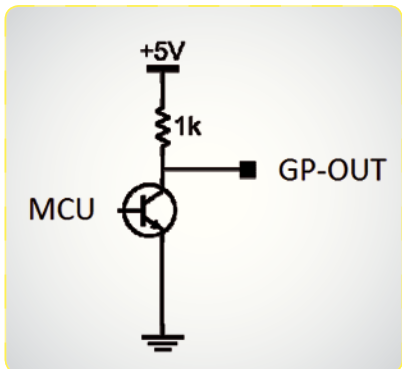
- **NONE profile**; nessuna funzione assegnata. Questo è il profilo del Master ma può essere usato anche per gli Slave che si vogliono gestire completamente da remoto (attivazione uscite, lettura stato ingressi ecc.); ogni messaggio diretto a un modulo in profilo NONE è disponibile sulla porta seriale e utilizzato

Il Master Remote Control per RFTide.





Stadio di ingresso (sopra) e di uscita (sotto) del modulo RFTide.



da un'eventuale logica programmabile per eseguire funzioni complesse o comunque non nativamente implementate dall'Aurel.

- **MOTOR** profile; interfacciando un motore bidirezionale con i relativi finecorsa, il modulo gestisce in automatico l'apertura e la chiusura di tende o tapparelle. In questa modalità il modulo riceve solamente i comandi di apertura, chiusura e stop del motore.
- **PWM** profile; in questa modalità l'uscita GPOUT0 genera un segnale PWM grazie al quale si può gestire (tramite adeguati driver di potenza) lampade o motori elettrici in continua. Il valore del PWM è controllato dai tasti CH+ e CH- del telecomando e possono assumere cinque differenti valori.
- **METERING** profile; in questa modalità, il nodo, periodicamente o alla commutazione del pin GPIN0, invia al Master lo stato degli ingressi ed il valore degli ADC. Il modulo è quindi

sempre in modalità di basso consumo, chiamata stand-by, attivando la trasmissione solo quando serve; anche la ricezione di un comando valido risveglia il modulo, il quale ritorna immediatamente in modalità di basso consumo. Il profilo è utile per la rilevazione di dati tramite sensori e negli impianti d'allarme.

- **REPEATER** profile; in questa modalità, il nodo non esegue alcuna funzione specifica e si limita a ritrasmettere il segnale ricevuto. Indispensabile se non ci sono abbastanza Slave per assicurare la copertura della rete.

RIPETITORI

Anche detti Wall-Repeater, sono dei nodi che ricevono il segnale radio e lo ritrasmettono; in altre parole, sono nodi non intelligenti che ritrasmettono i segnali che ricevono senza fare distinzioni. I Wall Repeater sono realizzati con un modulo configurato come ripetitore, uno stadio di alimentazione che gli permette di essere alimentati direttamente dalla tensione di rete a 230 Vca e una batteria di backup per assicurarne il funzionamento anche in caso di black-out.

Si tratta di ponti radio in miniatura che consentono ai Master di raggiungere gli Slave destinatari dei loro comandi, ma che sono troppo lontani per essere raggiunti direttamente dal segnale RF; viceversa, permettono ai Master di ricevere eventuali dati che hanno richiesto agli Slave troppo lontani. In una rete RFTide si possono installare quanti ripetitori servono, senza limite. Un malfunzionamento di uno o più ripetitori non pregiudica il funzionamento di tutta la rete, ma può renderne inutilizzabile la parte più lontana, la quale non può più ricevere il segnale dal Master.

Se nella rete sono posizionati sufficienti Slave, è possibile omettere questi ripetitori.

STADIO INGRESSO/USCITA

Nei moduli a 3V, i due ingressi possono essere configurati come analogici, così da poter leggere tensioni prese dall'esterno; tali ingressi giungono direttamente al microcontrollore tramite due resistenze in serie, del valore di 22 ohm. Applicando un eventuale pulsante, bisogna connettere all'ingresso interessato una resistenza di pull-up o pull-down esterna.

Nei moduli a 5V, i pin di uscita fanno uso di due transistor open collector dotati ciascuno di una resistenza di pull-up interna, del valore di 1 kohm. Quando l'uscita è attivata, il transistor è spento ed il livello di tensione in uscita è di +5V, mentre quando la stessa uscita è disattivata (in reset) il transistor è in saturazione e l'uscita assume il valore di circa zero volt. Lo stadio di ingresso è composto da un transistor alimentato in base; per attivare un ingresso è necessario che il contatto sia connesso tra un pin GP-IN e i +5V. Il pin di LEARN è già fornito internamente di una resistenza di pull-up, perciò per comandarlo basta collegare un semplice pulsante tra il pin LEARN e GND. Per l'antenna è sufficiente uno spezzone di filo di rame dello spessore di 1 mm, lungo 85 mm (1/4 d'onda) ma è possibile realizzare apparati con antenna integrata o esterna.

LA PRIMA APPLICAZIONE

Viste le caratteristiche e le potenzialità dei moduli RFTide, vediamo come realizzare una piccola rete che disponga delle caratteristiche di una rete avanzata di tipo mesh. Per realizzare una rete con i moduli RFTide è necessario

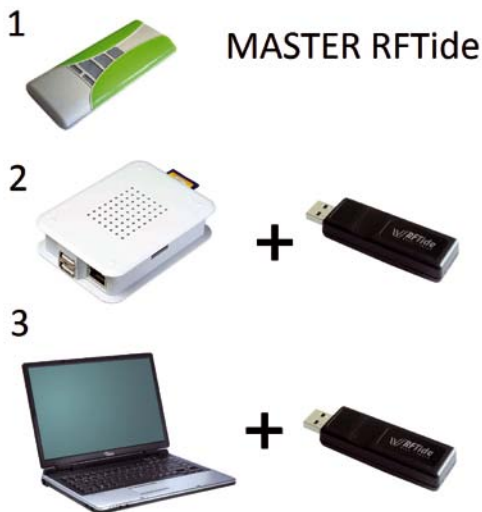
disporre di almeno un Master e, per il momento, prendiamo in considerazione solo componenti proposti dall'Aurel; nella prossima puntata pubblicheremo il progetto di un nostro Master interfacciabile con Arduino: sarà in pratica uno shield per Arduino contenente un modulo RFTide, per il quale forniremo il software in grado di realizzare un Master per RFTide.

Per il momento, diciamo che le soluzioni "commerciali" per realizzare un Master sono:

1. telecomando (Remote Control) per reti fino a 5 moduli;
2. USB-dongle da usarsi con PC e software dedicato (Aurel RFTide-v1.0);
3. Gateway basato su RaspBerryPI con USB-dongle.

In questo articolo prenderemo in considerazione le prime due soluzioni.

I moduli RFTide si presentano come un circuito stampato delle dimensioni di 39x17x2mm con un connettore strip single-in-line da 15 pin a passo standard di 2,54 mm. Per realizzare uno Slave è necessario interfacciare il modulo con dell'hardware; il minimo



Possibili Master in una rete RFTide.

Apple approda alla domotica

Sembra che il marchio della Mela non si accontenti più di fare smartphone e tablet, ma abbia deciso di allargarsi fino al settore della domotica; in un certo senso dispositivi come l'iPhone e in generale gli smartphone, già sono adatti al controllo domotico mediante App specifiche e hardware interfacciato via Bluetooth o Internet, ma ciò che Apple propone riguarda un dispositivo dedicato all'automazione domestica che consentirà di gestire tutta la casa mediante un iPhone, un iPad o un MacBook. Il sistema proposto è un hub che si interfaccia con qualsiasi apparecchio elettronico di casa (termostato per la climatizzazione, centralina di comando dell'illuminazione, elettrodomestici, TV ecc.) e, mediante il GPS, rileva la posizione degli utenti (localizzandoli tramite lo smartphone o un localizzatore in auto) e attivare, ad esempio, il riscaldamento quando stanno rincasando o accendere le luci di una stanza quando vi stanno entrando.

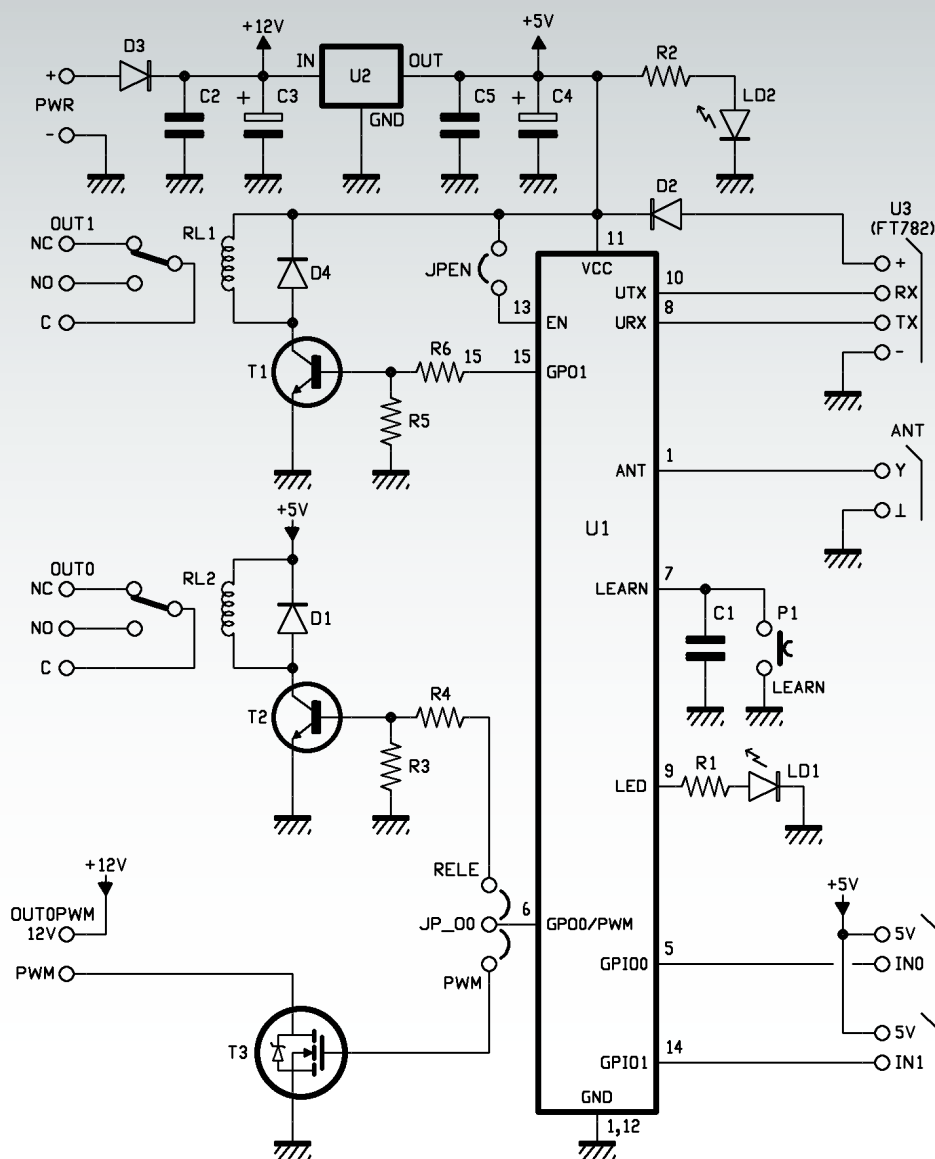


necessario per poterlo usare è di connettergli il pulsante di LEARN e, per testarne il funzionamento, due pulsanti per gli ingressi e due LED sulle uscite. Il tutto è facilmente assemblabile su di una breadboard seguendo lo schema riportato a pagina seguente. Per l'accensione dei LED, posti nelle uscite GPOUT, si sfrutta la resistenza di pull-up interna al modulo.

Per questo semplice Slave è possibile utilizzare un modulo RFTide (distribuito da Futura Elettronica con codice 8110-RFT-868-5V) e pertanto l'alimentazione deve essere tassativamente stabilizzata, del valore di 5 V; considerate che l'assorbimento non supera i 50 mA. Chi desidera una soluzione più pratica, può optare per la *breakout board* FT1083K che viene fornita in kit di montaggio da Futura Elettronica (www.futurashop.it). Si tratta di una scheda che permette di utilizzare rapidamente i moduli RFTide come unità Slave, ma può fungere anche da Master; essa contiene un modulo RFTide

a 5V, uno stadio di alimentazione, due pin di ingresso e uscita disponibili su una comoda morsettiera. Lo stampato viene fornito già pronto e serigrafato ed è sufficiente saldare i pochi componenti. Questo circuito è infatti pensato per essere cablatto direttamente ad interruttori e fincorsa e, tramite due relé, può comandare direttamente piccoli motori a bassa tensione; comunque, aggiungendo esternamente dei relé di potenza, è possibile gestire carichi di potenza, anche a 230 Vca. L'antenna può essere un semplice spezzone di filo di rame rigido della lunghezza di 85 mm oppure un'antenna specifica accordata sulla frequenza degli 868 MHz. Lo stampato dispone anche di un connettore per la comunicazione seriale, al quale può essere facilmente connesso un convertitore USB/TTL utile per interfacciare il modulo ad un PC; il connettore è compatibile con il convertitore USB/seriale FT782M. Per la sua alimentazione è sufficiente utilizzare un piccolo alimentatore non

[schema **ELETTRICO**]



stabilizzato da 9÷12 volt. La breakout board dispone di due ingressi digitali GPIN0 e GPIN1 in logica positiva, ovvero il contatto di chiusura è connesso tra l'ingresso e +5V. Sono disponibili due uscite digitali di tipo ON/OFF a relé ed il pin GPOUT0 può essere configurato come uscita PWM, secondo la necessità di utilizzo del modulo. Nella modalità ON/OFF, le due linee sono utilizzate per attivare i due relé, ma commutando il jumper su PWM, l'uscita GPOUT0 sarà disponibile sul morsetto di nome

PWM tramite un MOSFETT di potenza. Nel circuito è disponibile un jumper (JPEN) che gestisce l'abilitazione del modulo: sconnettendolo, il modulo va in standby. Il pulsante P1 può essere usato per la procedura di LEARN, ovvero per includere il modulo all'interno della rete, a completare il modulo un LED per segnalazione alimentazione ed un LED per la conferma di learning.

CREARE LA RETE RFTide

Supponiamo di disporre del telecomando originale Aurel a cinque

canali con funzione di Master e di voler configurare una semplicissima rete con un modulo slave stand-alone, la prima cosa da fare è associare lo slave con il Master e creare una prima rete con un nodo.

Procedura di learning (modo PWM o MOTOR)

La procedura di *learning* è necessaria per inserire il modulo nella rete ed assegnargli un suo indirizzo. Per prima cosa accendete il modulo Slave, poi premete il pulsante LEARN finché il LED verde LD1 (connesso al pin 9 del modulo) si accende.

Entro quindici secondi, sul telecomando, che funge da Master, selezionate il canale al quale volete associare il modulo (pulsanti CH+ e CH-) e quindi premete il pulsante PROG1 per l'associazione. Se durante questa fase il jumper MODE è inserito, verrà programmata la modalità PWM, altrimenti sarà configurata la modalità MOTOR. Se la procedura andrà a buon fine, il LED dello Slave lampeggerà per qualche secondo, mentre se il LED dovesse spegnersi senza lampeggiare significherà che la procedura è fallita.

Ricordiamo che in modalità PWM il duty-cycle può essere impostato da remoto su sei differenti valori tramite i pulsanti UP e DOWN: 0%, 0,4%, 1,6%, 6,2%, 25% e 100%. Per impostare valori differenti è necessario accedere alla rete con un Master più evoluto (ad esempio gestito da PC).

Procedura di learning (modo SENSOR)

È possibile configurare il modulo in modalità sensore, in modo che all'attivazione di un ingresso venga inviato un comando ad un altro modulo, invece che al Master: ciò può servire ad esempio per gestire l'attività di un avvisa-

tore acustico al rilevamento del segnale di un sensore di intrusione o di sconnettere, mediante un relé, un carico al superamento di un certo assorbimento di potenza. La configurazione si effettua premendo per circa due secondi il pulsante di LEARN del modulo, finché il LED si accende. Poi si preme il pulsante CH+ o CH- per attivare il telecomando che funge da Master e successivamente il pulsante PROG2, per inviare il comando di PROG_SENSOR_MESSAGE al modulo sensore. Se la procedura è andata a buon fine, il LED sul modulo lampeggerà alcuni secondi; se invece si spegnerà, la procedura sarà fallita.

Dunque, si deve nuovamente premere il pulsante di LEARN dello Slave per impostare la funzione, che verrà avviata qualora sarà attivato l'ingresso GPIN0 che funge da trigger.

Sul telecomando si preme la funzione che si desidera attivare (UP, STOP, DOWN); possono essere memorizzate fino ad un massimo di tre funzioni. Per modificare le operazioni salvate nel modulo Slave è necessario prima resettarlo e successivamente ripetere le operazioni di programmazione.

GESTIONE DELLA RETE TRAMITE TELECOMANDO

Tramite i pulsanti CH- e CH+ è

possibile selezionare lo Slave al quale inviare un comando; il relativo LED si illuminerà. Se tutti i LED sono accesi, significa che il comando viene a tutti i moduli Slave (*broadcast*).

Se lo Slave è configurato per la gestione di un motore, usando i pulsanti UP, STOP, DOWN è possibile attivare la corrispondente funzione. Se sono stati cablati anche i finecorsa, lo Slave provvederà a fermare il motore di conseguenza, funzione, questa, utile per la gestione di una tenda da sole in modo autonomo.

Se nel telecomando è inserito il jumper MODE, con i pulsanti UP e DOWN si potrà regolare

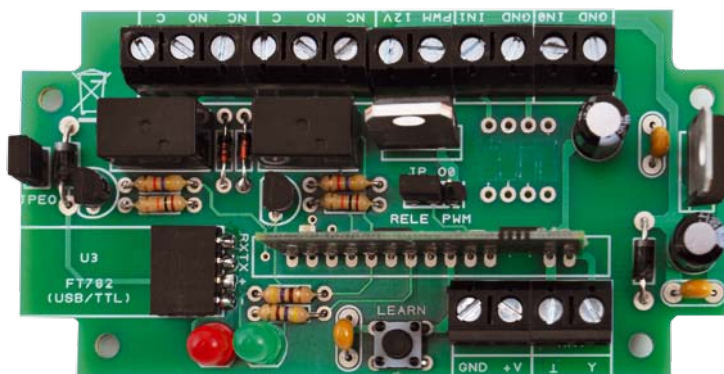
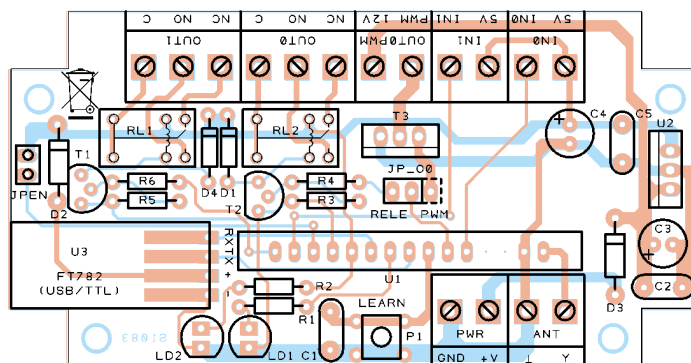
[piano di MONTAGGIO]

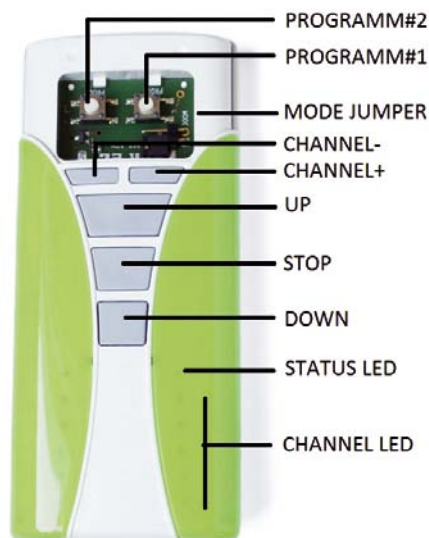
Elenco Componenti:

R1, R2: 470 ohm
 R4, R6: 4,7 kohm
 R3, R5: 10 kohm
 C1, C2: 100 nF multistrato
 C3, C4: 220 µF 25 VL elettrolitico
 C5: 100 nF multistrato
 T1, T2: BC547
 T3: STP36NF06
 P1: Microswitch
 LD1: LED 5 mm verde
 LD2: LED 5 mm rosso
 U1: Modulo RFTIDE (RFT-868-5V)
 U2: 7805
 D1, D4: 1N4148
 D2, D3: 1N4007
 RL1, RL2: Relé miniatura 5V

Varie:

- Morsetto 2 poli (5 pz.)
- Morsetto 3 poli (2 pz.)
- Strip maschio 2 poli (1 pz.)
- Strip maschio 3 poli (1 pz.)
- Strip femmina 4 poli 90° (1 pz.)
- Jumper (2 pz.)
- Antenna accordata 868 MHz
- Circuito stampato
- Alimentatore 12 Vdc





Comandi e impostazioni del Master Remote Control.

il duty-cycle dell'uscita PWM, mentre con il pulsante di STOP si potrà attivare o disattivare l'uscita PWM (comando PWM_ON e PWM_OFF).

Se azioniamo il comando di UP o DOWN, dal Master viene inviata periodicamente una richiesta di stato del motore sino a quando il motore non viene fermato tramite il comando di STOP o a causa dell'eccitazione di un finecorsa. Dopo aver inviato un comando, il Master attende risposta dallo Slave; durante la richiesta dello stato di funzionamento, il LED di notifica del telecomando assume il colore arancione. A seconda dello stato del motore ricevuto dallo Slave corrispondente, muterà la propria colorazione.

RESET DELLO SLAVE

È possibile ripristinare i valori di fabbrica di ciascun dispositivo Slave tramite la seguente procedura:

1. premere il pulsante LEARN per circa due secondi sullo Slave, oppure il pulsante di PROG1 sul telecomando subito dopo averlo attivato premendo i pulsanti CH+ o CH-;
2. premere nuovamente il pulsante di LEARN per altri due secondi, allorché il LED inizia a lampeggiare (dopo due secondi il LED si spegne ad indicare che la procedura di reset è terminata).

USO COME RIPETITORE

Per configurare un modulo Slave come ripetitore di segnale, si deve premere -su di esso- il pulsante di LEARN per circa due secondi, allorché il LED del modulo si accende. Entro quindici secondi si seleziona sul telecomando la funzione BROADCAST MODE (tutti i LED di canale accesi) e quindi si preme il pulsante PROG1. Il LED dello Slave lampeggia per alcuni secondi e poi si spegne se la procedura è andata a buon fine.

Come avete visto, creare una rete è davvero semplice e con il telecomando si possono gestire funzioni in modo semplice e intuitivo; allo stesso modo possono essere associati e gestiti sino a 5 nodi. Principalmente abbiamo creato una rete idonea alla gestione di qualche tenda o tapparella e niente più, tuttavia, alcuni moduli possono interagire tra loro per attivare direttamente utilizzatori.

USO DEL RFTide USB DONGLE

Si tratta di un modulo RFTide serializzato al quale è stato aggiunto un convertitore USB-seriale interfacciabile direttamente ad

un PC; anche il circuito FT1083K con a bordo un modulo Serializzato ed il convertitore USB-Seriale svolge di fatto le stesse funzioni del dongle USB. Appena inserito, il dispositivo, è riconosciuto come porta seriale virtuale similmente a quanto avviene con Arduino. Nell'area download del sito Aurel, previa registrazione, è possibile scaricare gratuitamente il software (per Windows) che permette la gestione completa di tutta la rete e con il quale è possibile configurare i parametri, che con il solo telecomando non è possibile modificare.

Appena inserita la periferica,

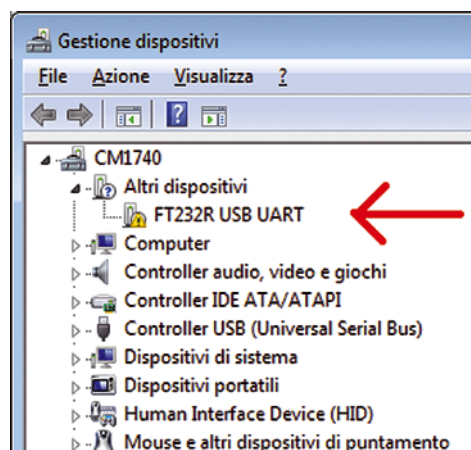
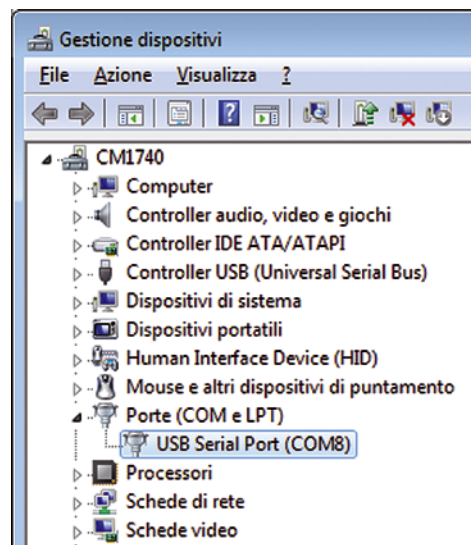


Fig. 1 - Installazione driver per dongle USB.



Colori del LED di notifica del Remote Control.

Profilo usato	LED arancione	LED blu	LED rosso	LED verde
motore	in attesa di risposta	motore in movimento e finecorsa non attivi	finecorsa in chiusura raggiunto	finecorsa in apertura raggiunto
Print	in attesa di risposta	-	PWM ON	PWM OFF

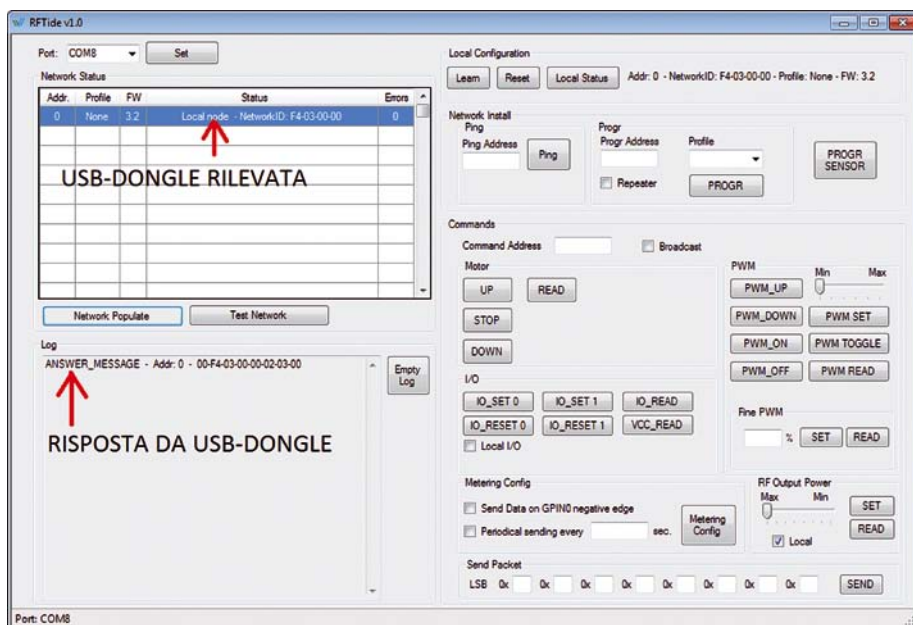


Fig. 2 - Popolazione della rete rilevata con connesso solo il dongle USB.

Windows cerca di installare i driver; se la procedura fallisce è necessario installare i driver manualmente accedendo alla "gestione dispositivi" del sistema operativo (vedere Fig. 1). I driver sono disponibili all'indirizzo <http://www.ftdichip.com/Drivers/VCP.htm>.

Il software RTFide non necessita di installazione ed è sufficiente avviarlo perché sia operativo; cliccando il pulsante *network populate* il software interroga la rete e crea un elenco dei moduli presenti con il profilo ad essi associato (Fig. 2). Se non ci sono altri moduli attivi, l'elenco comprende il solo modulo Master configurato all'indirizzo zero; non sono rilevati eventuali altri slave associati con un altro telecomando, perché facenti parte di una rete diversa.

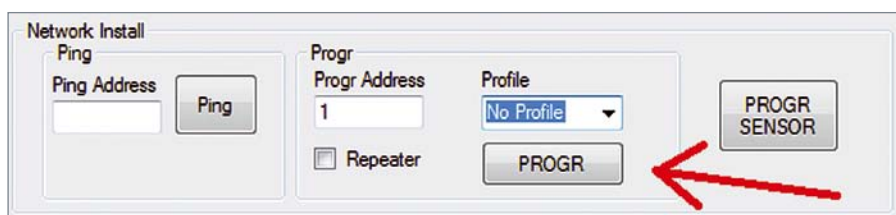
Per associare un modulo Slave alla rete (Fig. 3) basta seguire questa semplice procedura di LEARN:

1. premiamo il pulsante di LEARN sullo Slave che vogliamo includere nella rete, allorché il LED si accende);
2. nella schermata del software, nel riquadro *network install*

specifichiamo il tipo di profilo desiderato, quindi clicchiamo su **PROGR**.

Se il LED dello Slave lampeggia per alcune volte, il modulo è associato correttamente. Se vogliamo fare delle semplici prove non impostiamo alcun profilo (*no profile*) in modo da avere la libertà di attivare tutte le funzioni da remoto tramite il software, infatti in questo caso potremmo impostare manualmente sul modulo remoto lo stato delle uscite oppure leggere lo stato degli ingressi.

Fig. 3 - Programmazione di un nuovo modulo con software RFTide-v1.0



nella casella *prog address* inseriamo l'indirizzo che vogliamo assegnare al nuovo modulo e

Per verificare che tutto sia a posto interroghiamo il modulo inserendo nella casella *Ping Address*,

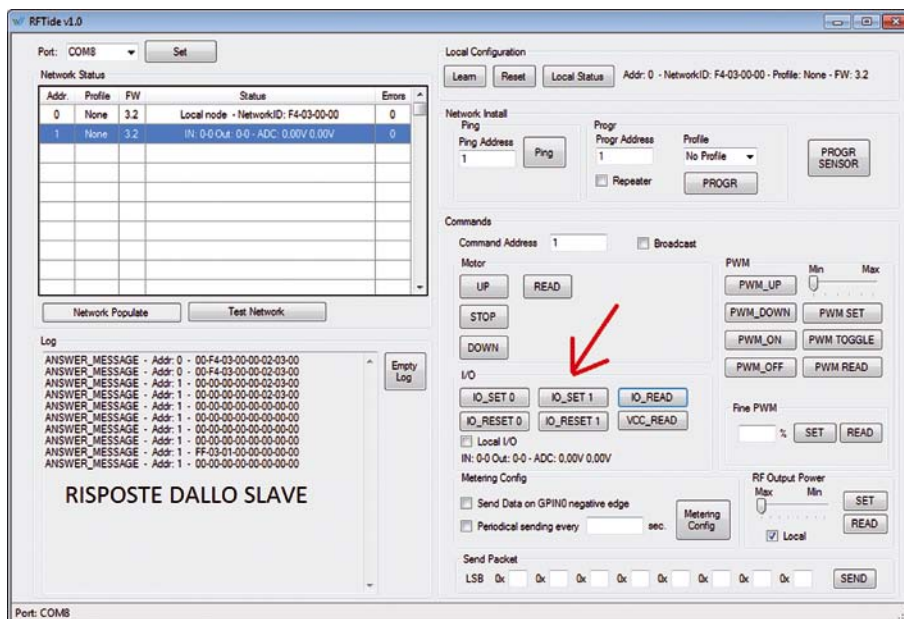
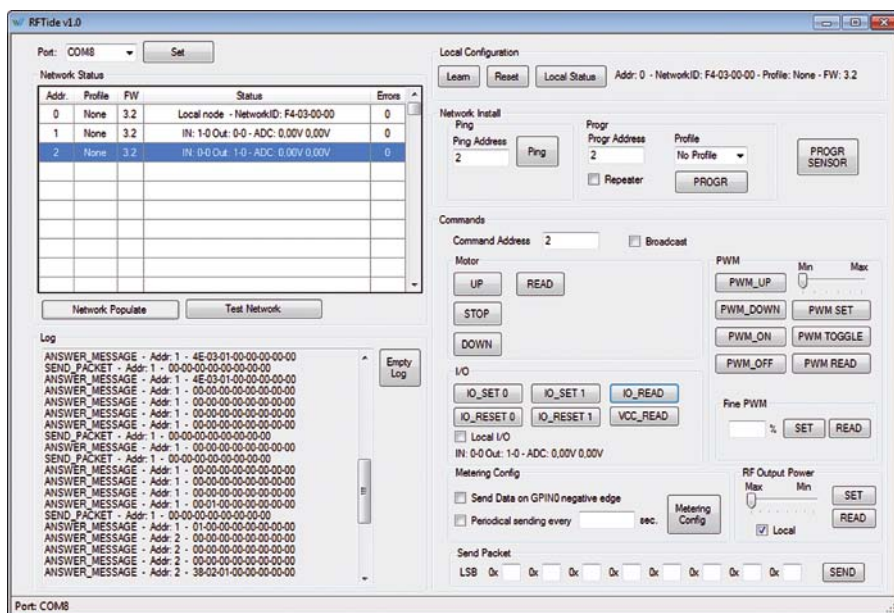


Fig. 4 - Gestione di GPIO remoti tramite software RTide-v.1.0.



Rete con tre moduli e traffico delle comunicazioni.

del riquadro *Ping*, l'indirizzo del modulo (1 nel nostro caso) e clicchiamo sul pulsante *PING*: otterremo lo stato di funzionamento del modulo. Possiamo interagire con il modulo usando i comandi del riquadro *I/O* (Fig. 4) attivando o disattivando le uscite o leggendo lo stato degli ingressi; pur riportando il valore dei due ADC i valori saranno sempre zero (solo per il modulo a 5V).

Allo stesso modo potremmo associare un altro modulo alla rete. Nel riquadro *network install* alla voce *ping address* specificate l'indirizzo del modulo con cui volete interagire e cliccate su *Ping* per interrogare il modulo e mostrarne lo stato di funzionamento ed il profilo impostato. Per modificare il profilo usate il riquadro *Prog*, inserendo nuovamente nel campo *Prog Address* l'indirizzo del dispositivo e il profilo desiderato. Ricordatevi che può essere programmato un solo modulo per volta, altrimenti più moduli avranno lo stesso indirizzo.

Se il modulo è configurato come *motor* potete usare i comandi del riquadro *Motor* per interagire con il modulo; il pulsante *READ* permette di sapere lo stato dei finecorsa e

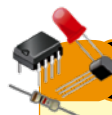
del motore mentre con i pulsanti *UP*, *DOWN*, *STOP* è possibile gestire lo stato del motore. Se avete selezionato il profilo *PWM* potete usare il riquadro *PWM* per settare i valori o per modificare i valori predefiniti. Il software proposto dall'Aurel è solo di esempio e permette di capire appieno le potenzialità dei loro moduli; nulla vieta di realizzare un software con un'interfaccia personalizzata per l'utilizzo in una specifica applicazione; per questo è però necessario studiarsi attentamente il protocollo di comunicazione riportato nella *RFTide-User-Guide*, disponibile nel sito dell'Aurel.

Qualsiasi modulo non serializzato connesso ad una logica programmabile può svolgere funzione di Master però la rete avrà necessariamente Network ID 0x0000 e non sarà possibile realizzare reti indipendenti. Non è possibile modificare il Network ID perché altrimenti potrebbero verificarsi delle spiacevoli interferenze tra reti; se, ad esempio, in una palazzina un appartamento avesse un certo ID, è necessario sia assicurato che l'appartamento attiguo abbia un diverso ID pena spiacevoli interferenze. Per questo motivo

esistono i moduli serializzati che dispongono (assegnato in fabbrica e non modificabile) un proprio e unico Network ID con i quali realizzare la funzione di Master ed istruire una nuova rete. Tutti gli Slave associati con un determinato Master saranno istruiti per colpire solo con altri moduli che hanno Network ID uguale a quello del Master.

CONCLUSIONI

Per questa puntata ci fermiamo qua. Avete visto come, con i prodotti RFTide, la realizzazione di una rete mesh sia estremamente facile. Nella prossima puntata vedremo come utilizzare Arduino per creare e gestire in modo semplice ed economico una rete basata sui moduli Aurel e vi illustreremo l'RFTide shield, ossia una scheda in grado di ospitare un modulo RFTide e farlo gestire mediante Arduino.



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità; il master del circuito stampato può essere scaricato gratuitamente dal sito della rivista (www.elettronicaonline.it). Il circuito descritto in questo articolo è disponibile in scatola di montaggio al prezzo di 34,00 Euro (cod. 1083K); il kit comprende la basetta forata e serigrafata, tutti i componenti ed il modulo radio RFT-868-5V. Non sono compresi l'antenna AG868 (9,00 Euro) né il convertitore USB/Seriale FT782M (15,00 Euro) che vanno acquistati separatamente. 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>

PRIMAVERA DI ROBOTICA

Tre eventi tengono banco in questi mesi

di DAVIDE SCULLINO

Con l'arrivo della bella stagione tornano eventi e manifestazioni che fanno fiorire l'interesse per la robotica. Sotto i riflettori c'è la **FIRST LEGO League (FLL)**, che il 14 e 15 marzo vede in scena la fase finale dell'edizione italiana, cui parteciperanno i team che hanno superato le qualificazioni regionali che si sono svolte il 18 gennaio a Pistoia per il Centro Italia, il 26 gennaio a Cuneo (per il Nord-Ovest), il 2 febbraio a Udine (per il Nord-Est) e il 6 febbraio a Catania per il Sud Italia.

FIRST LEGO League è una manifestazione nata per promuovere la robotica educativa ed è il risultato della collaborazione tra **FIRST (For Inspiration and Recognition of Science and Technology)** e il gruppo **LEGO**. È un campionato a squadre, tra giovanissimi (dai 9 ai 16 anni) che progettano, costruiscono e programmano robot autonomi; in realtà è un progetto di ampie dimensioni, che aiuta i giovani a scoprire il divertimento nella scienza e nella tecnologia, sviluppando nel contempo autostima, conoscenza, competenze valide per il lavoro e per la vita. FLL sfida i giovani a pensare come scienziati e ingegneri: insieme a tutor adulti, le squadre della FLL risolvono problemi reali usando concetti ingegneristici e matematici e servendosi di tecnologie LEGO MINDSTORMS.

FLL si disputa in tutto il mondo e attualmente coinvolge 61 nazioni appartenenti a 5 continenti; le qualificazioni partono dalla fase regionale, per proseguire in quella nazionale che fornisce l'accesso alle manifestazioni internazionali, continentali e mondiali. Quest'anno, i team che si aggiudicheranno la FLL Italia parteciperanno alla finale mondiale -World Festival- prevista dal 23 al 26 aprile a St. Louis (Stati Uniti).

Quest'anno la competizione pren-



de il nome di **Nature's Fury** e prevede che le squadre partecipanti debbano identificare una comunità che potrebbe subire una possibile calamità naturale, identificare un problema che va affrontato quando si verifica il disastro e sviluppare una soluzione innovativa che aiuti le persone a prepararsi all'evento, a rimanere al sicuro e a ricostruire. La manifestazione si articola in tre prove, la prima delle quali è la gara di robotica: le squadre dovranno realizzare e programmare un robot autonomo che realizzi il mag-

assegnata. L'edizione 2013/2014 della FLL gode del supporto scientifico di OGS (Istituto Nazionale di Oceanografia e Geofisica Sperimentale).

Ai fini della classifica ogni prova ha lo stesso peso, risulterà quindi nei primi posti il team che avrà ben figurato nelle tre prove e si sarà distinto per rispetto dei valori fondamentali (core values).

Per la FLL Italia, le fasi regionali ormai concluse hanno fornito ciascuna i nominativi dei team che accedono alla finale nazionale:

tra robot realizzati dai giovani delle scuole, promossa dalla Fondazione Mondo Digitale con il contributo della Camera di Commercio di Roma. In scena prototipi robotici di ultima generazione, competizioni e sfide amichevoli tra giovani di ogni età per le selezioni nazionali, ma anche laboratori didattici per far conoscere a tutti le potenzialità della robotica. Oltre alle competizioni del VII RomeCup, l'evento ospiterà le selezioni nazionali di alcune categorie della Robocup Junior. La manifestazione capito-



gior numero delle missioni presenti sul tavolo di gara in 150 secondi (il percorso di gara viene fornito ai concorrenti almeno due mesi prima, in modo che questi possano testare le soluzioni adottate con largo anticipo e scegliere la strategia di gara adeguata); la seconda prova prevede la documentazione e presentazione delle soluzioni robotiche adottate a una giuria di esperti (il team presenta con la modalità che preferisce le soluzioni adottate per risolvere alcune missioni rispondendo ai quesiti dei giurati ed effettuando dimostrazioni del funzionamento del robot). La prova conclusiva consiste nella realizzazione e presentazione di una ricerca scientifica sulla tematica

per il Centro Italia accedono alla finale i team Flood Buster, DBP Team Mirandola, Miccorobokids, Erre Come Robot e 2E_Xtreme. I qualificati per il Nord-Ovest sono i team Maserati, Maserati 1.0, Melting Pot, Firestorm, VVCalci e Calasanzio. Passano la fase di qualificazione di Udine, le squadre Team A, The Legend is Now, GAF, Team B, RAIROB, SCALCERLE A e MARCONI B. Infine, dalla fase di qualificazione del Sud Italia passano le squadre Pluchirobot, Villaggio dei ragazzi, Sigonella Robotics Team e Bosco Alife Deltacon.

RomeCup 2014 - Roma

A marzo torna anche **RomeCup**, ottava edizione della competizione

lina torna con la classica formula dei tre eventi in uno: competizioni, area dimostrativa con laboratori e workshop e fasi finali con cerimonia di premiazione.

RomeCup riunisce le eccellenze della robotica italiana nella capitale dal 19 al 21 marzo 2014 e quest'anno propone novità a partire dalle location: il Tempio di Adriano a Piazza di Pietra, il Campidoglio e l'istituto professionale Carlo Cattaneo. Roma accoglierà l'eccellenza della robotica nel tempio romano eretto in nome dell'Imperatore Adriano: un impero che fu caratterizzato da efficienza e splendore...le parole chiave di questa nuova e promettente edizione della RomeCup. Nel cuore della

Capitale prenderà forma una città della robotica: mentre i team di scuole e università si sfideranno nelle diverse categorie, i cittadini potranno scoprire le potenzialità della robotica educativa e di servizio. Quest'anno l'evento verrà raccontato anche su Twitter con l'hashtag #romecup2014 e al racconto potranno partecipare tutti i team, sparsi in ogni parte del territorio, per dare vita alla robotica in crowdsourcing.

Durante i primi due giorni, nel Tempio di Adriano si terranno le selezioni nazionali e la competizione per il "Trofeo Internazionale Città di Roma di Robotica" nelle categorie soccer, dance, rescue, explorer e cospace theatre. In totale verranno assegnati nove premi, uno per ogni categoria in gara. Nell'apposita area dimostrativa, aziende, centri di ricerca, università e scuole di ogni ordine e grado porteranno l'eccellenza italiana della robotica a Roma e offriranno ai visitatori laboratori e workshop per studenti e docenti. È prevista anche l'esibizione di umanoidi NAO a cura del Dipartimento di Ingegneria Informatica Automatica e Gestionale "A. Ruberti" - Università Sapienza

di Roma. Il terzo giorno si terranno in Campidoglio, presso la Sala della Protomoteca, le fasi finali delle competizioni e la cerimonia di premiazione delle squadre vincitrici, con la partecipazione di importanti ospiti del mondo accademico e istituzionale. RomeCup 2014, VIII Trofeo Internazionale Città di Roma di Robotica, si articolerà nelle seguenti gare:

- Soccer (primary e secondary);
- Rescue (primary e secondary);
- Dance (primary e secondary);
- Explorer (junior e senior).

Per quanto riguarda le Selezioni nazionali Robocup Junior, sono previste le gare Soccer (primary e secondary) e Cospace Theatre.

Robot & Makers Show

Altro appuntamento di primavera, questa volta fieristico, è in programma per il 29 e 30 marzo al Parco Esposizioni di Novegro (MI), dove si terrà la prima edizione di **Robot & Makers Show** - mostra mercato della robotica e dei makers.

La manifestazione, cui saremo presenti anche noi di ElettronicaIn con i nostri progetti robotici e per makers, rappresenta l'esposizione dei



più importanti distributori italiani e internazionali di robot e di robotica. In un padiglione di 2.400 metri quadrati, all'insegna dell'innovazione tecnologica e del divertimento, prenderà vita la fiera mercato per veri "smanettoni" nella quale decine di espositori offriranno, nei loro banchi, materiale per costruire in casa il vostro robot.

Makers e inventori metteranno in mostra le loro creature per divertirsi e far divertire: robot musicisti e ballerini, robot di pubblica utilità ecc. Inoltre, stampa creativa in 3D, workshop e laboratori aperti a tutti, sezioni didattiche: un paradiso per gli hobbysti che vogliono imparare tutto sul fai da te.

La prima edizione di Robot & Makers Show viene inaugurata in concomitanza con RADIANT & SILICON, la fiera dell'elettronica che si svolge da oltre 20 anni presso il Parco Esposizioni di Novegro, con orario dalle 9 alle 18 il sabato e dalle 9 alle 17 la domenica. □



Ecco Taranis, l'UAV della Regina

In un mondo che vede i principali operatori dell'aviazione militare contendersi i cieli a suon di UAV, ha fatto la sua comparsa anche Taranis, lo stealth inglese sviluppato da British Aerospace per la Difesa britannica. Per ora è solo un prototipo in fase di sviluppo, ma potrebbe a breve diventare il primo drone da combattimento Stealth made in UK; lo sviluppo del velivolo -il cui nome deriva da quello del dio del tuono nella mitologia celtica- è costato 185 milioni di sterline (circa 223 milioni di euro), stanziati dal Ministero per la Difesa inglese e da BAE Systems. La progettazione e realizzazione hanno richiesto circa un milione e mezzo

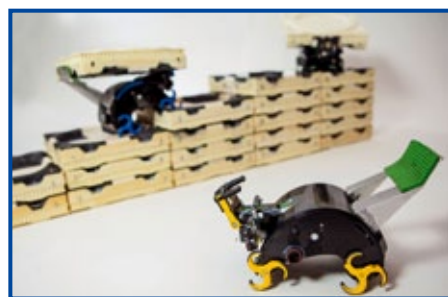
di ore di lavoro e sono frutto della collaborazione di 250 società inglesi. Il drone, la cui esistenza è stata svelata al mondo da poco, è in realtà stato collaudato l'anno scorso (il 10 agosto 2013), con un volo durato 15 minuti in una zona tenuta segreta. Sotto il controllo remoto del pilota collaudatore Bob Fraser, il velivolo dimostratore ha compiuto un perfetto decollo, una completa rotazione in rollio, una decisa cabrata e un atterraggio. A quello di collaudo sono seguiti, nel corso dell'anno passato, una serie di voli di durata via-via crescente, fino ad un'ora, a varie altitudini e velocità di crociera. A regime, il velivolo dovrebbe

LE CYBER TERMITI LAVORANO SODO

Prendendo l'ispirazione dalla natura, studiosi dell'Harvard School of Engineering and Applied Sciences (SEAS) e del Wyss Institute for Biologically Inspired Engineering at Harvard University, hanno creato uno sciame di robot che come le termiti sono capaci, in completa autonomia e sulla base di un programma preventivamente impostato, di costruire strutture tridimensionali complesse. Durata quattro anni, questo progetto ha dimostrato la possibilità di realizzare gruppi di robot "operai" cooperanti capaci di costruire, ad esempio, barriere contro gli allagamenti o erigere costruzioni su altri pianeti, ad esempio su Marte, oppure di edificare in ambienti ostili all'uomo.

I TERMES robot, così vengono chiamati i piccoli automi (grandi

come un mattone pieno), possono trasportare mattoni salendo scale, arrampicandosi (grazie alle "zampe a ruota" uncinati di cui dispongono) e scavalcare ostacoli, per poi andarli a deporre, ad esempio, in cima alla costruzione che gli altri simili stanno realizzando. I mattoni vengono caricati grazie a due zampe anteriori simili alle braccia di un muletto e caricati sulla schiena, dove vengono fermati posteriormente da quella che appare una coda sollevata. La particolarità dei robot termiti è che, similmente a quanto avviene in natura, non eseguono un programma predefinito con una gerarchia, ma sono cooperanti in quanto si adattano alla situazione del momento: vedono a che punto è il lavoro e svolgono le fasi che mancano, quindi sanno come dev'essere fat-



ta la costruzione ed aggiungono i mattoni mancanti. Se in un posto è stato appena inserito un mattone, depongono quello che trasportano nel successivo. A ciò provvede l'apposito software, che comunica il modello matematico della costruzione, in base al quale, grazie a specifici sensori, ogni robot valuta lo stato di avanzamento del lavoro e posiziona il proprio mattone.

<http://blogs.unb.ca/newsroom>



viaggiare a velocità supersonica senza essere rilevato dai radar, e coprire distanze intercontinentali (grazie all'autonomia di cui disporrà). In futuro potrà trasportare armi, anche se ufficialmente è stato sviluppato per scopi di sorveglianza e raccolta di informazioni.

www.baesystems.com



IL ROBOT MANIPOLATORE È LOW-COST

Unbounded Robotics ha da poco presentato UBR-1, un robot finalizzato alla ricerca universitaria e all'automazione industriale, capace di manipolare oggetti con precisione e compiere varie lavorazioni al posto dell'uomo. La novità di questo progetto è che è stato finalizzato all'abbattimento del prezzo d'acquisto: infatti il robot viene proposto ad appena 35.000 dollari, contro i 300.000÷400.000 che sistemi del genere costano.

Il robot pesa 68 kg ed ha un'altezza variabile (così da prendere oggetti in basso e portarli in alto, per esempio...) che arriva a un massimo di 1,3 metri; è dotato di un braccio con sette gradi di libertà, capace di sollevare 1,5 kg, i cui movimenti sono guidati da uno scanner detto PrimeSense, che raccoglie le informazioni dall'esterno. In dotazione, anche uno scanner secondario, inserito nella base, che consente ad UBR-1 di riconoscere ed evitare gli ostacoli (e gli umani) mentre si muove; questo scanner secondario e i bordi arrotondati

del robot, nascono nell'ottica di una progettazione mirata a consentire al robot di evitare incidenti quando lavora in ambienti frequentati dall'uomo. UBR-1 è un "mobile manipulation platform", cioè un automa capace di muoversi da sé, percepire l'ambiente in cui si muove e di manipolare oggetti.

La sua piattaforma di controllo è un computer basato su processore Intel Core i5 di quarta generazione, 8 GB di RAM e 120 GB di HD, sul quale gira un sistema operativo open source ROS sviluppato da Willow Garage.

Il sistema di controllo permette di comandare il robot manualmente mediante un controller della PS3, oppure gestirlo tramite il software appositamente sviluppato. Pur essendo basato sulla piattaforma ROS, quest'ultima viene utilizzata solo come framework, con la possibilità di impartire i comandi all'UBR-1 senza dover programmare ogni singola attività.

<http://robotnor.no>



RICEVITORE 1 E 2 CANALI

ALESSANDRO SOTTOCORNOLA

Ricevitori mono e bicanale
a 433,92 MHz capaci di
essere abbinati a un
massimo di 10 TX ciascuno.
Monostabile o bistabile.



Quella basata sull'integrato MM53200 è una delle più datate e collaudate codifiche adottate nei radiocomandi generici e per apricancello, tanto che molti trasmettitori reperibili in commercio per tali applicazioni la adottano; essendo in commercio da molto tempo, comporta il problema che gli integrati codificatori si sono nel tempo estinti e sono stati sostituiti ad esempio dall'UM3750 e poi dall'UM86409 (entrambi di produzione UMC). Per questa ragione è buona cosa

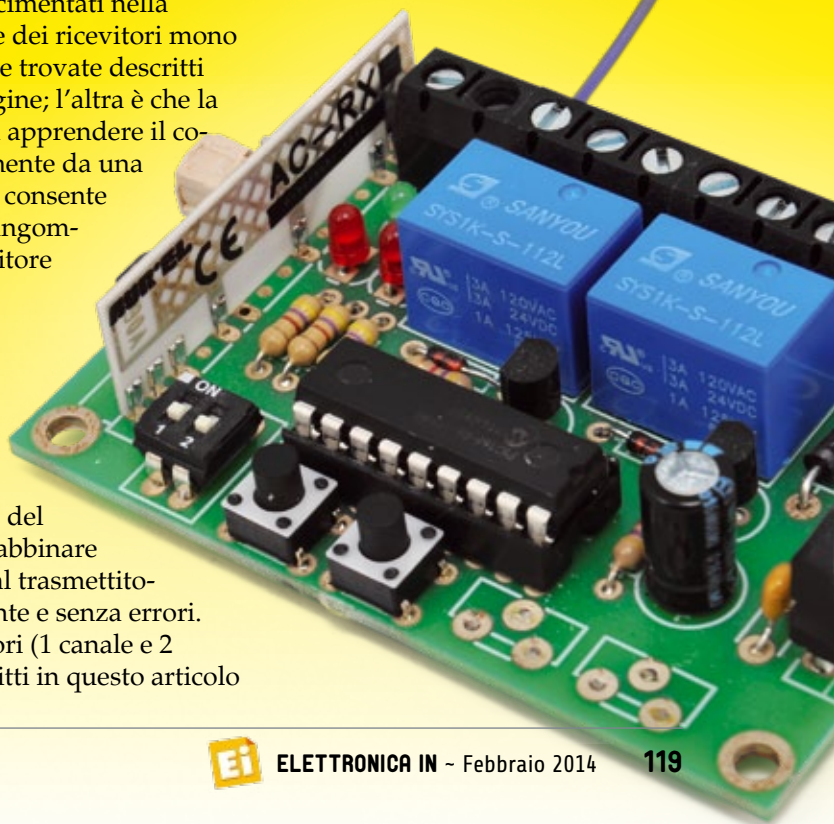
procurarsi un ricevitore a microcontrollore in grado di adattarsi a questa codifica, apprendendola in modo automatico; ciò mette al riparo dall'estinzione dei chip e consente altresì di replicare i ricevitori anche senza disporre dei decoder.

Questa è una delle ragioni per cui ci siamo cimentati nella realizzazione dei ricevitori mono e bicanale che trovate descritti in queste pagine; l'altra è che la possibilità di apprendere il codice direttamente da una trasmissione consente di limitare l'ingombro del ricevitore (perché non servono i dip-switch a 12 poli che invece l'impostazione manuale richiederebbe) del circuito e di abbinare il ricevitore al trasmettitore rapidamente e senza errori. I due ricevitori (1 canale e 2 canali) descritti in questo articolo

sono compatibili, oltre che con gli encoder MM53200, UM86409 e UM3759, anche con i più recenti Holtek HT-12, che hanno un'impostazione dei bit di codifica compatibile.

SCHEMA ELETTRICO 1 CH

Partiamo dall'analisi del circuito monocanale, basato su un



microcontrollore PIC16F683 e su un modulo ibrido radiorecettore AC-RX2, accordato a 433,92 MHz; il modulo U2 contiene la parte radio del circuito ed è un ricevitore specifico Aurel AC-RX2 provvisto di amplificatore del segnale d'antenna (che conferisce una sensibilità di ben -106 dB), stadio di sintonia superrigenerativo sintonizzato a 433,92 MHz mediante compensatore tarato in fabbrica e dotato di filtro RF (il filtro serve a migliorare la selettività, che nei superrigenerativi non è elevata) e demodulatore d'ampiezza. Completa la dotazione del modulo, uno squadratore a comparatore del segnale digitale (a livello TTL) uscente dal piedino 14 e un amplificatore LF del segnale in uscita dal demodulatore AM. Subito dopo l'accensione, il micro inizializza i propri I/O impostando GP0 e GP1 come uscite utilizzate, rispettivamente, per comandare il LED di segnalazione LD2 (che segnala sia il modo di funzionamento, sia lo stato della procedura di autoap-

prendimento) e il transistor NPN T1 che pilota il relé di comando del carico. Quest'ultimo viene eccitato dallo stato di saturazione del transistor, a sua volta polarizzato in base quando GP1 si trova a livello logico alto (R1 limita la corrente di base del T1 ad evitare che la giunzione base-emettitore venga danneggiata); ogni volta che il transistor conduce, oltre alla bobina del relé alimenta R2 e LD1, che ci fornisce la segnalazione di relé attivo. In parallelo alla bobina del relé si trova anche il diodo D2, la cui funzione è cortocircuitare l'extratensione inversa che la bobina genera quando il transistor, andando in interdizione, le sospende la corrente; senza diodo, la tensione si chiuderebbe sulla giunzione base-collettore del T1, danneggiandola in breve tempo.

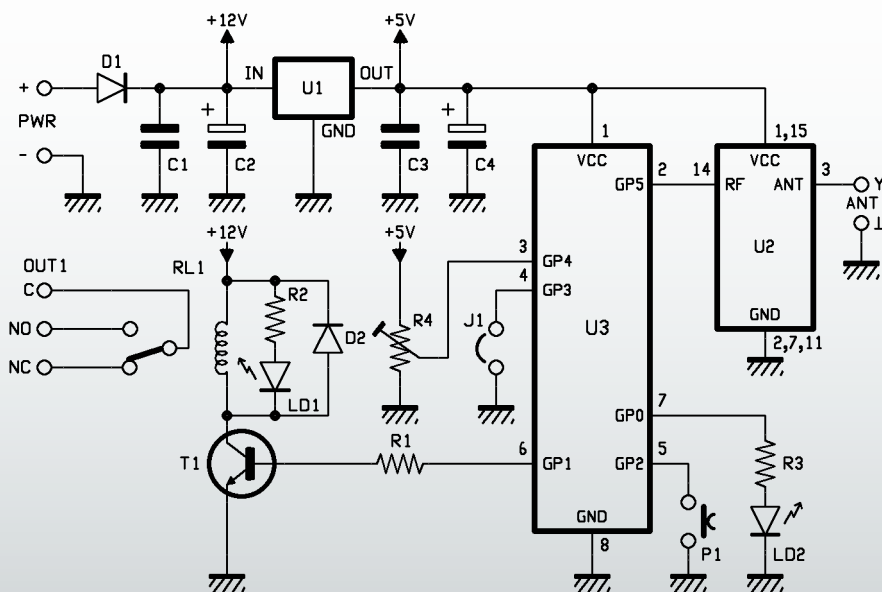
Continuando con l'inizializzazione, il PIC setta GP2, GP3 e GP5 come ingressi, dedicati rispettivamente alla lettura del pulsante P1, del jumper J1 e dell'uscita dati dell'AC-RX2; per le prime due è

CARATTERISTICHE TECNICHE

- **Numero di uscite: 1-2**
- **Modalità uscita: Monostabile, Bistabile**
- **Alimentazione: 12 Vcc**
- **Assorbimento: 40 mA max.**
- **Memoria: 10 codici per canale**
- **Codifica: MM53200/HT-12**

attivo il pull-up interno. La linea GP4 è inizializzata come un input assegnato all'A/D converter ed è destinata a futuri sviluppi del firmware. Al momento è irrilevante sul funzionamento del circuito. Completa il circuito il blocco di alimentazione, che parte dai punti + e - PWR (la morsettiera di alimentazione) e, passando dal diodo D1 (che protegge dai danni causati dall'inversione di polarità) raggiunge il regolatore integrato a tre terminali U1, il cui ingresso è filtrato da C1 e C2, che riducono l'entità dell'eventuale ondulazione della tensione di alimentazione (C2) e dei disturbi ad RF e impulsivi captati dai fili di alimentazione (C1). Analoga funzione la svolgono, ma per l'uscita del regolatore, C3 e C4. Detto ciò, possiamo spiegare come funziona il ricevitore a 1 canale: fornendo alimentazione al circuito, il firmware, dopo l'inizializzazione degli I/O fa emettere a LD2 5 lampeggi, ad indicare il corretto avvio e il normale funzionamento, che corrisponde all'esecuzione del comando fornito a distanza; poi il main gira in loop verificando sia la variazione di livello sul piedino 4 (cioè la modifica dello stato del jumper J1) sia l'eventuale pressione del P1. Un'apposita routine testa periodicamente lo stato del piedino 2, ovvero l'arrivo di dati dall'AC-RX2; quando premiamo uno dei pulsanti sul trasmettitore, il segnale RF irra-

[schema elettrico del **RICEVITORE 1 CANALE**]



diato da quest'ultimo raggiunge l'antenna ricevente del modulo AC-RX2, il quale demodula la componente dati e la invia al piedino 14 per renderla disponibile al microcontrollore, che la legge dal piedino 2. Il micro funziona da decoder del codice corrispondente al tasto premuto sul telecomando (ma il firmware prevede anche una routine per apprendere in automatico in codice di base dello stesso senza impostare dip-switch) e opera in questo modo: il microcontrollore prende gli impulsi TTL, li colloca in RAM e li analizza con un'apposita routine firmware che prima di tutto discerne, tra i tanti segnali captati dall'etere, quello compatibile con il formato della codifica UM3750, quindi, in caso affermativo verifica se il codice è uno di quelli memorizzati durante la fase di autoapprendimento, mentre in caso contrario elimina i dati dalla RAM e si predispone a una nuova analisi.

Vedremo in un secondo momento come abbinare il trasmettitore

Tabella 1 - Significato dei LED nel ricevitore monocanale.

Diodo luminoso	Normale funzionamento	Programmazione
LD1	attività OUT1: acceso=RL1 eccitato spento=RL1 a riposo	-
LD2	all'avvio o all'uscita dall'apprendimento o cancellazione dei codici, segnala con 5 lampeggi l'avvio in modalità normale.	si accende per 2 secondi quando si entra in programmazione; lampeggia quando il circuito ha appreso il codice trasmesso; è fisso se l'apprendimento è fallito.

al circuito, ricorrendo all'autoapprendimento; per ora vi basti sapere che il firmware prevede di apprendere un massimo di 10 codici, anche di trasmettitori differenti. Infatti ogni codice viene memorizzato tutto, ossia tutti i 12 bit che lo compongono.

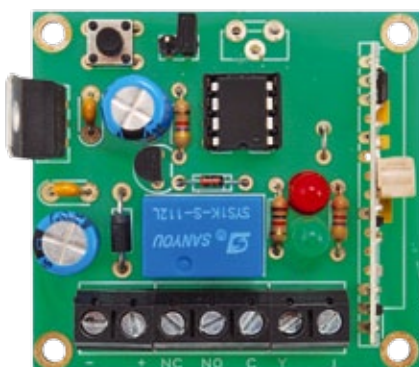
Ora vediamo cosa accade se il segnale ricevuto contiene uno dei codici appresi e mantenuti in EEPROM di lavoro dal PIC: in questa evenienza, si avvia l'apposita routine di gestione del relé, che determina azioni diverse a seconda che il modo di funzionamento sia monostabile o bistabile; l'impostazione della modalità si effettua con J1

(jumper aperto vuol dire funzionamento monostabile e jumper chiuso corrisponde al modo bistabile). Nel modo bistabile il relé cambia la propria condizione ogniquale volta il microcontrollore riconosce un codice valido, mentre in quello monostabile (o impulsivo) il relé si porta a livello alto alla ricezione di un codice valido e torna a riposo quando il pulsante corrispondente al codice viene rilasciato, ovvero al cessare del segnale che ne ha causato l'attivazione: in breve, il relé segue il pulsante. Se simultaneamente la trasmissione smette ma ne viene ricevuta una proveniente da un altro radiocomando

[piano di montaggio del **RICEVITORE 1 CANALE**]

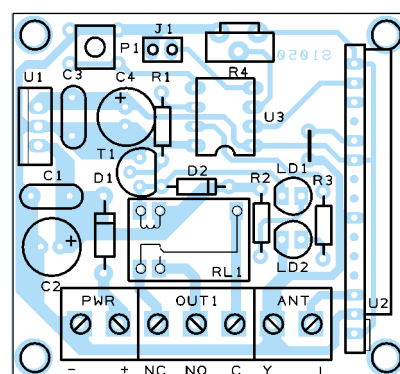
Elenco Componenti:

R1: 4,7 kohm
R2: 1 Kohm
R3: 470 ohm
R4: non montato
C1: 100 nF multistrato
C2: 100 µF 35 VL elettrolitico
C3: 100 nF multistrato
C4: 10 µF 35 VL elettrolitico
U1: 7805
U2: AC-RX2
U3: PIC12F683 (MF1050)
D1: 1N4007
D2: 1N4148
LD1: LED 5 mm rosso
LD2: LED 5 mm verde
T1: BC547
P1: Microswitch
RL1: Relè 12V



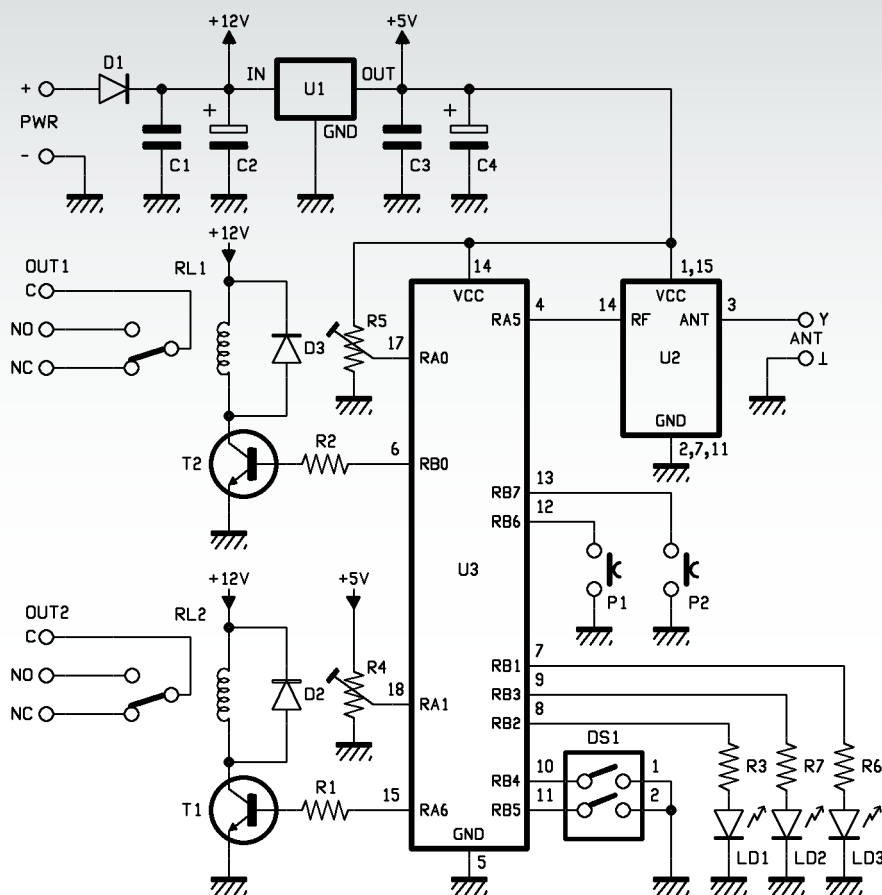
Varie:

- Morsetto 2 poli (2 pz.)
- Morsetto 3 poli
- Zoccolo 4+4



- Strip maschio 2 vie
- Jumper
- Circuito stampato

[schema elettrico del **RICEVITORE 2 CANALI**]



che è stato abbinato (ovvero il cui codice sia uno dei 10 appresi) entro il time-out concesso dal firmware, il microcontrollore considera che la trasmissione non si sia mai interrotta. Se giunge il codice di un altro TX quando il relé è attivo perché sta ricevendo la trasmissione da un altro trasmettitore valido, il discorso non funziona più perché il secondo interferisce con il segnale del primo e il ricevitore radio demodula il battimento tra i due stream di dati; come risultato, il microcontrollore ritiene che sia cessato il segnale di attivazione e fa ricadere il relé. Notate che la modalità di funzionamento dell'uscita può essere cambiata anche durante il funzionamento, cioè non serve spegnere e riavviare il circuito,

in quanto lo stato del jumper J1 viene letto in continuazione.

PROCEDURA DI APPRENDIMENTO MODULO 1 CH

Visto come il ricevitore si comporta quando arriva un comando, vediamo ora come inserire i codici, ovvero come si svolge l'apprendimento; ciò può essere fatto in ogni momento premendo e mantenendo premuto il pulsante P1 fin quando il LED verde (LD2) si accende: allora è iniziata la fase di autoapprendimento. A questo punto bisogna trasmettere con il ricevitore di cui apprendere il codice; se il TX dispone di più pulsanti e quindi più canali, si possono apprendere i codici di tutti; dunque, bisogna premere il pulsante del radiocomando che si desidera apprendere, e attendere

che il LED lampeggi, segno che l'apprendimento è andato a buon fine. Se il LED rimane acceso a luce fissa, significa che la memoria è piena o il codice trasmesso non è valido (non ha il formato richiesto perché inviato da un TX con codificatore MM53200, UM3750, UM86409 o HT-12).

Fate attenzione ad un dettaglio: per come funziona il firmware e per come viene gestita la EEPROM del microcontrollore, il circuito può memorizzare i codici dei trasmettitori senza preclusioni, fuorché per quella che prevede tutti i bit ad 1 (tutti i dip-switch del radiocomando impostati su ON); quindi del trasmettitore potranno essere usate 4.095 invece di 4.096 combinazioni.

Volendo cancellare i codici, va detto che la procedura è prevista ma non è possibile operare individualmente, nel senso che la memoria si può cancellare tutta e non in parte. Per cancellare la EEPROM e quindi rimuovere i codici già appresi, bisogna togliere alimentazione al circuito e riaccenderlo dopo aver premuto il pulsante P1; quest'ultimo va rilasciato solo quando il LED verde rimane acceso fisso per 2 secondi, ad indicare l'avvenuta cancellazione della memoria. A questo punto si può rilasciare il pulsante: il LED verde emetterà 5 lampeggi ad indicare l'uscita dalla procedura di cancellazione e l'avvio normale del ricevitore (Tabella 1).

SCHEMA ELETTRICO 2 CH

Diamo ora uno sguardo al circuito a 2 canali, senza però ripetere quanto già spiegato per il ricevitore monocanale; dando uno sguardo allo schema elettrico notiamo sempre lo stadio di alimentazione per il quale vale il discorso fatto per lo schema monocanale. Troviamo il solito

stadio radioricevente, anch'esso uguale. Il microcontrollore è differente, in quanto si tratta di un PIC16F688, la cui scelta è stata dettata dal maggior numero di I/O di cui dispone rispetto al PIC16F683 impiegato nella versione monocanale.

Ovviamente abbiamo due stadi a relé, funzionanti come già descritto, ed altrettanti diodi di protezione in parallelo alle relative bobine.

Troviamo anche un dip-switch a due vie, che serve ad impostare la modalità di attivazione delle uscite dei due canali, e due trimmer, che come detto nella descrizione del circuito monocanale, al momento sono destinati a futuri sviluppi e non vengono utilizzati. Subito dopo l'accensione (segnalata da una sequenza di 5 lampeggi del LED verde ad indicare il corretto avvio) il micro inizializza i propri I/O impostando RA4 come input per l'acquisizione dei dati in arrivo dall'AC-RX2, RB4 ed RB5 come ingressi (provvisi di pull-up integrato) per leggere lo stato dei dip-switch (tali linee vengono attivate dotate del pull-up interno) ed RB6, RB7 per leggere lo stato dei pulsanti P1 (avvio apprendimento codici per attivare il canale 1) e P2 (avvio dell'apprendimento dei codici per il canale 2); anche queste ultime linee di I/O hanno attivo il pull-up interno.

Procedendo con l'inizializzazione, il microcontrollore imposta RB1, RB2, RB3 come uscite dedicate, rispettivamente, al comando del LED di segnalazione LD3 (verde, che segnala i passaggi della procedura di autoapprendimento e il normale funzionamento) LD1, che segnala l'attività del canale 1, ed LD2 fa lo stesso per il canale 2. Per i due relé il discorso è analogo a quello fatto per RL1 del circuito monocanale, dato che i

due stadi sono uguali a quello del ricevitore 1 canale: T1, che pilota il relé 2, viene mandato in saturazione quando RA6 si trova a livello logico alto, mentre T2 è attivato dallo stato logico alto sulla linea RB0. Non mancano i diodi di protezione D2 e D3.

Continuando con l'inizializzazione, il PIC setta GP2, GP3 e GP5 come ingressi, dedicati rispettivamente alla lettura del pulsante P1, del jumper J1 e dell'uscita dati dell'AC-RX2; per le prime due è attivo il pull-up interno.

I dip-switch del DS1 servono per impostare la modalità di funzionamento desiderata per ciascun canale; più esattamente, il Dip1 consente l'impostazione dell'OUT1 e il Dip2 imposta l'attività dell'uscita OUT2.

Dip chiuso significa funzionamento bistabile, mentre dip aperto significa funzionamento monostabile.

Come nel circuito monocanale, in modalità monostabile l'uscita non è temporizzata, pertanto si attiva premendo il pulsante sul trasmettitore e torna a riposo al rilascio del pulsante stesso. I trimmer R4 ed R5 non vengono montati, ma sono stati previsti per futuri sviluppi.

I due LED rossi seguono lo stato delle uscite durante il funzionamento. LD1 si accende se il relé dell'OUT1 viene eccitato e LD2 si accende se il relé dell'OUT2 viene eccitato.

PROCEDURA DI APPRENDIMENTO MODULO 2 CH

Per abbinare i trasmettitori al circuito bisogna procedere così: premere e tenere premuto il pulsante relativo al canale che si desidera memorizzare (P1 per il canale OUT1 e P2 per il canale OUT2). Il LED rosso relativo all'uscita (LD1 per OUT1 e LD2 per OUT2) si accende ad indicare che si è in autoapprendimento. Fatto ciò bisogna premere il pulsante del radiocomando che si desidera apprendere e attendere che il LED lampeggi, segno che l'apprendimento è andato a buon fine. Se il LED rimane fisso, vuol dire che la memoria è piena o il codice trasmesso non è valido. Notate che non è ammesso l'apprendimento di codici corrispondenti a tutti i dip-switch del trasmettitore ad ON, nel senso che per come viene gestita la EEPROM del microcontrollore non è ammesso il codice che corrisponde ad 1111111111. Ne deriva che, come per il ricevitore ad un canale, sono disponibili 4.095 combinazioni invece di 4.096. Quanto alla cancellazione dei codici, si può solo cancellare la memoria relativa ad un singolo canale di uscita, ma non un singolo codice di radiocomando. Per cancellare la memoria di un canale bisogna accendere il circuito premendo e mantenendo premuto il pulsante relativo al canale che si desidera cancellare: P1

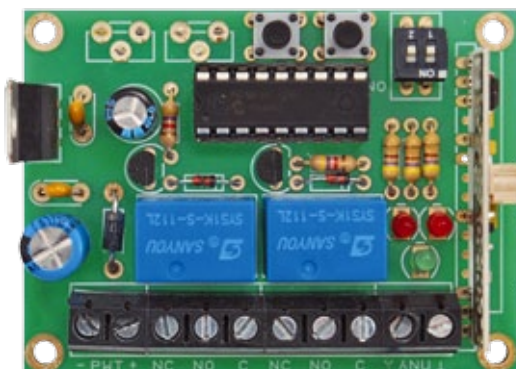
Tabella 2 - Significato dei LED nel ricevitore a 2 canali.

Diodo luminoso	Normale funzionamento	Programmazione
LD1	attività OUT1: acceso=RL1 eccitato spento=RL1 a riposo	-
LD2	attività OUT2: acceso=RL2 eccitato spento=RL2 a riposo	-
LD3	all'avvio o all'uscita dall'apprendimento o cancellazione dei codici, segnala con 5 lampeggi l'avvio in modalità normale.	si accende per 2 secondi quando si entra in programmazione; lampeggia quando il circuito ha appreso il codice trasmesso; è fisso se l'apprendimento è fallito.

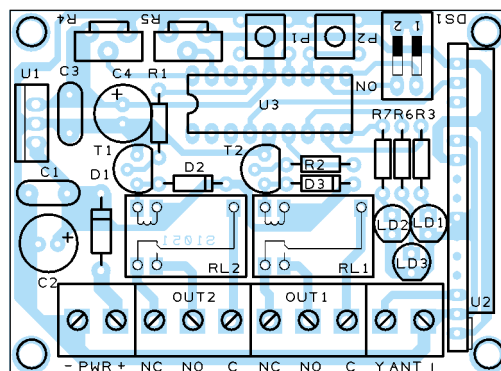
[piano di montaggio del **RICEVITORE 2 CANALI**]

Elenco Componenti:

R1: 4,7 kohm
 R2: 4,7 kohm
 R3: 470 ohm
 R4: non montato
 R5: non montato
 R6, R7: 470 ohm
 C1, C3: 100 nF multistrato
 C2, C4: 100 μ F 35 VL elettrolitico
 U1: 7805
 U2: AC-RX2
 U3: PIC16F88 (MF1051)
 D1: 1N4007
 D2, D3: 1N4148
 LD1: LED 3 mm rosso
 LD2: LED 3 mm rosso
 LD3: LED 3 mm verde



T1-T2: BC547
 P1: Microswitch
 P2: Microswitch
 RL1: Relè 12V
 RL2: Relè 12V
 DS1: Dip-switch 2 vie



Varie:

- Morsetto 2 poli (2 pz.)
- Morsetto 3 poli (2 pz.)
- Zoccolo 9+9
- Circuito stampato

per il canale 1 (OUT1) o P2 per il canale 2 (OUT2). Rilasciare il pulsante quando il LED verde (LD3) rimane acceso fisso per 2 secondi ad indicare l'avvenuta cancellazione del banco di memoria relativo al canale scelto. Il LED verde emetterà 5 lampeggi ad indicare l'uscita dalla procedura di cancellazione e l'avvio normale del ricevitore (**Tabella 2**).

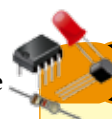
REALIZZAZIONE PRATICA

Per ciascuno dei circuiti abbiamo previsto uno stampato, del quale rendiamo disponibile (nel nostro sito web www.elettronica.in.it) la traccia lato rame: usatela per realizzare la pellicola occorrente alla fotoincisione, quindi procedete alla costruzione del c.s. Ottenuta e forata la basetta, montatevi i pochi componenti occorrenti, partendo da resistenze e diodi al silicio, quindi proseguendo con gli zoccoli, il dip switch (presente solo sulla versione 2 canali) i condensatori (prima i non polarizzati e poi gli elettrolitici) i transistor, i LED, i relé, il regolatore di tensione, le morsettiere ecc. Per il corretto orientamento degli elementi po-

larizzati (transistor, diodi, regolatore e microcontrollore) seguite il rispettivo piano di montaggio, che trovate in queste pagine. Completato il montaggio, non vi resta che inserire il microcontrollore, già programmato con il relativo firmware) nell'apposito zoccolo dello stampato, badando all'orientamento. Ricordate anche di dotare i ricevitori di un'adeguata antenna accordata sui 434 MHz: per esempio una ground-plane o un semplice stilo; il collegamento va realizzato preferibilmente con uno spezzone di cavo coassiale RG59, il cui conduttore centrale (anima) deve unire lo stilo con il morsetto Y della morsettieria ANT, mentre la calza di schermo deve andare a massa sulla predetta morsettieria dal lato del circuito e dall'altro all'eventuale piano di massa dell'antenna (se è un semplice stilo, isolate la massa dal lato dell'antenna, ossia collegatela solo allo stampato). L'antenna può essere anche un semplice spezzone di filo in rame in guaina lungo 17 cm (per avere un'antenna a $\frac{1}{4}$ d'onda) o 35 cm (per avere un'antenna a mezz'onda) collegato al contatto

Y della morsettieria ANT. Tutto questo discorso vale per entrambi i ricevitori (1 o 2 canali).

Per l'alimentazione dei circuiti, servono 12÷16 Vcc e una corrente continua di 70 mA per il ricevitore monocanale e 100 per il bicanale. ■



per il MATERIALE

Tutti i componenti utilizzati in questi progetti sono di facile reperibilità. I master dei circuiti stampati ed i firmware possono essere scaricati dal sito della rivista (www.elettronica.in.it). Entrambi i ricevitori sono anche disponibili in kit: cod. FT1050K (monocanale, 17,00 Euro); cod. FT1051K (bicanale, 19,00 Euro). Con questi ricevitori è possibile utilizzare trasmettitori a 2 canali (cod. 8110-TX3750-2CS Euro 19,00) e 4 canali (cod. 8110-TX3750-4CS Euro 22,00). Tutti i prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11,
 21013 Gallarate (VA)

Tel: 0331-799775 • Fax: 0331-792287

<http://www.futurashop.it>

Continuiamo il nostro viaggio alla scoperta di Flowcode, l'innovativo sistema di sviluppo grafico per microcontrollori proposto dalla Matrix Multimedia. In questa puntata analizzeremo una periferica wireless: ZigBee. Settima puntata.

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

Nelle puntate precedenti ci siamo occupati di diverse periferiche di comunicazione, che avevano una caratteristica comune, ossia quella di essere tutte periferiche di comunicazione cablate. Analizziamo adesso l'implementazione Flowcode di una periferica wireless, ossia interfacciamo il nostro microcontrollore con una periferica ZigBee.

Il Protocollo ZigBee

Il protocollo ZigBee nasce ufficialmente il 14 dicembre 2004, giorno in cui la ZigBee Alliance (consorzio di aziende che si occupano dello sviluppo e dell'aggiornamento dello standard e della promozione

della tecnologia su di esso basata) rilascia le specifiche 1.0. Come era già avvenuto in passato per la definizione di altri standard, il lavoro relativo allo standard ZigBee è stato diviso in due tronconi. Al task group IEEE 802.15.4 (parte del work group IEEE 802.15) è stato assegnato il compito di definire le specifiche relative al livello fisico e MAC dello standard, mentre la ZigBee Alliance è stata incaricata di arrivare alla definizione delle specifiche di alto livello (livelli Network e Application). La relazione che esiste tra 802.15.4 e ZigBee è la stessa che esiste tra 802.11 e WiFi e tra 802.15.1 e Bluetooth. Non bisogna quindi confondere le

Tabella 1 - Confronto tra le più diffuse tecnologie wireless.

	ZigBee™ 802.15.4	Bluetooth™ 802.15.1	Wi-Fi™ 802.11b	GPRS/GSM 1XRTT/CDMA
Application Focus	Monitoring & Control	Cable Replacement	Web, Video, Email	WAN, Voice, Data
System Resource	4KB-32KB	250KB+	1MB+	16MB+
Battery Life (days)	100-1000+	1-7	1-5	1-7
Nodes Per Network	255/65K+	7	30	1,000
Bandwidth (kbps)	20-250	720	11,000+	64-128
Range (meters)	1-75+	1-10+	1-100	1,000+
Key Attributes	Reliable, Low Power, Cost Effective	Cost, Convenience	Speed, Flexibility	Reach, Quality

specifiche dello standard IEEE 802.15.4 con lo standard ZigBee: con il primo ci si riferisce ad un insieme di specifiche di basso livello sulle quali si basa il secondo per la definizione di uno standard più complesso. Sulle specifiche indicate dallo standard ZigBee si basano i costruttori per la realizzazione del silicio. Chi a sua volta acquista i chip dai produttori, li ingloba in un sistema più complesso, il quale aggiunge un ulteriore strato a livello applicativo, caratteristico dell'applicazione finale vera e propria. Come è stato già accennato nell'introduzione il protocollo ZigBee è un protocollo wireless caratterizzato da bassi consumi energetici, semplicità nell'implementazione del codice (generalmente indicata come consumo approssimativo di memoria programma per la realizzazione di un nodo) ed elevato numero di nodi costituenti la rete; la velocità di comunicazione è relativamente bassa.

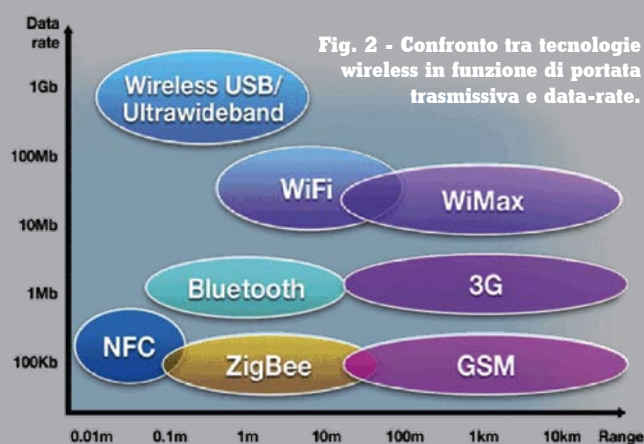
I dispositivi facenti parte di una PAN ZigBee si chiamano nodi; i nodi non sono tutti uguali ma lo standard ZigBee ne prevede tre tipi:

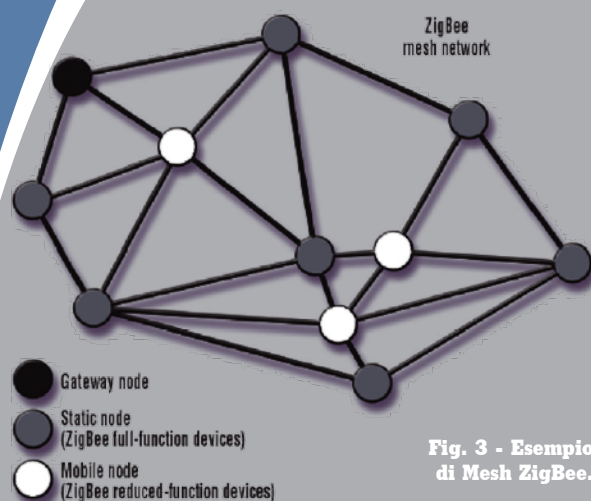
- *Coordinator*; è il nodo centrale della rete e ad esso sono demandate le funzioni più importanti, come ad esempio la creazione della rete stessa, l'assegnazione degli indirizzi, la scelta del canale RF ed altro ancora; ne può esistere al massimo uno per rete e può fungere da ponte tra reti diverse (è anche chiamato Gateway);
- *Router*; agisce come router intermedio, passando i dati da e verso altri dispositivi;
- *End Device*; include le funzionalità minime per dialogare con il loro nodo padre (Coordinator o Router), non possono trasmettere dati provenienti da altri dispositivi.

La **Tabella 1** mette a confronto lo standard ZigBee con alcune tra le più diffuse tecnologie wireless esistenti. Il data-rate tipico dichiarato sul canale radio è 250 kbps, mentre il data rate dell'interfaccia può assumere valori che variano da 1.200 a 115.200 bps; tale parametro non sempre si trova nei data-sheet dei dispositivi, in quanto lo ZigBee non è stato specificamente studiato per lo scambio di grosse moli di dati, come potrebbe invece essere per il WiFi.

La portata dipende dal dispositivo considerato, dall'antenna e dalla potenza irradiata in trasmissione: valori tipici sono compresi tra le decine e le centinaia di metri. In **Fig. 2** è riportato un grafico che mette a confronto diverse tecnologie wireless in funzione del range trasmissivo e del data rate (in Mbps). Come si vede Bluetooth e ZigBee occupano zone molto vicine nel grafico, ma mentre il primo

Fig. 1
Logo della
ZigBee
Alliance.





privilegia la velocità in trasmissione, il secondo la sacrifica in funzione di una maggiore portata trasmissiva. Oltre a ciò, ZigBee presenta una interessante caratteristica che lo differenzia da Bluetooth e da altri protocolli wireless, e gli permette di estendere il suo range utile a valori molto superiori rispetto alla portata di un singolo dispositivo. Infatti lo standard permette di mettere in comunicazione due nodi qualsiasi anche se non posti in collegamento radio diretto, è sufficiente che tra i due nodi in questione esista almeno un percorso costituito da nodi intermedi, che in questo caso fungono da routers. Questa interessante caratteristica viene generalmente indicata come *Mesh Networking*, una cui rappresentazione grafica è riportata in Fig. 3.

La gestione di reti Mesh è una prerogativa del protocollo, che viene implementata adottando i più recenti algoritmi di routing, oltre ad un efficiente sistema di indirizzamento dei dispositivi, che si avvale di due indirizzi, un indirizzo denominato MAC ed uno denominato Short. Il MAC è un codice identificativo a 64 bit (8 byte), univoco per ogni dispositivo prodotto al mondo. Tale codice è memorizzato all'interno di

un'apposita memoria ROM e non può essere modificato. Invece lo Short è un codice identificativo a 16 bit (2 byte) che permette di identificare univocamente un dispositivo all'interno di una rete ZigBee. Viene memorizzato in un'area di memoria protetta, ma riscrivibile, quindi può cambiare, in seguito, ad esempio, all'associazione del dispositivo con un'altra rete. Gli indirizzi Short sono assegnati dal Coordinator all'atto della creazione della rete.

Per quanto riguarda le bande radio utilizzate, lo standard sfrutta le frequenze assegnate nella banda ISM (Industrial, Scientific, Medical), che variano a seconda del paese considerato. In particolare può usufruire delle frequenze radio a 433MHz e 868 MHz in Europa e a 315 MHz e 915 MHz negli USA. Inoltre può usufruire della banda radio a 2,4 GHz praticamente in tutto il mondo. Su questa frequenza sono disponibili ben 16 canali e quindi, a meno di casi particolari, questa risulta essere la scelta più frequente dei costruttori. Le caratteristiche intrinseche di ZigBee rendono questo standard la scelta ideale per un gran numero di applicazioni. Si va dalla building automation al controllo industriale, dalle periferiche wireless per PC (come tastiere e mouse) ad applicazioni wireless di sicurezza. Un importante segmento è sicuramente costituito dalla

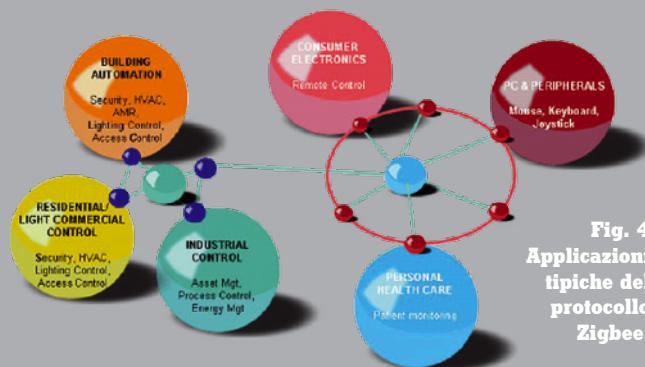


Tabella 2 - Applicazioni del protocollo ZigBee.

Settore di applicazione	Possibili applicazioni
Termoregolazione	HVAC, controllo temperatura, controllo umidità
Monitoraggio ambientale	Terra, qualità aria, qualità acqua.
Automazione industriale	Accelerometri, Vibrazioni, Pressione, Spostamenti
Monitoraggio strutturale	Monitoraggio gallerie, Monitoraggio ponti, Monitoraggio dighe, Monitoraggio edifici a rilevanza storica o artistica
Domotica	Sistemi di sicurezza e controllo



Fig. 5 - Modulo Xbee Serie 2.

realizzazione di reti che vengono generalmente indicate come WSN (Wireless Sensor Network) ossia reti di sensori wireless custom per applicazioni di monitoraggio di vario tipo. La Fig. 4 illustra alcune delle applicazioni tipiche del protocollo.

La **Tabella 2** elenca sinteticamente alcune delle più comuni applicazioni del protocollo. Tra i vari dispositivi ZigBee presenti al mondo hanno acquisito una certa fama (specialmente nel mondo hobbistico) i dispositivi Xbee, prodotti dalla Digi/Maxstream. Si tratta di una famiglia di radiomodem con interfaccia seriale, il cui esemplare è riportato in Fig. 5. Tali dispositivi sono stati scelti anche dalla Matrix come modulo hardware ZigBee per i dispositivi e-block.

L'hardware è molto semplice, e consiste in un piccolo PCB, all'interno del quale trova posto tutta l'elettronica necessaria al funzionamento, inclusa l'antenna che può essere a chip, a filo o con connettore RPSMA. La Fig. 6 illustra le diverse possibili scelte per l'antenna dei moduli Xbee. L'interfacciamento verso l'esterno è affidato a due file di 10 pin a passo 2 mm. L'interfaccia di comunicazione è una seriale standard UART, con baud rate selezionabile da 1200 bps a 115200 bps, sicché il collegamento minimo richiesto per il funzionamento dei moduli consiste nelle due linee della seriale, TX ed RX e nelle due linee di alimentazione. Sono previste

due modalità di funzionamento, modalità trasparente (denominata normalmente modalità AT) e modalità API. In modalità trasparente tutto ciò che viene inviato sulla seriale del modulo viene ritrasmesso sul canale radio e viene successivamente ritrasmesso in uscita sulla porta seriale del modulo ricevente. I radiomodem aggiungono alla sorgente e tolgono alla destinazione tutto l'header necessario alla comunicazione, ecco perché la modalità viene definita "trasparente". Non è possibile intervenire sull'header, che contiene informazioni quali indirizzo destinazione, lunghezza campo dati, checksum, etc. In modalità API, invece, la gestione dei pacchetti è decisamente più complicata.

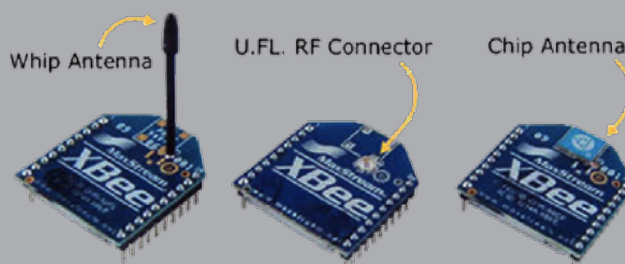


Fig. 6 - Possibili scelte per l'antenna dei moduli Xbee.

Nei data-sheet sono definiti una serie di frame utilizzabili, divisi per funzione, e caratterizzati da differente numero di byte, codice identificativo ed altro ancora.

Il componente ZigBee in Flowcode

Analizziamo adesso il componente ZigBee integrato in Flowcode. Il componente si trova sotto il gruppo dei componenti wireless, all'interno della barra dei componenti. Per inserirlo nel nostro progetto è sufficiente trascinarlo sul pannello,

Tabella 3 - Caratteristiche moduli Standard e PR.

Specification	Xbee	Xbee PRO
Indoor/Urban Range	30 m	100 m
Outdoor Range	100 m	1500 m
Transmit Power Output	1 mW (0 dBm)	100 mW (20 dBm)
RF data rate	250 kbps	250 kbps
Interface Data Rate	1200 - 115200 bps	1200 - 115200 bps
Receiver Sensitività	- 92 dBm	- 100 dBm
Supply Voltage	2.8 - 3.4 V	2.8 - 3.4 V
Transmit Current	45 mA	270 mA
Receive Current	50 mA	55 mA

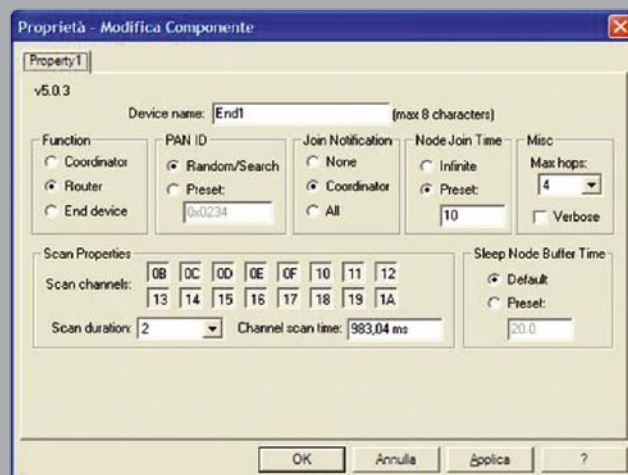
Fig. 7 – Proprietà extra.

come più volte abbiamo fatto in precedenza. Una volta posizionato il componente possiamo esaminarne le proprietà extra, cliccando con il tasto destro. La finestra delle proprietà extra del componente è riportata in **Fig. 7**. Come si può vedere è possibile scegliere che tipo di componente si vuole implementare (Coordinator, Router, End node), impostare la PAN id (random e impostata dall'utente), impostare il numero massimo di hops, i canali da includere all'interno della procedura di scan ecc.

Come per gli altri componenti, sono disponibili delle API che ne facilitano l'utilizzo, sinteticamente riportate in **Tabella 4**.

Progetto pratico: nodo wireless per controllo relé

Passiamo adesso al progetto pratico relativo a questa puntata. Questa volta ci proponiamo di realizzare una mesh ZigBee costituita da nodi in grado di controllare una batteria di 4 relé. Ogni nodo sarà identificato all'interno della mesh tramite un indirizzo costituito da due cifre numeriche codificate ASCII. Sarà possibile modificare l'indirizzo del nodo tramite il canale wireless e sarà inoltre possibile intervenire su alcuni parametri di funzionamento. I vari parametri saranno memorizzati all'interno della EEPROM del dispositivo, in modo che possano essere recuperati anche nel caso in cui il nodo venga privato della sorgente di alimentazione, in modo da non dover ripristinare



la configurazione in futuro.

Come già fatto per gli altri progetti pratici, esaminiamo le specifiche di progetto, elencate di seguito.

- Controllo di ogni singolo nodo tramite indirizzo in formato ASCII su due cifre.
- Attivazione/disattivazione di 4 relé.
- Gestione delle seguenti opzioni:
 - abilitazione/disabilitazione LED funzionamento;
 - abilitazione/disabilitazione Eco comando;
 - selezione periodo lampeggio LED.
- Memorizzazione delle opzioni di sistema su memoria EEPROM.

Il nostro progetto prevede quindi l'utilizzo di un protocollo livello applicazione che si appoggia sul protocollo ZigBee. La definizione del protocollo di livello applicazione è riportata di seguito.

- La stringa di comando inviata dal master è una stringa ASCII composta da 7 caratteri.

Tabella 4 - Macro del componente ZigBee Flowcode.

Nome Macro	Descrizione
Init_Network	Inizializza il modulo Xbee e prova ad eseguire il setup della network (coordinator) oppure ad effettuare il join ad una network esistente (router o end device).
Restart_Network	Re-inizializza la network Zigbee.
Scan_Network	Esegui uno scan per individuare una rete associabile.
Connect_To_Address	Connette il modulo ad uno specifico indirizzo all'interno della network.
Connect_To_Coordinator	Connettiti al coordinator.
Connect_To_All	Connettiti a tutti i nodi (trasmissione broadcast)
Node_Configure_Sleep	Imposta la modalità sleep del nodo (applicabile solo agli end device). Prende in ingresso due parametri, la modalità di sleep (1=Hibernate, 2=Doze, 4=Cyclic Sleep, 5=Cyclic Doze) e il timeout in ms.
Node_Wake	Risveglia i nodi figli dallo sleep mode.
Get_Signal_Level	Leggi il livello del segnale radio
Send_Char	Invia un byte sul canale radio.
Receive_Char	Restituisci il carattere ricevuto dal canale radio.

- La stringa di comandi è divisa in due sezioni, una sezione di identificazione ed una sezione di controllo, separate dal carattere ASCII “-”.
- La sezione di identificazione serve ad indirizzare il nodo che si vuole controllare. È composta da 3 caratteri ASCII: un primo carattere S (Slave) seguito da due caratteri numerici ASCII da “0” a “9”. Quindi per identificare il nodo 01, si invierà la stringa “S01”.
- La sezione di controllo è divisa in 3 sottofunzioni:
 - sottofunzione “R”, usata per il controllo diretto dei relé. In questa sottofunzione la lettera “R” è seguita da un carattere numerico ASCII da 1 a 4, usato per identificare il relé e dai caratteri “a” e “d” usati, rispetti-

vamente, per abilitare/disabilitare il relé (ad esempio per abilitare il relé 1 sul nodo 15 si userà la stringa S15-R1a);

- sottofunzione “I”, usata per impostare un nuovo indirizzo del nodo. In questa sottofunzione, la lettera “I” è seguita da due caratteri numerici da “0” a “9”, che costituiscono la nuova coppia di valori per l’indirizzo del nodo (Es. per cambiare l’indirizzo del nodo 0 in 35 si userà la stringa S00-I35);
- sottofunzione “C”, usata per impostare la configurazione del sistema. Le opzioni di configurazione attualmente disponibili permettono di abilitare/disabilitare il working LED e selezionarne il periodo di lampeggio ed abilitare/disabilitare l’eco.

La lista degli e-block necessari per la realizzazione di un nodo slave è riportata di seguito:

- 1x EB006 “USB PICmicro multiprogrammer board” (PIC18F4580 e quarzo a 12 MHz);
- 1x EB051 “ZigBee board” (configurazione Router/End Node);
- 1x EB030 “Relay board”;
- 1x EB004 “LED board”.

Per quanto riguarda il nodo Coordinatore, possiamo utilizzare una delle tante board di interfacciamento per nodi Xbee, come ad esempio la “Xbee explorer USB”, distribuita da Futura Elettronica. Quanto alle connessioni, dovremo collegare la EB051 sulla porta C (in modo da poter utilizzare la porta USART hardware per

Fig. 8
Connessione
E-Block per
progetto pratico.

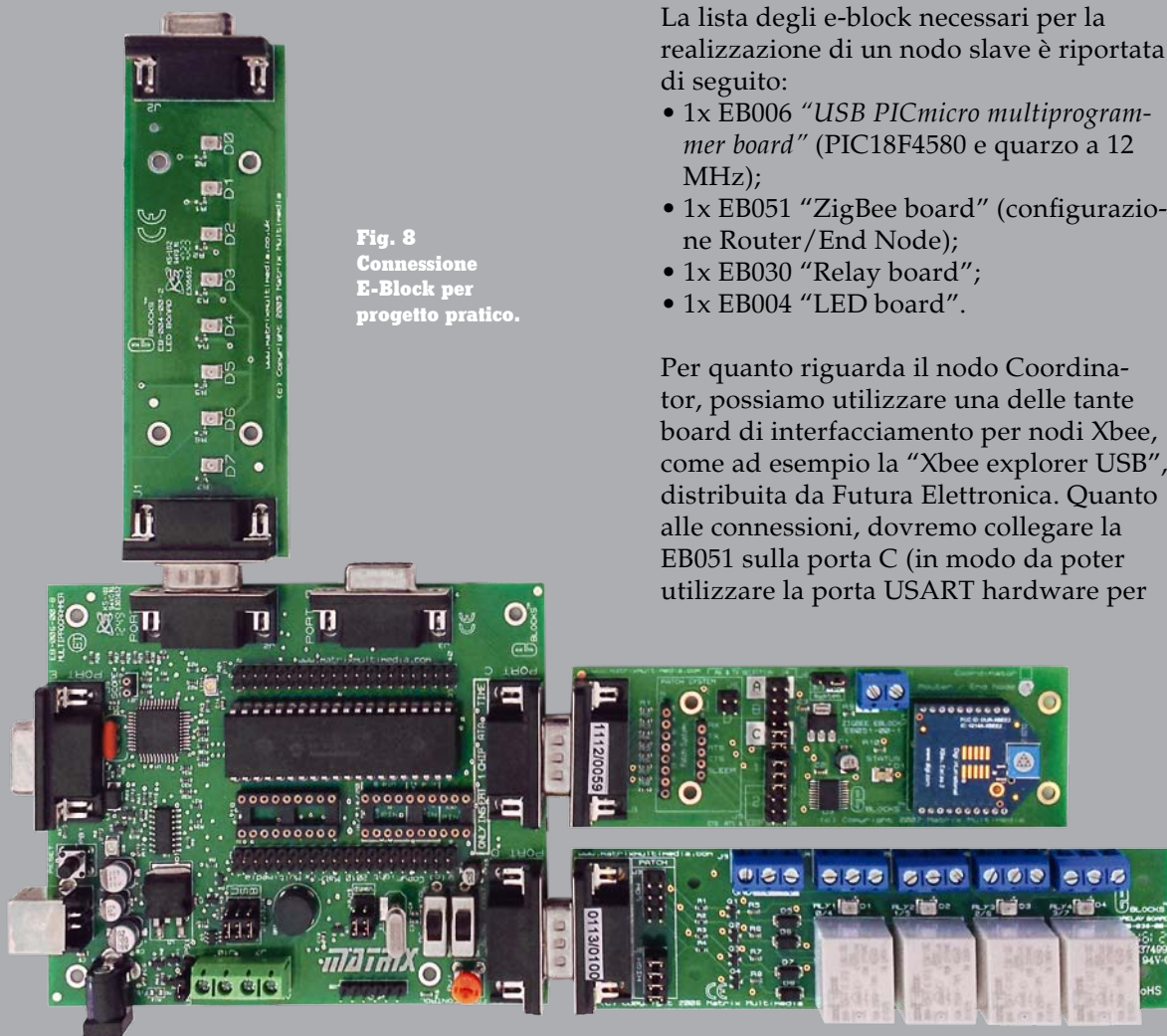


Tabella 5 - Periodicità e descrizione dei task del software del nodo ZigBee slave.

Task	Periodo [ms]	Descrizione
RxTask	1 (attivo), 1000 (riposo)	Riceve e decodifica i dati sul canale wireless.
RelayTask	500	Attua lo stato dei relé impostato dai comando wireless.
EepromTask	100	Aggiorna i parametri su memoria non volatile.
LedTask	Impostabile	Gestisce il working LED.

parlare con il modulo Xbee), e la EB030 sulla porta D. Infine colleghiamo la EB004 sulla porta A.

Non dimentichiamo che sia la EB051 che la EB030 necessitano anche dei collegamenti di alimentazione. In **Fig. 8** sono rappresentati i collegamenti hardware dei vari E-block.

Implementazione software

A questo punto possiamo iniziare ad esaminare i dettagli implementativi del software del nodo slave. Secondo le specifiche che ci siamo forniti, il nodo slave deve essere in grado di gestire i 4 relé, prevedere una gestione della memoria non volatile, gestire il "working" LED (LED che segnala che il nodo è connesso alla rete ed operativo) e, naturalmente, gestire il protocollo applicazione che abbiamo descritto in precedenza. L'architettura utilizzata per l'implementazione del nodo prevede l'utilizzo di uno scheduler ad 1 ms, ed un totale di 4 task, riportati per completezza in **Tabella 5**, insieme alle relative periodicità. L'implementazione dello scheduler di sistema è del tutto simile a quella vista nella puntata 5, con la differenza che questa volta è stato impostato un interrupt sul timer 2 con periodo di 1ms, in modo da avere la granularità richiesta. Si noti la differenza di tempistiche del task RxTask. Questo è il task più delicato dell'intero sistema, e quello che occupa maggiori risorse. Si è quindi deciso di diminuire l'intervallo tra le chiamate quando non ci sono dati in transito sul canale wireless ed aumentarla al massimo durante la fase di ricezione attiva. Analizzeremo in seguito il meccanismo software implementato per ottenere questo comportamento. L'implementazione del main scheduler è riportata in **Fig. 9**.

Come si può vedere, le chiamate ai vari task sono scandite dallo stato delle 4 variabili:

- *TSK_RelayTaskCounter*;

- *TSK_EepromTaskCounter*;
- *TSK_LedTaskCounter*;
- *TSK_RxTaskCounter*.

Non potendo analizzare tutti i task implementati in quanto una singola puntata non sarebbe sufficiente, ci soffermeremo

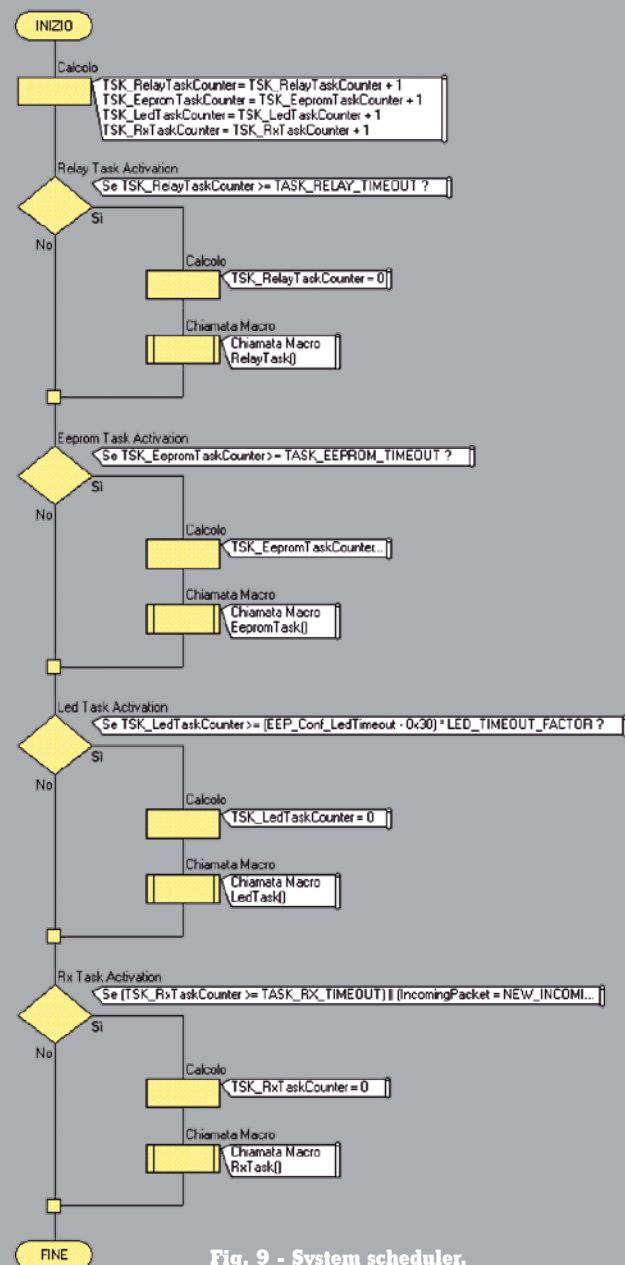


Fig. 9 - System scheduler.

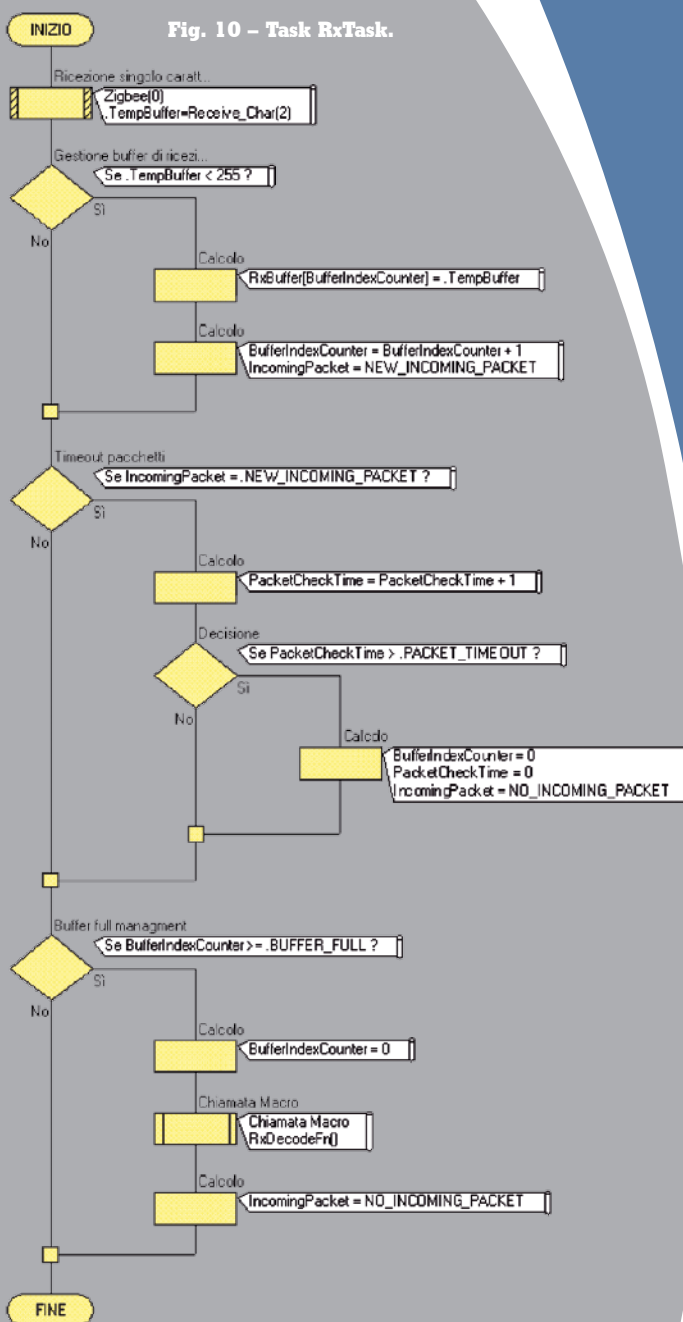
solo sui task più importanti. Prendiamo ad esempio il task RxTask, che esegue uno dei compiti più importanti del sistema, ossia la decodifica del pacchetto dati inviato dal Coordinator.

Poiché questo task risulta essere piuttosto pesante per la CPU durante l'esecuzione, anche senza pacchetti in arrivo, si è scelto di dargli una periodicità differente in caso di ricezione dati e non. In sostanza, in caso di assenza di attività RF il task esegue una volta al secondo, mentre se c'è attività sul canale radio, il tempo di esecuzione passa ad 1ms, in modo da decodificare immediatamente il pacchetto e passare nuovamente allo stato di attesa (con conseguente sovraccarico minimo della CPU). Questo tipo di comportamento è stato realizzato gestendo un flag globale denominato "IncomingPacket", che può avere due valori: NEW_INCOMING_PACKET e NO_INCOMING_PACKET. Normalmente il valore è NO_INCOMING_PACKET, ma in caso di inizio di una procedura di ricezione, il task RxTask lo forza al valore NEW_INCOMING_PACKET fino al termine della ricezione. Se andiamo a controllare la condizione di chiamata del task RxTask sullo scheduler, noteremo che essa vale: $(TSK_RxTaskCounter \geq TASK_RX_TIMEOUT) \vee (IncomingPacket = NEW_INCOMING_PACKET)$

Ossia, se il valore di IncomingPacket è diverso da NEW_INCOMING_PACKET (situazione di attesa), il task viene eseguito con la sua periodicità classica (TASK_RX_TIMEOUT), altrimenti si passa ad una esecuzione veloce ad ogni chiamata dello scheduler stesso (ossia una volta al ms).

Lo schema a blocchi del task RxTask è riportato in Fig. 10. Come si può vedere dallo schema a blocchi, se c'è un byte in ricezione sul canale, lo si acquisisce facendo uso della macro Receive_Char(), messa a disposizione dal componente ZigBee. Dopo aver ricevuto il byte su un buffer di servizio, lo si sposta sul buffer di ricezione (e si aumenta l'indice associato al buffer) e si setta il flag IncomingPacket al valore NEW_INCOMING_PACKET.

Fig. 10 - Task RxTask.



PACKET. Contestualmente viene attivato un timer (PacketCheckTime) che rimane attivo per tutta la fase di trasmissione. Scopo del timer è evitare che pacchetti tronchi blocchino il task in attesa di una parte del pacchetto. Dopo 200ms si resetta tutto e si ricomincia daccapo. Infine, quando il pacchetto è ricevuto completamente si resettano contatori e flag e viene chiamata la funzione di decodifica (RxDecodeFn).

La RxDecodeFn è una funzione (non un task, quindi la sua esecuzione non è

periodica ma event triggered) che ha il solo scopo di decodificare il pacchetto in ricezione e impostare correttamente il valore delle variabili globali affinché gli altri task compiano correttamente i loro compiti. Lo schema a blocchi della RxDecodeFn è riportato in Fig. 11. Descriviamo brevemente l'implementazione. In pratica la prima operazione eseguita è l'assegnamento dei vari byte del pacchetto a delle specifiche variabili, in maniera da agevolare la fase succes-

Function_Parameter2 = RxBuffer[.DEC_PAR_2]

Il primo controllo effettuato è sulla prima parte del pacchetto, quella contenente l'indirizzo del nodo. Il controllo effettuato è il seguente:

*(NodeType = NODE_TYPE_SLAVE) &&
(Packet_IDMajor = EEP_Id_NodeMajor)
&& (Packet_IDMinor = EEP_Id_NodeMinor)*

Quindi si controlla che il nodo sia uno slave ("S") e che l'ID corrisponda con i valori caricati dalla memoria EEPROM.

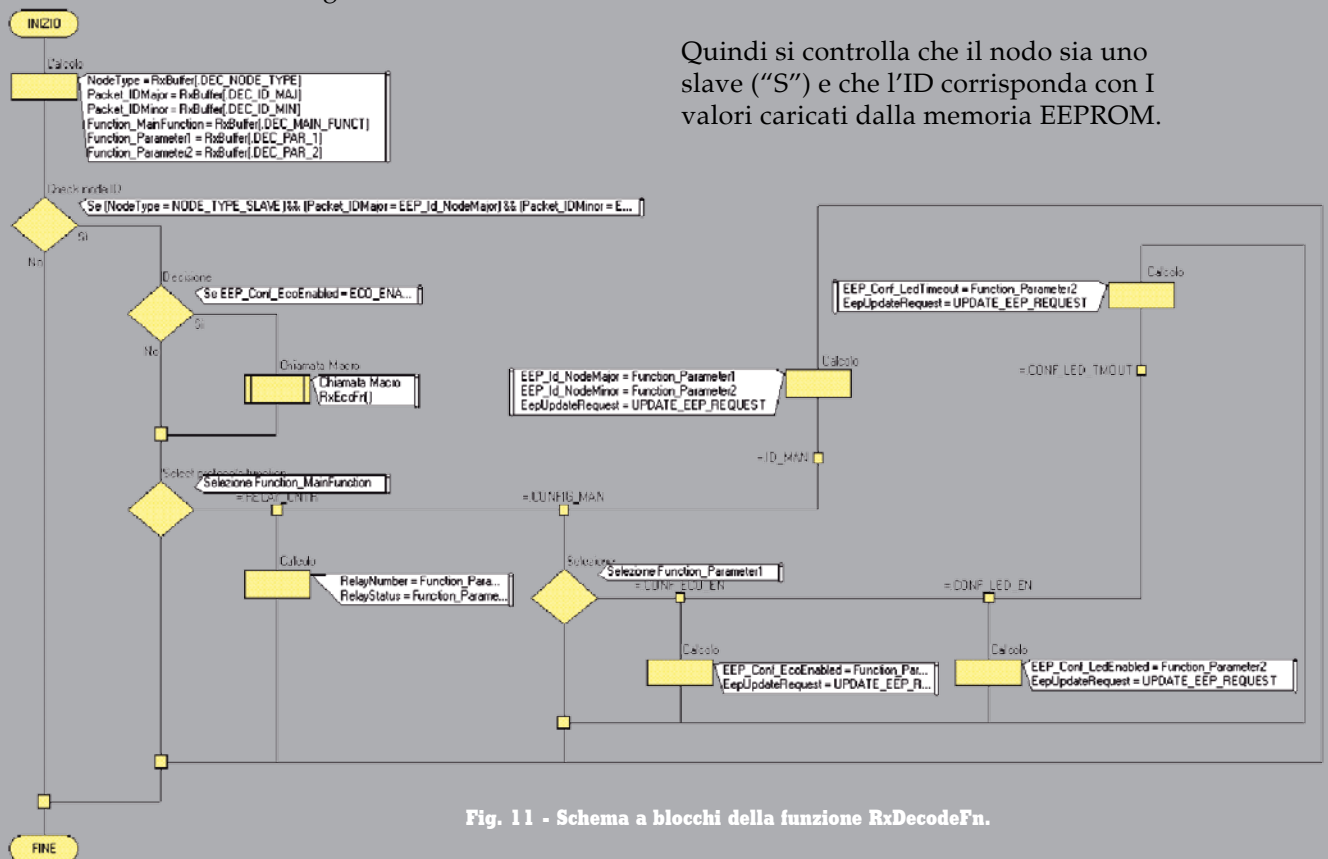


Fig. 11 - Schema a blocchi della funzione RxDecodeFn.

siva. Nello specifico viene utilizzato un blocco calcolo che esegue le seguenti assegnazioni:

*NodeType = RxBuffer[.DEC_NODE_TYPE]
Packet_IDMajor = RxBuffer[.DEC_ID_MAJ]
Packet_IDMinor = RxBuffer[.DEC_ID_MIN]
Function_MainFunction = RxBuffer[.DEC_MAIN_FUNCT]
Function_Parameter1 = RxBuffer[.DEC_PAR_1]
Function_Parameter2 = RxBuffer[.DEC_PAR_2]*

Se il controllo ha esito positivo si prosegue, altrimenti il pacchetto viene scartato. Dopo aver effettuato questo check preliminare si passa alla decodifica dell'istruzione vera e propria del pacchetto. Prima di farlo però, si controlla se l'opzione Eco è attiva (*EEP_Conf_EcoEnabled = ECO_ENABLED?*) e, in caso positivo, si trasmette l'eco del pacchetto ricevuto. Successivamente, un blocco selezione multipla decodifica il campo dati effettivo del pacchetto, impostando adeguatamente le variabili globali utilizzate dagli

Fig. 12 - Schema a blocchi del task RelayTask.

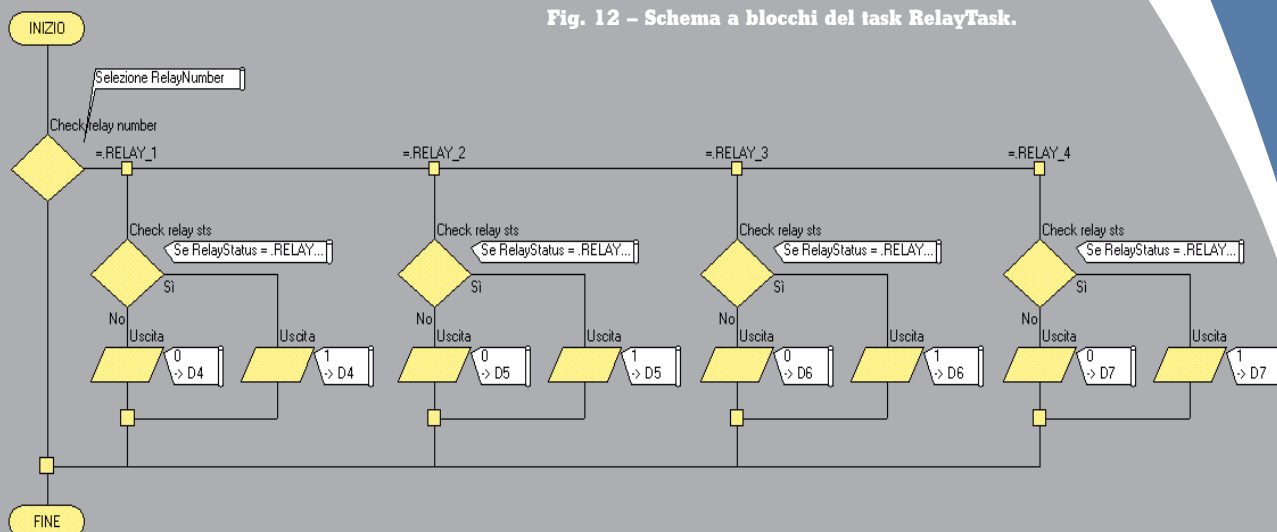
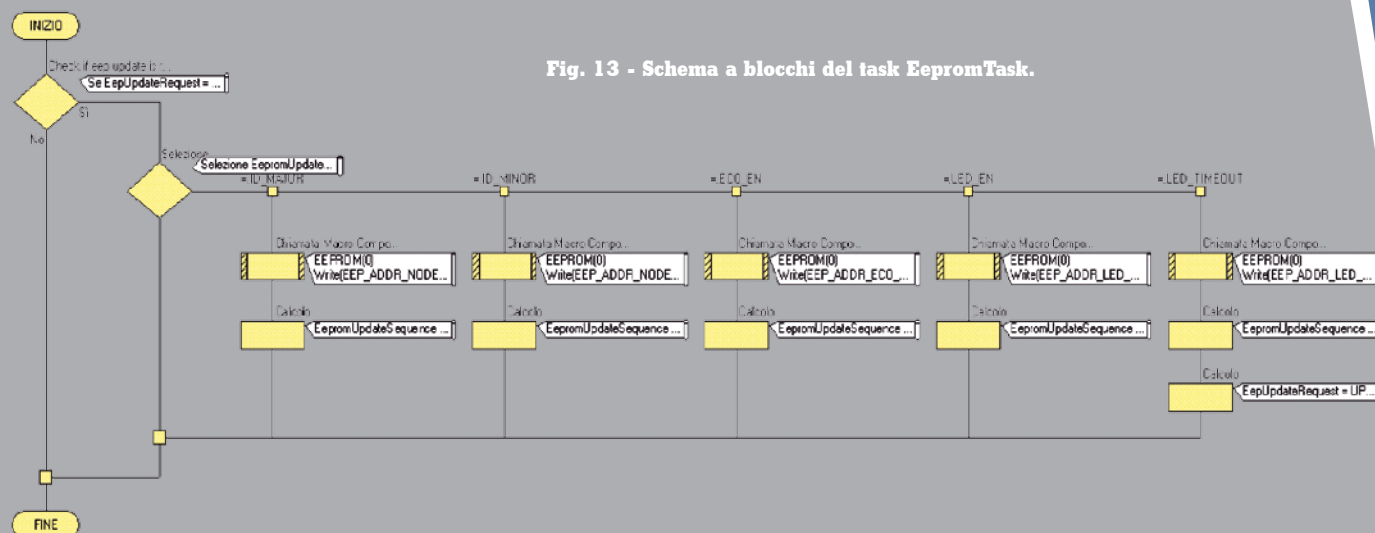


Fig. 13 - Schema a blocchi del task EepromTask.



altri task per l'esecuzione delle azioni specifiche.

In particolare, la sezione relativa al controllo dei relé, una volta verificato che il byte Function_MainFunction corrisponda al valore ASCII "R", effettua le seguenti assegnazioni:

RelayNumber = *Function_Parameter1*
RelayStatus = *Function_Parameter2*

Queste informazioni vengono poi utilizzate dal task RelayTask per il controllo effettivo dei relé. Una parola a parte merita la gestione della memorizzazione dei parametri in EEPROM (questo riguarda, ad esempio, tutte le opzioni di configurazione). Tale gestione è demandata al relativo task, ma, anche in questo caso, per sapere se i parametri in memoria sono

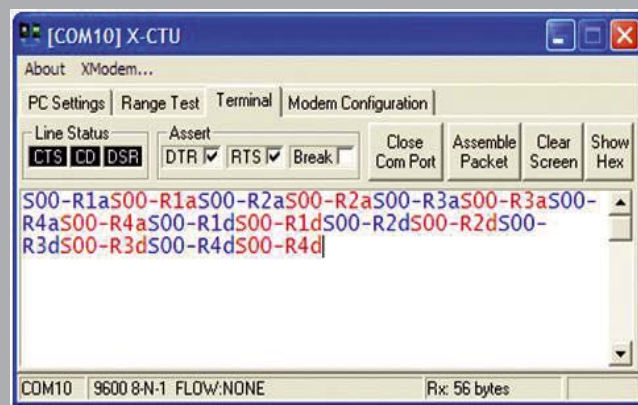
da aggiornare, si utilizza una variabile globale. Così, ad esempio, per aggiornare il parametro di configurazione relativo all'abilitazione dell'eco, il relativo blocco calcolo imposta:

Eep_Conf_EcoEnabled = *Function_Parameter2*
EepUpdateRequest = *UPDATE_EEP_REQUEST*

Eep_Conf_EcoEnabled contiene il valore del parametro, mentre il flag *EepUpdateRequest* segnala al task EepromTask che deve essere eseguito un aggiornamento dei parametri in memoria. Passiamo ora ad esaminare il task di controllo dei relé, che risulterà di semplice comprensione, ora che è stata esaminata la logica di comunicazione tra i task. Lo

Fig. 14 - Esempio di comunicazione.

schema a blocchi del task RelayTask è riportato in Fig. 12. Questo task gira a 500ms e non fa altro che monitorare lo stato delle due variabili globali RelayNumber e RelayStatus. La prima dice qual è il relé da controllare, mentre la seconda ne indica lo stato. Come abbiamo potuto esaminare in precedenza, queste variabili sono impostate dal task RxTask, ed in particolare dalla RxDecodeFn. Una volta determinato il relé da controllare ed il suo stato si agisce opportunamente sulla porta D tramite dei blocchi di uscita. Vediamo adesso come viene gestita la memoria non volatile, elemento fondamentale nel funzionamento del sistema, dato che contiene l'indirizzo del nodo ed anche i parametri di configurazione. Ci sono diversi modi di gestire una memoria EEPROM in un progetto embedded, e la scelta della modalità non è mai una questione banale. In questo progetto si è deciso di utilizzare una gestione di tipo mirror. Ossia, i dati immagazzinati nella memoria EEPROM sono copiati in opportune locazioni di memoria RAM all'avvio del sistema (mirror della EEPROM). Se durante la vita operativa del sistema, uno di questi parametri cambia, viene avviata una procedura che aggiorna la EEPROM (il mirror viene ricopiato sulla EEPROM). In questo modo, al successivo riavvio verranno caricati i dati aggiornati. Quindi il blocco EEPROM all'interno di questo progetto è costituito da due parti distinte. Una funzione di inizializzazione, che copia il contenuto della EEPROM all'interno delle variabili "mirror", ed un task run time che si occupa dell'aggiornamento, quando necessario. La funzione di inizializzazione non è particolarmente complessa, quindi la tralasciamo. Esaminiamo invece il task, il cui schema a blocchi è riportato in Fig. 13. Come abbiamo spiegato in precedenza, il compito del task è quello di aggiornare i dati in EEPROM copiandovi il contenuto delle rispettive variabili "mirror". Per evitare di gravare troppo sulla CPU i dati non sono copiati tutti insieme, ma è stata costruita una sequenza che copia un dato per ciclo di schedu-



ling, se la procedura è attiva. In questo modo riduciamo al minimo indispensabile il carico di lavoro in fase di aggiornamento dei dati in EEPROM, al prezzo di un tempo complessivo maggiore per eseguire l'intera operazione. L'implementazione è molto semplice: all'inizio il task controlla se c'è stata una richiesta di aggiornamento della memoria non volatile (EepUpdateRequest = UPDATE_EEP_REQUEST?). In caso affermativo viene avviata la procedura e i dati sono salvati in sequenza, uno per ciclo. La sequenza è stata realizzata facendo uso di un blocco selezione multipla e di una variabile globale (EepromUpdateSequence). Al termine della sequenza viene resettato anche il flag EepUpdateRequest (EepUpdateRequest = UPDATE_EEP_COMPLETE). Con quest'ultima analisi i task più significativi sono stati analizzati. Possiamo adesso testare il nostro sistema, usando un qualsiasi programma terminale per comunicare con il coordinator. Per eseguire i nostri test abbiamo utilizzato X-CTU, il software fornito dalla Digi per la configurazione dei radiomodem Xbee. In Fig. 14 è possibile vedere uno screenshot di comunicazione nel quale era stata precedentemente attivata la funzione eco (vengono attivati e poi disattivati in sequenza i quattro relé). In questa settima puntata abbiamo analizzato in dettaglio il componente ZigBee, oltre a presentare un progetto pratico di una certa complessità, che ci ha permesso di fissare alcuni importanti concetti visti nelle puntate precedenti. Nella prossima e ultima puntata analizzeremo (e proviamo con un progetto pratico) i componenti TCP/IP e il Webserver. ■