

MusicXmas: ritmo e colore per il tuo Natale

- Arduino+Intel per la nuova board Galileo
- Controller video con Arduino Yún
- 4G & IPv6 per soluzioni più smart
- myKIT, misure con fotoresistenza



Web Server Botanico

- Dotklok: ecco chi ha vinto la 3Drag
- Aurel WiFi: wireless per tutti
- Rome Maker Faire, un successo annunciato
- Stampa e vinci con la 3Drag
- Corso mbed

MODULO AUREL XTR-WiFi WIRELESS PER TUTTI

di MARCO MAGAGNIN

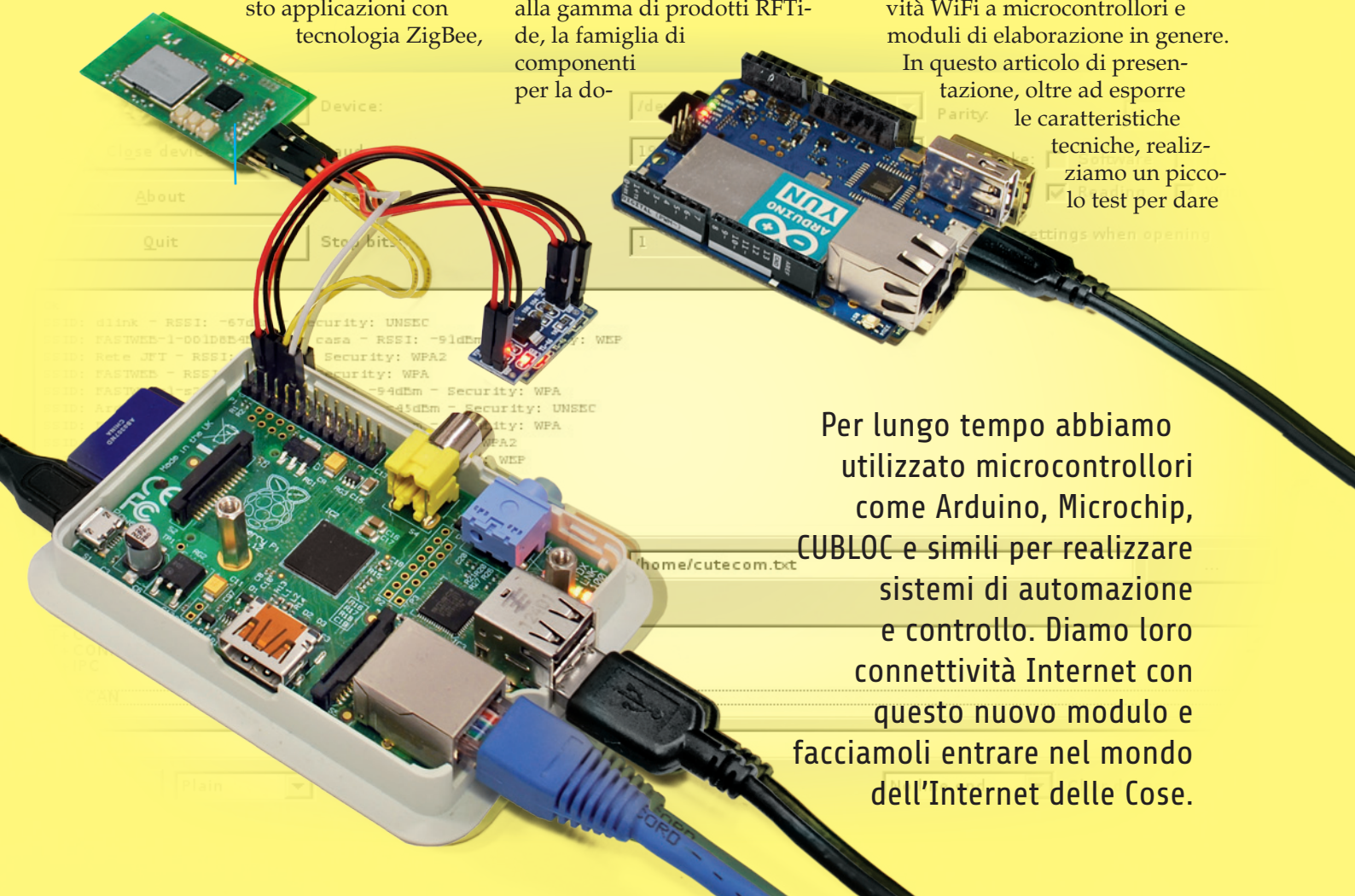
Aurel, azienda italiana leader nella produzione di moduli miniaturizzati ibridi per comunicazione senza fili su diverse frequenze e con differenti potenze, dopo aver proposto applicazioni con tecnologia ZigBee,

è ora entrata anche nel mercato dei moduli WiFi rilasciando un nuovo prodotto di comunicazione che risponde agli standard WiFi IEEE802.11 b/g. Questo modulo, nato come estensione alla gamma di prodotti RFTi-de, la famiglia di componenti per la do-

motica che abbiamo presentato in anteprima nel numero 173, può essere utilizzato anche in una miriade di altre applicazioni, hobbistiche e professionali, dove sia necessario fornire connettività WiFi a microcontrollori e moduli di elaborazione in genere.

In questo articolo di presentazione, oltre ad esporre le caratteristiche tecniche, realiz-
ziamo un piccolo test per dare

Per lungo tempo abbiamo utilizzato microcontrollori come Arduino, Microchip, CUBLOC e simili per realizzare sistemi di automazione e controllo. Diamo loro connettività Internet con questo nuovo modulo e facciamoli entrare nel mondo dell'Internet delle Cose.





Pin-Out

1 GND	7 GPID1
2 Button 2	8 GPID 0
3 SPI-MDSI/UART TX	9 SPI-MDSO/UART RX
4 SPI CLK	10 SPI-CS
5 GND	11 RST
6 Button 1	12 VCC

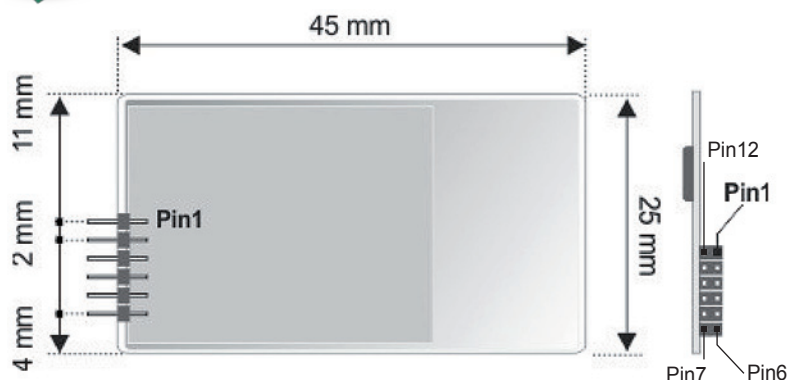


Fig. 1 - Caratteristiche fisiche del modulo XTR-WiFi.

connettività WiFi ad una scheda Raspberry Pi. XTR-WiFi è un modulo di interfaccia a 2,4 GHz progettato per collegare un generico dispositivo dotato di interfaccia seriale, o SPI, ad una rete TCP/IP in WiFi. Il modulo deve essere associato come client ad un access point WiFi. Il firmware del modulo realizza tutta l'infrastruttura necessaria a configurare e supportare la connessione ed il traffico di rete. Con semplici comandi AT si può configurare il modulo per associarlo ad un determinato

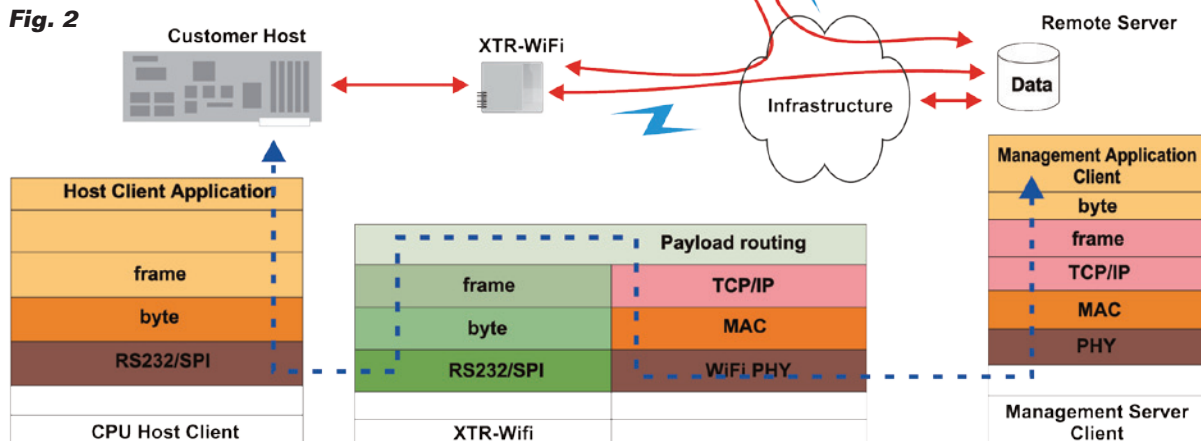
network WiFi, soddisfacendo le protezioni di sicurezza WEP, WPA e WPA2. Il client WiFi può essere configurato con IP statico o dinamico, da acquisire via DHCP. Sono supportati i protocolli TCP, ICMP, DHCP e mDNS. La comunicazione può essere impostata sia in modalità server che client. La potenza di trasmissione è di +17 dBm e la sensibilità in ricezione di -92 dBm. La comunicazione tra modulo e dispositivo collegato può

Il modulo XTR-WiFi si configura utilizzando comandi AT dedicati da inviare sulla porta seriale. Al momento dell'accensione il modulo invia un messaggio di benvenuto. Dopo ciascun comando si ottengono messaggi di risposta all'azione richiesta o segnalazioni di errori incontrati. Per una più dettagliata descrizione dell'utilizzo dei comandi e dei messaggi di conferma ed errore vi rimandiamo al manuale "Aurel XTR-WiFi User Guide". Utilizzando il metodo descritto nell'articolo è comunque possibile sperimentare approfonditamente il funzionamento del modulo, analizzando in concreto il comportamento di ciascun comando. Nella **Tabella 1** sono elencati i comandi con una breve descrizione della sintassi e del funzionamento. Nella **Tabella 2** sono riportati i significati dei codici d'errore che possono verificarsi nella configurazione del modulo.

Tabella 2

Codice errore	Descrizione
1	Errore di sintassi
2	Errore di connessione verso l'Access Point
3	Errore nell'apertura del socket TCP
4	Errore nel binding del socket TCP
5	Errore in fase di ascolto di connessioni in ingresso
6	Errore nell'invio dei dati
7	Errore nella chiusura del socket
8	Numero di argomenti errato
9	Errore nella connessione con il server remoto
10	Errore protocollo mDNS
11	Errore nell'interrogazione del server DNS, durante la risoluzione di un nome simbolico
12	Errore nell'esecuzione di una richiesta ping

Fig. 2



Comandi AT di configurazione XTR-WiFi

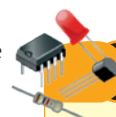
Tabella 1

Comando	Descrizione
All'accensione	Al momento dell'accensione o dopo un reset il modulo risponde con il messaggio di benvenuto che contiene la versione del firmware. ad es.: <i>Aurel WiFi: Version 0.1-1.10</i>
AT	Comando generico per testare il corretto funzionamento del modulo – risposta OK
AT+SCAN	Esegue la scansione degli Access Point disponibili. Viene presentato l'elenco degli AP disponibili con il livello del segnale ed il tipo di protezione.
AT+DHCP	Permette di configurare il modulo in modo che riceva un IP dinamico dal server DNS dell'access point al quale è stato associato.
AT+CONF=<IPAddress>;<NetMask>;<Gateway>;<DNS Server>	Permette di configurare il modulo con un indirizzo IP statico, compatibile con lo schema di indirizzi associati alla rete alla quale si vuole associare il modulo (idem per gli altri parametri, in genere desumibili dalla configurazione del router) IPAddress – Indirizzo IP statico (es. 192.168.0.33) NetMask – maschera di rete (es. 255.255.255.0) Gateway – (es. 192.168.0.1) DNSServer – (es. 8.8.8.8)
AT+CONN=[SecurityLevel];[Ssid];<SecurityKey>	Permette di configurare il modulo per associare un access point disponibile, specificando: SecurityLevel (UNSEC, WEP, WPA o WPA2) Ssid dell'access point La chiave di sicurezza corrispondente al livello di sicurezza richiesto dall'access point.
AT+SMRT	Permette di iniziare la configurazione in modalità smart config. Dopo aver posto il modulo in modalità smart config, la configurazione si perfeziona premendo l'apposito pulsante sul router al quale si vuole collegare il modulo. È in alternativa al comando AT+CONN.
AT+IPC	Permette di interrogare il modulo per leggerne i parametri di configurazione. Corrisponde al comando "ifconfig" in GNU/Linux e "ipconfig" in Windows.
AT+PING=[IP Address]	Classico comando di interrogazione di un host remoto per verificarne la raggiungibilità. È possibile specificare, oltre ad un indirizzo IP, anche un nome host, per verificare le corrette impostazioni dei server DNS.
AT+MDNS=MyDevice	Con questo comando è possibile assegnare un nome host al modulo e propagarlo sulla rete. . In seguito viene instaurata una procedura di comunicazione broadcast del nome associato al modulo secondo le modalità del protocollo mDNS.
AT+SRVR=[Port Number]	Imposta il modulo per funzionare in modalità server. È necessario impostare un numero porta (es. 4444) sulla quale il modulo si pone in ascolto di richieste in entrata. La configurazione server in ascolto sulla porta 4444 è la configurazione predefinita dopo che il modulo si associa ad un access point. È possibile utilizzare questo comando per cambiare il numero di porta sulla quale il modulo si mette in ascolto. Per configurare il modulo in modalità client utilizzare il comando AT+CLNT.
AT+STOP	Comando per interrompere, in modalità server, l'ascolto di messaggi in arrivo sulla porta impostata.
AT+CLNT=[IP Address];[Port Number]	Imposta il modulo per funzionare in modalità client, permettendo il collegamento ad un host remoto (per. Es. un altro modulo XTR-WiFi) configurato come server : IPAddress – Indirizzo IP del server al quale si desidera collegarsi PortNumber – Numero porta del server in ascolto per i messaggi in arrivo.
AT+DSCN	Disconnette il modulo dall'access point al quale era associato
Sequenza di escape "+++"	Permette di riportare il modulo in modalità comandi, interrompendo qualsiasi connessione ed attività di comunicazione in corso.

essere configurata come seriale o SPI. La tensione di alimentazione deve stare nell'intervallo tra 2,7 e 3,6 VDC, così come i livelli dei protocolli di comunicazione. In **Fig. 1** è visibile la pin-out del modulo XTR-WiFi. Il connettore dei pin ha passo 2 mm. Il modulo viene fornito in quattro versioni; con e senza presa per l'antenna esterna e per ciascuna versione con il connettore lineare o orientato a 90 gradi. Nella configurazione senza connettore

per l'antenna esterna, l'antenna è ovviamente ricavata sul circuito stampato.

Queste caratteristiche rendono il modulo adatto a dotare di connettività WiFi e quindi visibilità in rete dispositivi di controllo tradizionali, come moduli Arduino, PIC, Teensy e, perchè no, Raspberry Pi e schede Linux-based dotate di porta seriale, per renderle compatibili con la filosofia dell'Internet delle Cose. Ovviamente è possibile configu-

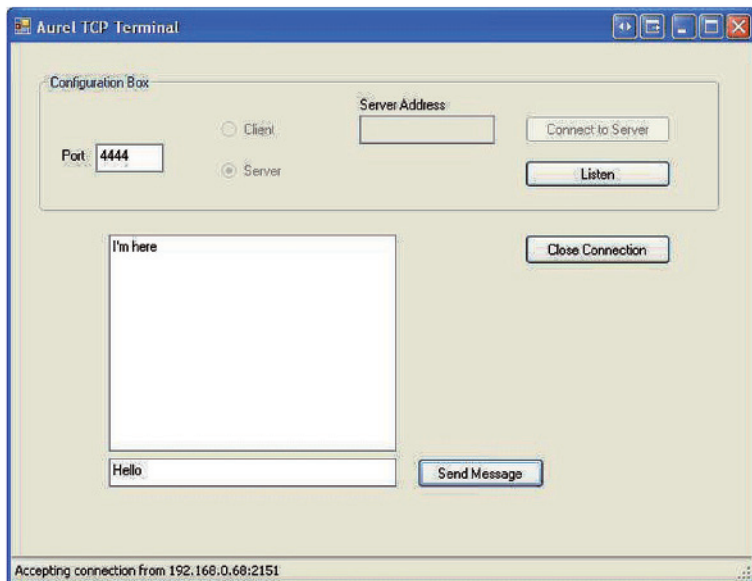


per il MATERIALE

Il modulo Aurel WiFi è disponibile presso Futura Elettronica al prezzo di 48,00 Euro IVA compresa (cod. 8110-XTR-WIFI). La board Arduino Yún costa invece 62,90 Euro.

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

Fig. 3



rare due o più moduli XTR-WiFi in modo da farli lavorare tra loro sfruttando la comunicando via rete WiFi.

SCHEMA DI FUNZIONAMENTO

In Fig. 2 vediamo lo schema di funzionamento del modulo XTR-WiFi. Lo schema a blocchi nella parte centrale della figura mostra tutto il lavoro che il modulo svolge per semplificare la vita all'utilizzatore.

Sono visibili due modalità di funzionamento:

- Modulo XTR-WiFi configurato come client. A seguito di una richiesta sulla porta seriale, il modulo XTR-WiFi inizializza la connessione TCP/IP sulla base delle configurazioni impostate con i comandi AT. In pratica viene aperto un socket verso il server host sulla base dell'indirizzo IP e della porta. A questo punto il canale di comunicazione seriale può essere utilizzato in entrambe le direzioni.
- Modulo XTR-WiFi configurato come server. Questa configurazione consente di potersi collegare al modulo con dispositivi client come PC o smartphone. In questo caso il front end TCP/IP del modulo viene confi-

gurato come server e si pone immediatamente in ascolto sulla porta specificata in attesa di richieste dai client.

In entrambe le configurazioni il segnale seriale viene normalizzato in formato stringa che viene incapsulato in un pacchetto TCP/IP e poi inviato sulla rete WiFi attraverso la connessione stabilita inizialmente; nel caso dei segnali in ingresso viene eseguito il processo inverso, dal pacchetto TCP/IP viene estratto il contenuto del messaggio e ritrasformato in codifica seriale. Tutto quanto detto, normalizza-

zione a parte, vale anche per le comunicazioni in protocollo SPI.

CONFIGURAZIONE INIZIALE E UTILIZZO

Le operazioni per rendere operativo il modulo XTR-WiFi si dividono in due semplici fasi:

- Configurazione iniziale – In questa fase devono essere impostati, tramite gli appositi comandi AT, i parametri tipici di una connessione WiFi verso un access point come SSID e chiave di sicurezza WEP/WPA. È anche necessario configurare i parametri di rete come l'indirizzo IP (statico o dinamico), la netmask ed il default gateway. Infine la modalità di funzionamento, come client o server.
- Utilizzo come canale di comunicazione: a questo punto il modulo può essere utilizzato come canale di comunicazione seriale incapsulato nella connessione TCP/IP.

Per eseguire la configurazione seriale è possibile utilizzare un terminale TCP per Windows scaricabile dal sito www.aurelwireless.com (Fig. 3) anche se noi abbiamo adottato una soluzione più "domestica". Per fare un semplice test abbiamo

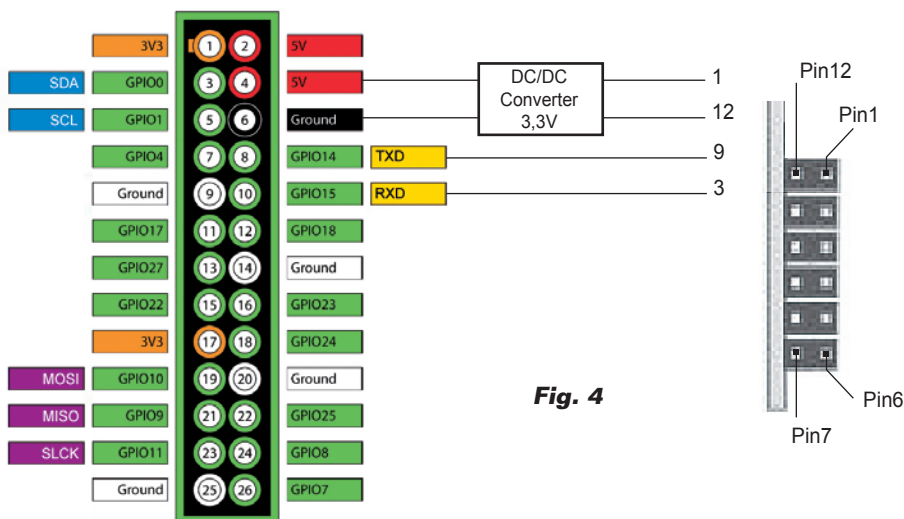


Fig. 4

collegato il modulo XTR-WiFi al connettore GPIO di Raspberry Pi, che abbiamo utilizzato per la configurazione e per un semplice test di funzionamento. Per fare questo abbiamo realizzato il piccolo circuito di Fig. 4. Siccome il pin 3,3 V di Raspberry Pi non è in grado di erogare la corrente necessaria per l'alimentazione del modulo XTR-WiFi abbiamo prelevato la tensione di alimentazione dal pin a 5 V del connettore GPIO portandola a 3,3V con il convertitore DC/DC reperibile presso Futura Elettronica con il codice 2846-CONVDCDC3V.

L'intenzione è di far comunicare il nostro PC portatile con Raspberry Pi mediante una connessione seriale incapsulata nel protocollo WiFi. Per questo dobbiamo instaurare una connessione tra modulo e PC utilizzando un access point. Come access point abbiamo utilizzato la scheda Arduino Yún che abbiamo presentato in questo stesso numero. A tal fine è sufficiente collegare il solo cavo di alimentazione: la configurazione finale è visibile nella foto di Fig. 5.

Come già descritto nell'articolo citato, Arduino Yún nasce già come access point WiFi.

Per le prove, abbiamo connesso il nostro PC tramite WiFi all'access point di Arduino Yún. Abbiamo collegato il nostro PC in rete locale con Raspberry Pi e aperto il terminale cutecom su Raspberry Pi utilizzando Xming come server X. Per approfondire l'utilizzo del terminale seriale e di Xming rimandiamo ai numeri precedenti della rivista ed al libro "Raspberry Pi - Il mio primo Linux embedded". Con cutecom abbiamo programmato il modulo XTR-WiFi collegato alla seriale /dev/ttyAMA0 di Raspberry Pi. Impostiamo la velocità a 19200, 8 bit di protocollo, N come parità e 1 stop bit. All'accensione il modulo ci risponde con il messaggio di benvenuto:

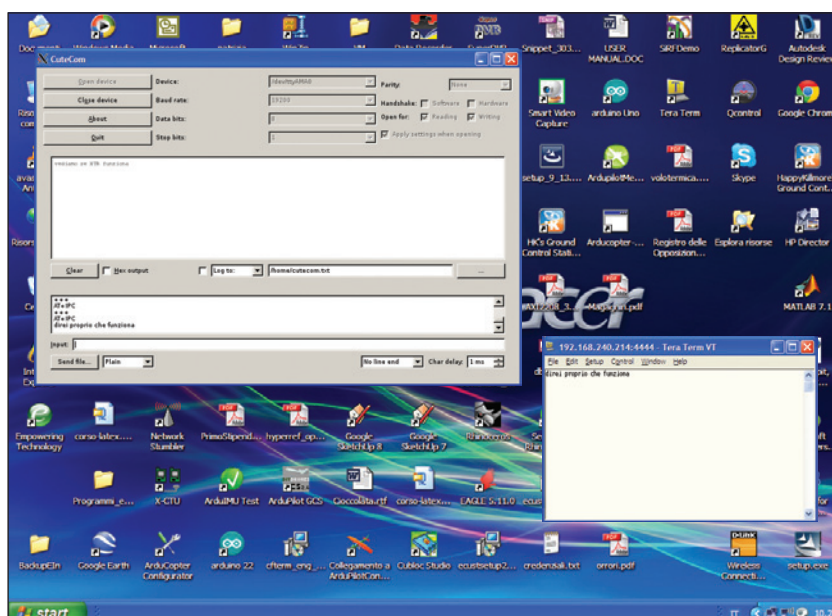


Fig. 6

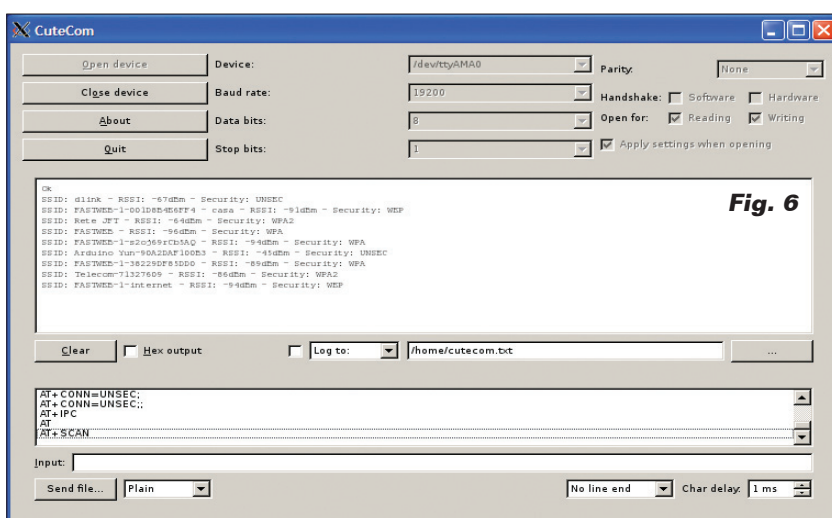
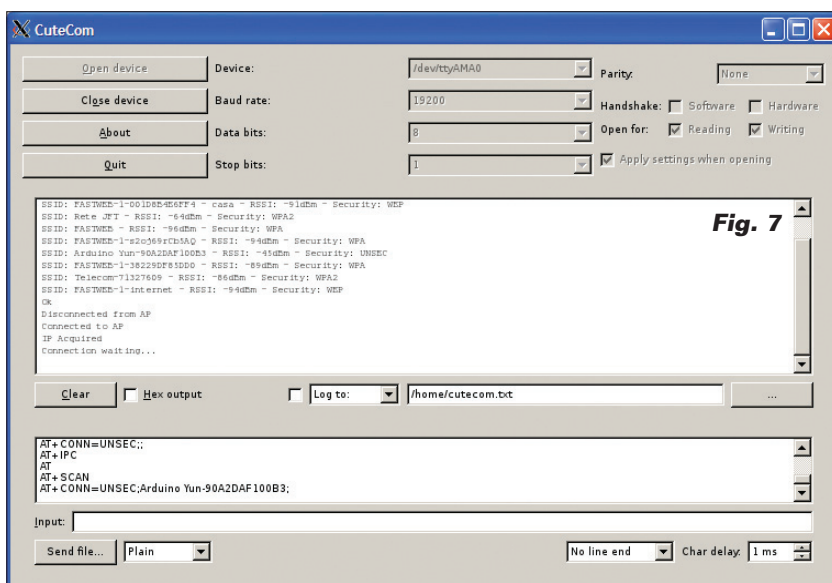


Fig. 7



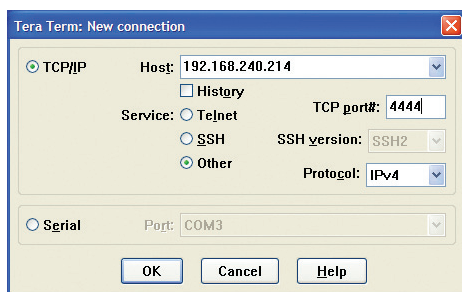


Fig. 8

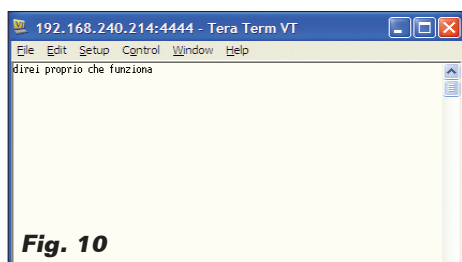


Fig. 10

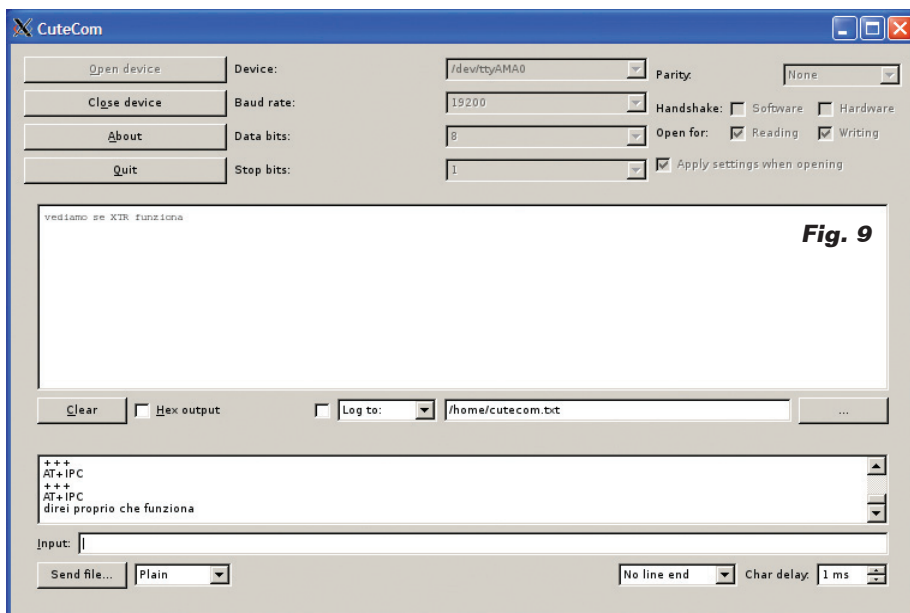


Fig. 9

Aurel WiFi: Version 0.1-1.10

I comandi di configurazione sono descritti nel riquadro dedicato "Comandi AT di configurazione XTR-WiFi".

Diamo il comando:

AT per verificare se il modulo è attivo, dovremmo ottenere la risposta "OK".

Ora, con il comando:

AT+SCAN otteniamo la lista degli access point visibili dal modulo (Fig. 6). Individuiamo Arduino Yún e componiamo il comando di connessione all'AP:

AT+CONN=UNSEC;Arduino Yun-90A2DAF100B3;

Configuriamo la connessione senza password, l'SSID è composto da Arduino Yún più il MAC address di Arduino. Impostiamo il modulo come server sulla porta 4444:

AT+SRVR=4444

Con questo comando terminiamo la configurazione del modulo.

Nell'area di risposta di "cutecom" dovremmo vedere i messaggi che dicono che il modulo è correttamente configurato e si mette in attesa di comunicazioni in arrivo (Fig. 7). Per verificare se tutto funziona correttamente utilizziamo sul PC il programma "Tera Term", un programma di emulazione

terminale gratuito, già presentato sulla rivista in diverse occasioni. Lo apriamo con i parametri visibili in Fig. 8. Dalla finestra di comunicazione inseriamo un messaggio, per esempio "Vediamo se XTR funziona". Dovremmo vedere il messaggio apparire nella finestra di "cutecom" (Fig. 9). Rispondiamo con un messaggio nel campo Input di cutecom che vedremo apparire nell'area di risposta di "Tera Term" (Fig. 10). Bene per ora terminiamo qui ma non mancheremo di utilizzare uno o più di questi moduli in progetti distribuiti, magari con diverse tipologie di dispositivi collegati tra loro in rete wireless. ■

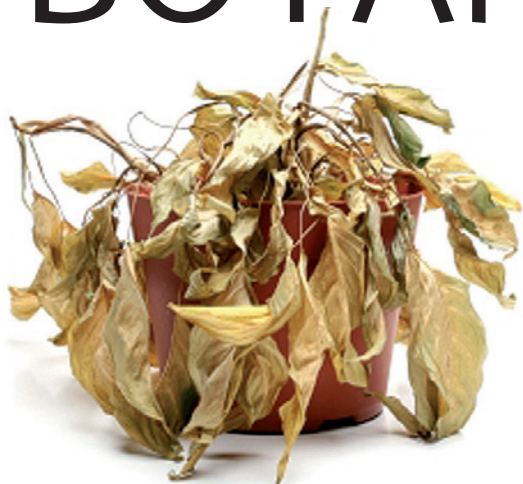
RM ELETTRONICA

Forniture per hobbisti, scuole e industria

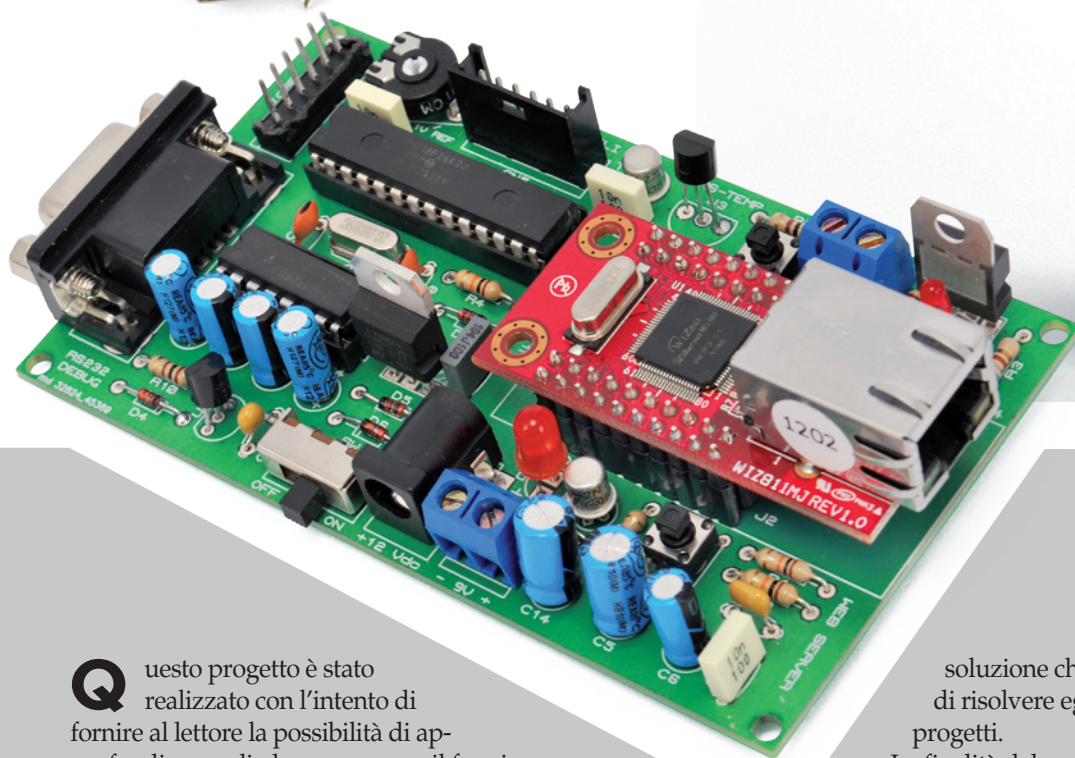
Distributore autorizzato:

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

WEB SERVER BOTANICO



Sistema embedded dedicato alla gestione in remoto, via web, dell'irrigazione ottimale di piante domestiche.



di FULVIO DE SANTIS

Questo progetto è stato realizzato con l'intento di fornire al lettore la possibilità di approfondire meglio la conoscenza, il funzionamento e le modalità di programmazione di un PIC e dei suoi moduli quali SPI, ADC e Timer, assieme al modulo ethernet WIZ811MJ della Wiznet; la medesima applicazione è chiaramente possibile sfruttando altre piattaforme embedded e per la prototipazione rapida, ma abbiamo intenzionalmente scelto la soluzione di Microchip e del linguaggio C per non "dimenticare" una

soluzione che per anni ha permesso di risolvere egregiamente numerosi progetti.

La finalità del progetto è anche rendere partecipe il lettore quanto più possibile alle scelte metodologiche progettuali e agli strumenti utilizzati per realizzarlo. Un'attenta lettura delle parti relative alla progettazione dei moduli sicuramente ne darà ai lettori la padronanza applicativa e di conseguenza la possibilità di utilizzarli per i propri futuri personali progetti e credo che per un

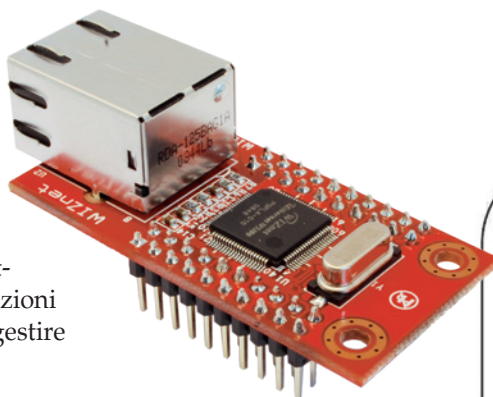
CARATTERISTICHE TECNICHE del modulo WIZ811MJ

- TCP/IP, MAC protocol layer: W5100
- Physical layer: Included in W5100
- Connector: MAG-JACK (RJ45 with Transformer)
- Supports 10/100 Base TX
- Supports half/full duplex operation
- Supports auto-negotiation and auto cross-over detection
- IEEE 802.3/802.3u Compliance
- Operates 3.3V with 5V I/O signal tolerance
- Supports network status indicator LEDs
- Includes Hardware Internet protocols: TCP, IP Ver.4, UDP, ICMP, ARP, PPPoE, IGMP
- Includes Hardware Ethernet protocols: DLC, MAC
- Supports 4 independent connections simultaneously
- Supports MCU bus Interface and SPI Interface
- Supports Direct/Indirect mode bus access
- Supports Socket API for easy application programming

appassionato di elettronica sia il massimo traguardo che si voglia raggiungere. Siamo sicuri che questo metodo di presentazione dei progetti costituisca un valore aggiunto per gli appassionati i quali, oltre a conoscere dei circuiti prodotti, possono ricavarne un accrescimento culturale, al momento nel settore hobbistico, ma che in futuro può estendersi in campo professionale.

Veniamo, dunque, al nostro progetto.

Ormai in tutte le moderne abitazioni è presente una connessione Internet a cui si agganciano gadget personali come smartphone e tablet che, nel loro insieme, offrono un'ampia gamma di applicazioni per il divertimento ma anche, spesso, di grande utilità. Se a tutto ciò aggiungiamo un altro elemento - un server con interfaccia Ethernet - e lo colleghiamo via cavo LAN al router ADSL, otteniamo un web server a cui gli altri dispositivi possono interfacciarsi per ottenere dati o per attuare funzioni personalizzate, magari per gestire



allarmi, rilevare dati climatici, accendere o spegnere lampade, controllare l'irrigazione del giardino o dei balconi e terrazzi e così via.

In questo articolo viene descritto il "web server botanico": un sistema embedded realizzato per la gestione remota via web dell'irrigazione ottimale di piante domestiche. Il Web Server botanico sfrutta la connessione Internet e un qualsiasi browser di navigazione come I.E., Mozilla ecc. per controllare da remoto le condizioni climatiche ambientali delle piante (temperatura ambiente e umidità del terreno) e, conseguentemente, può attivare sistemi per irrigarle se l'umidità del terriccio e la temperatura ambiente lo richiedono, agendo sull'accensione o spegnimento di una qualsiasi centralina di irrigazione automatica ad esso collegata. Il cuore del sistema è il modulo di rete WIZ811MJ composto dal chip W5100 al cui interno è contenuto un hardware TCP/IP cablati - incluso PHY - un connettore RJ45 filtrato con trasformatore e logica varia. Nel Web Server Botanico il controller PIC 18F26K20 di Microchip svolge le funzioni di comunicazione con il modulo WIZ811MJ mediante l'interfaccia SPI.

In Fig. 1 è riportato lo schema a blocchi sintetico del modulo WIZ811MJ, mentre in Fig. 2 lo schema a blocchi con il dettaglio dell'interno del chip W5100. Come si evince dallo schema a blocchi di Fig. 2, il W5100 ha un modulo TCP/IP cablati con incluso il controller Ethernet PHY (physical layer); ciò consente la realizzazione di una più semplice applicazione basata su protocollo TCP/IP. Dal layer PHY del

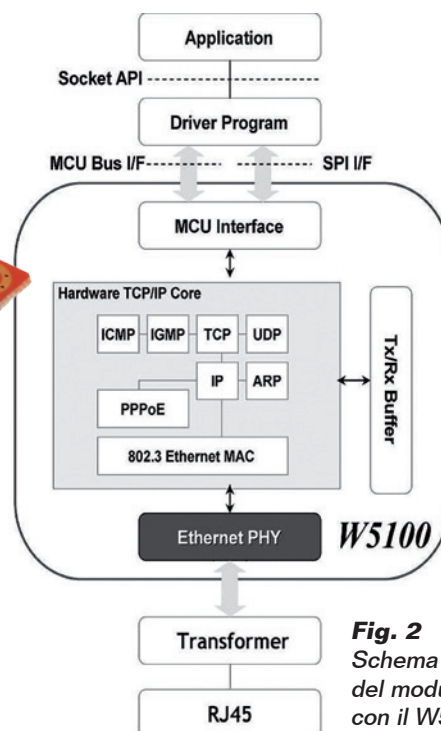
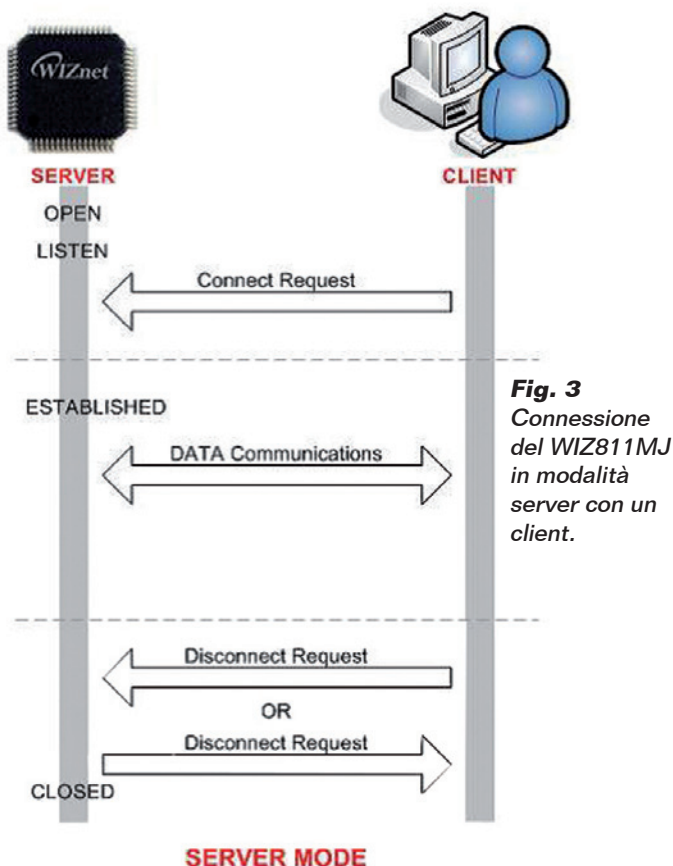


Fig. 2
Schema a blocchi del modulo WIZ811MJ con il W5100.



W5100, mediante trasformatore e connettore RJ45, il WIZ811MJ si connette al mondo esterno via Ethernet. Un'interfaccia MCU consente il collegamento del modulo ad un microcontrollore esterno, nel nostro caso al PIC 18F26K20, mediante bus di comunicazione SPI. In Fig. 3 è riportato il processo di connessione SERVER-CLIENT in cui il modulo WIZ811MJ è configurato per il funzionamento in modalità server. Il server basato sul modulo WIZ811MJ utilizza il protocollo HTTP (Hypertext Transfer Protocol) per interagire con il client (browser di navigazione Internet) attraverso il protocollo TCP/IP mediante lo scambio di informazioni sotto forma di testo HTML (Hypertext Markup Language). Il server opera rimanendo in attesa di richieste HTTP di tipo GET o POST provenienti dal client (browser) sulla porta 80 standard HTTP. Inviata la richiesta dal client, il server risponde inviando al client un header del tipo "HTTP/1.0 200 OK and Content-Type: text/html" seguito dal messaggio HTML (body); successivamente al trasferimento del messaggio, il server disconetterà automaticamente la connessione stabilita con il client. Di seguito un esempio di richiesta inviata dal client (in questo caso il browser Mozilla) e la relativa risposta del server:

Richiesta del Client:

```
GET / HTTP/1.1
Host: 192.168.2.101
User-Agent: Mozilla/5.0 (Windows; U; Windows
NT 5.1; en-US; rv:1.9.2.3) Gecko/20
100401 Firefox/3.6.3
Accept: text/html,application/
xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
```

Risposta del Server HTTP:

```
HTTP/1.0 200 OK
Content-Type: text/html
<html>
<body>
<span style="color:#0000A0">
<h1>Web Server Botanico</h1>
<h3> Controllo Temperatura ambiente e Umidità del
terreno</h3>
<p><form method="POST">
<strong>Temp: <input type="text" size=2 va-
lue="26"> <sup>O</sup></strong>
<strong>Umid: <input type="text" size=2 va-
lue="70">
<p><input type="radio" name="radio" value="0"
>Water OFF
<br><input type="radio" name="radio" value="1"
checked>Water ON
</strong><p>
<input type="submit">
</form></span>
</body>
</html>
```

Ottenuta la risposta dal server, il client tradurrà il testo HTML ricevuto visualizzandolo sullo schermo del browser; nel nostro caso apparirà oltre all'installazione dell'applicazione anche il valore della temperatura ambiente e di umidità del terreno della pianta sotto misurazione. Appariranno inoltre due pulsanti: "Water OFF", spuntato di default, e "Water ON" non spuntato, che consentono di selezionare dal browser (richiesta POST al server) l'accensione o lo spegnimento dell'eventuale centralina di irrigazione comandata dal PIC 18F26K20 attraverso una delle linee I/O della PORTB disponibili. In Fig. 4 il diagramma di flusso dell'applicazione SERVER (WIZ811MJ)/CLIENT (browser).

DESCRIZIONE DELLO SCHEMA ELETTRICO

Vediamo ora com'è composto il Web Server Botanico analizzando lo schema elettrico. Il controller PIC 18F26K20 U1 mediante l'interfaccia di comunicazione SPI gestisce la configurazione e le attività di rete del modulo WIZ811MJ U4 e nel contempo, mediante l'integrato U2 MAX3232 di Maxim - un traslatore di livelli TTL/RS232 - e il connettore RS232 CN4, colloquia con un PC per le attività di debug del firmware. Il PIC, mediante due canali analogici del modulo ADC a 10 bit, acquisisce e converte in formato numerico il valore di temperatura in °C mediante il sensore integrato U3 LM35, mentre acquisisce sul secondo canale il tasso percentuale di umidità assoluta mediante il sensore costituito da T1, R5, R6 e dal connettore CN1 a cui sono collegati i conduttori della sonda costituita da due barrette metalliche infisse nel terreno da controllare. I dati di temperatura ambiente e umidità rilevati saranno inviati via web dal modulo server WIZ811MJ a un qualsiasi dispositivo client che, mediante browser, abbia richiesto la connessione al server. L'utente che sta utilizzando il client, in base ai dati ricevuti, potrà selezionare l'azione da far effettuare al server relativamente all'accensione o allo spegnimento di una centralina di irrigazione utilizzando una delle sei linee I/O disponibili al connettore CN5 e da questo connesse a una eventuale scheda relè esterna. La sezione di alimentazione usa una sorgente esterna a 12Vcc ed è costituita dal regolatore di tensione a 5V U5, 7805 la cui uscita stabilizzata viene inviata al regolatore di tensione a 3,3V LM1086, U6; questo, a sua volta, alimenta la restante circuiteria tranne il sensore LM35 U3 che è alimentato a +5V. Il generatore di corrente costante U7 LM317 alimenta una batteria in tampone da 9V che si inserisce per sopperire ad eventuali interruzioni dell'energia elettrica di rete.

DESCRIZIONE DEL SOFTWARE

Configurazione del server HTTP W5100

Per configurare e controllare il server occorre leggere e scrivere nei registri di controllo del socket 0 (il socket,

o canale di comunicazione, dei quattro disponibili utilizzato in questo progetto) del Wiznet W5100 e nel buffer di memoria relativo di dimensione 2 kB per RX e TX dello stesso W5100.

Nella **Tabella 1** ricavata dal

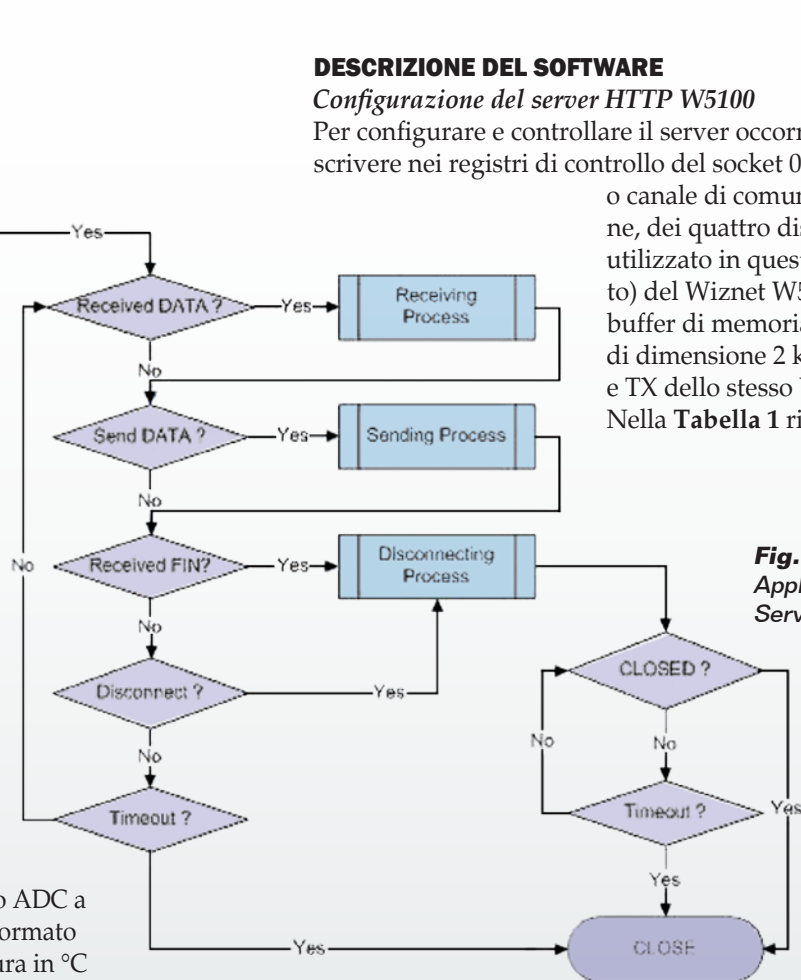


Fig. 4
Applicazione
Server-Client.

datasheet del W5100 sono indicati i registri del socket 0 e i relativi indirizzi.

L'impostazione di questi registri e la lettura/scrittura nei buffer RX e TX del W5100 vengono effettuate mediante comunicazione seriale SPI (Serial Peripheral Interface) fra il PIC 18F26K20 in modalità Master e il chip slave W5100 (Fig. 5).

Tabella 1 - Registri del W5100

Registers	Description	Address	Usage
S0_MR	Socket 0 Mode Register	0x0400	Select the protocol to be used i.e. TCP, UDP, PPP
S0_CR	Socket 0 Command Register	0x0401	Used for controlling the socket operation i.e. OPEN, LISTEN, CONNECT, DISCONNECT, CLOSE, SEND, RECEIVE
S0_SR	Socket 0 Status Register	0x0403	Contain the socket operation status value i.e. CLOSE, INITIAL, LISTEN, ESTABLISHED, CLOSED
S0_PORT	Socket 0 Source Port	0x0404 0x0405	Hold the TCP/IP Port value e.g. 80 for the standard HTTP server
S0_TX_FSR	Socket 0 TX Free Size Register	0x0420 0x0421	Maximum TX Free Memory buffer available for sending data
S0_TX_RD	Socket 0 TX Read Pointer	0x0422 0x0423	The start pointer at the available TX memory buffer for sending data
S0_TX_WR	Socket 0 TX Write Pointer	0x0424 0x0425	The end pointer mark at the TX memory buffer, this value will always be the same as the S0_TX_RD after the "SEND" command
S0_RX_RSR	Socket 0 RX Received Size	0x0426 0x0427	Contain the receiving data size at the RX memory buffer
S0_RX_RD	Socket 0 RX Read Pointer	0x0428 0x0429	The start pointer of the received data at the RX memory buffer for reading the data
2 KB Socket 0 TX Memory Buffer 0x0400 to 0x04FF			
2 KB Socket 0 RX Memory Buffer 0x0600 to 0x06FF			

The Wiznet W5100 Socket 0 Registers used for the HTTP Server

Impostazioni del modulo SPI del PIC 18F26K20

In questo progetto mediante il registro di controllo **SSPCON1** e il registro di stato **SSPSTAT** il modulo SPI del PIC 18F26K20 viene impostato in modalità **MASTER Mode 0**, modalità supportata dal W5100. Nella **Tabella 2** sono indicati i registri e i pin del modulo SPI.

Nella **Tabella 3** sono riportate le modalità di configurazione dei registri **SSPCON1** e **SSPSTAT**.

Per prima cosa, mediante i registri **TRISC** e **TRISA**, configureremo la direzione delle porte I/O del PIC: **RC5=SDO** e **RC3=SCK** come uscite e **RC4=SDI** come ingresso, mentre **RA5=SS** sarà il pin di controllo di uscita SPI master per l'abilitazione/disabilitazione del chip slave W5100.

Dopo l'inizializzazione delle porte, mediante il registro **SSPCON1**, attiviamo il modulo SPI del PIC in

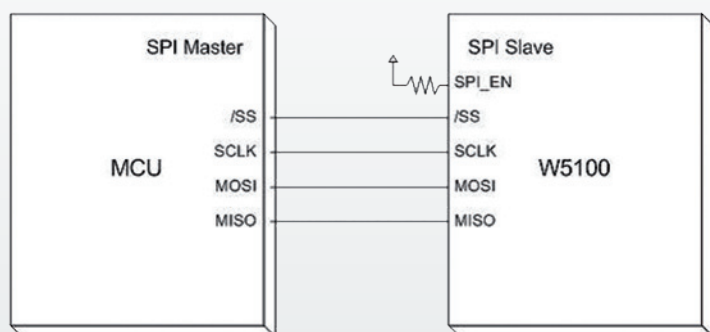


Fig. 5 - Collegamento SPI Master-Slave.

MASTER Mode settando a 1 il bit **SSPEN** e impostiamo la frequenza del clock SPI master configurando i bit **SPM<3:0>=0001** per $F_{osc}/16$.

La modalità di ricezione/trasmisione dati SPI, o SPI bus Mode, è impostata in **Mode 0** dai bit **CKP** (Clock Polarity) del registro **SSPCON1** e **CKE** (Clock Select) del registro **SSPSTAT** (Tabella 4). Il PIC 18F26K20 può

Tabella 2 - Pin e Registri SPI del 18F26K20.

Pin	Descrizione	Direzione	Porte I/O
SDO	Serial Data Out	Out	RC5
SDI	Serial Data In	In	RC4
SCK	Serial Clock (SPI Master Mode)	Out	RC3
SS	Serial Select (Master Mode)	Out	RA5

Registri	Descrizione
SSPCON1	Control 1 Register
SSPSTAT	Status Register
SSPBUF	Serial Receive/Transmit Buffer

Inizializzazione Usart con le funzioni `uart_init()` e `uart_flush()`

```

/*****uart_init*****/

//Inizializzazione dell'UART
void uart_init(void) {

/*Inizializzazione USART
**
** Interruzioni TX e RX disattivate
** Formato 8 bit - 1 bit di stop
**spbrg=64 per Baud rate 19200bit/s (Quarzo 20MHz)
**Asynchronous mode
**High speed
**Fosc / (16 * (spbrg + 1))
**
*/
OpenUSART(USART_TX_INT_OFF &
           USART_RX_INT_OFF &
           USART_ASYNC_MODE &
           USART_EIGHT_BIT &
           USART_CONT_RX &
           USART_BRGH_HIGH &
           USART_ADDEN_OFF,
           64);
}

/*****uart_flush*****/

//Funzione che svuota il registro RX dell'UART
void uart_flush(void)
{
    unsigned char dummy;//dummy è una variabile di appoggio fittizia

    while (DataRdyUSART()) dummy=RCREG; // DataRdyUSART=1 _
    se dato presente nel RCREG
}

```

gestire 4 SPI bus Mode e in questo progetto è stato selezionato il Mode 0, gestibile anche dalla periferica SPI del W5100.

L'ultimo bit **SMP** del registro **SSPSTAT** viene utilizzato per controllare il modo in cui campionare i dati di input. Con il bit a "0" logico significa campionare i dati in arrivo a metà del tempo di uscita dei dati, men-

SSPCON1 - MSSP Control 1 Register (SPI Mode)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7				bit 0			

SMP<3:0>	Description
0011	SPI Master mode, clock = TMR2 output/2
0010	SPI Master mode, clock = FOSC/64
0001	SPI Master mode, clock = FOSC/16
0000	SPI Master mode, clock = FOSC/4

SSPSTAT - MSSP Status Register (SPI Mode)

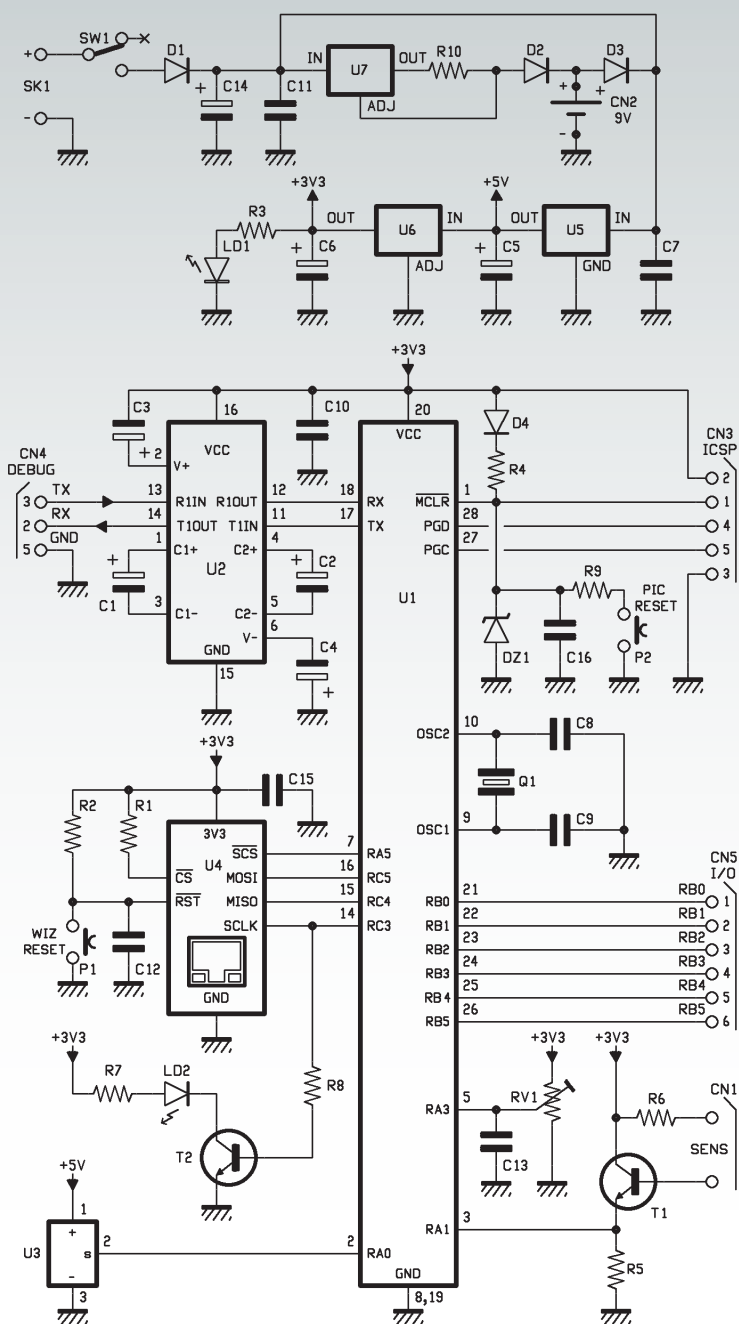
R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/A	P	S	R/W	UA	BF
bit 7				bit 0			

SMP - Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode

CKE - Clock Select bit
1 = Transmit occurs on transition from active to idle clock state
0 = Transmit occurs on transition from idle to active clock state

Tabella 3 - Registri SSPCON1 e SSPSTAT.

[schema ELETTRICO]



tre con il bit impostato a "1" i dati in arrivo saranno campionati alla fine del tempo di uscita dei dati. In questo progetto SMP viene settato a "0". La trasmissione e la ricezione dei dati viene effettuata dal registro **SSBUF**, ciò significa che caricando i dati in questo registro inizierà la trasmissione dei dati. Dopo 8 cicli di clock, il dato di 8 bit presente in questo registro sarà shiftato fuori dal pin **SDO** verso il chip slave abilitato, ma nello stesso tempo riceverà il dato a 8 bit dal chip slave attraverso il pin **SDI**. È possibile

monitorare il ricevitore controllando in polling lo stato del bit BF (Buffer Full) del registro **SSPSTAT**. Prima di iniziare la trasmissione dati verso il chip slave occorre assicurarsi di aver attivato il pin Chip Select (**SS=0**) verso lo slave e di disattivarlo dopo la fine della trasmissione del dato.

Inizializzazione del W5100

Per inizializzare il W5100 dobbiamo inserire i dati di rete scrivendo nei registri comuni **MR** (Mode Register), **SUBR** (Subnet mask Register), **SAR** (Source Hardware Register), **SIPR** (Source IP Register), **RMSR** (Receive Memory Size Register) e **TMSR** (Transmit Memory Size Register).

Nella **Tabella 5** è riportata la mappa di memoria in cui sono indicati i banchi di memoria dei registri e dei buffer RX e TX del W5100.

Dato che tutti gli indirizzi dei registri del W5100 sono a 16 bit, mentre la comunicazione SPI del PIC 18F26K20 è a 8 bit, per realizzare le operazioni di scrittura e lettura dobbiamo passare prima gli otto bit più significativi (MSB), poi i bit meno significativi (LSB) per ognuno degli indirizzi dei registri del W5100. Il W5100 utilizza due differenti codici di comando: **0xF0** per la scrittura e **0x0F** per la lettura. La funzione che realizza l'operazione di scrittura è **SPI_Write()** mentre per la lettura è **SPI_Read()**. Il registro **SAR** è noto come MAC address (Media Access Control).

Questo registro rappresenta l'unico identificativo hardware nella rete.

Il MAC address è assegnato e gestito dall'Institute of Electrical and Electronics Engineers (IEEE) per ogni NIC (Network Interface Card) dove i primi 3 byte dei 6 sono usati per identificare l'organizzazione che ha inviato l'identificatore e sono noti come OUI (Organizationally Unique Identifier). Di seguito una lista di OUI di Atmel, Microchip e Wiznet:

00-04-25 (hex) Atmel Corporation

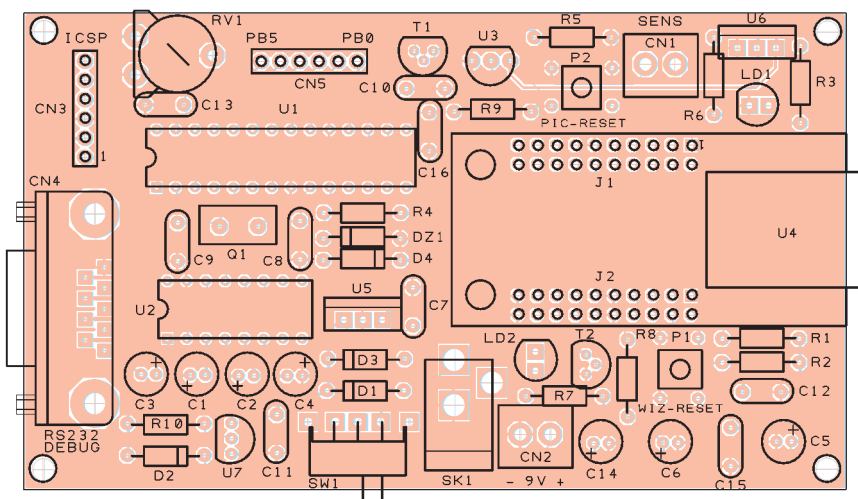
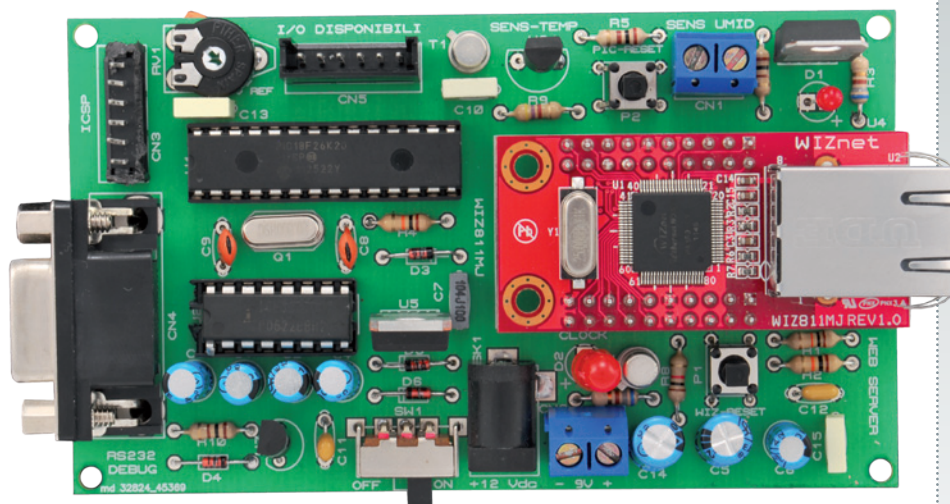
000425 (base 16) Atmel Corporation
Multimedia & Communications Group
2200 Gateway Centre, Suite 201
Morrisville NC 27560
UNITED STATES

00-04-A3 (hex) Microchip Technology, Inc.
0004A3 (base 16) Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler AZ 85224
UNITED STATES

Piano di **MONTAGGIO**

Elenco Componenti:

R1: 10 kohm
 R2: 10 kohm
 R3: 470 ohm
 R4: 10 kohm
 R5: 100 ohm
 R6: 100 ohm
 R7: 470 ohm
 R8: 1 kohm
 R9: 470 ohm
 R10: 150 ohm
 RV1: Trimmer 10 kohm MO
 C1: 1 μ F 63 VL elettrolitico
 C2: 1 μ F 63 VL elettrolitico
 C3: 1 μ F 63 VL elettrolitico
 C4: 1 μ F 63 VL elettrolitico
 C5: 10 μ F 63 VL elettrolitico
 C6: 10 μ F 63 VL elettrolitico
 C7: 330 nF 100 VL poliestere
 C8: 22 pF ceramico
 C9: 22 pF ceramico
 C10: 10 nF 100 VL poliestere
 C11: 100 nF multistrato
 C12: 100 nF multistrato
 C13: 100 nF 100 VL poliestere
 C14: 220 μ F 16 VL elettrolitico
 C15: 10 nF 100 VL poliestere
 C16: 100 nF 100 VL poliestere
 Q1: Quarzo 20 MHz
 U1: PIC18F26K20 (MF1085)
 U2: MAX3232
 U3: LM35
 U4: WIZ811MJ (WIZNET)
 U5: 7805
 U6: LD10863V3
 U7: LM317L
 T1: 2N2222
 T2: 2N2222
 P1: Microswitch
 P2: Microswitch
 LD1: LED 5 mm rosso
 LD2: LED 5 mm rosso
 SW1: Deviatore a slitta 90°



D1 ÷ D4: 1N4148
 DZ1: Zener 6,8V/400mW

Varie:
 - Zoccolo 8+8
 - Zoccolo 14+14

- Strip maschio 6 poli (2 pz.)
 - Strip femmina 10 poli (4 pz.)
 - Plug alimentazione
 - Morsetto 2 poli (2 pz.)
 - Connettore DB9 Femmina
 - Circuito stampato

00-08-DC (hex) Wiznet
 0008DC (base 16) Wiznet
 5F Simmtech bldg., 228-3,
 Nonyhun, Kangnam
 Seoul 135-830
 KOREA, REPUBLIC OF

Ora è necessario assegnare un MAC address al W5100 in quanto non ha un proprio identificativo MAC, a meno di registrare l'OUI al IEEE, ma ciò per scopi hob-

bystici sarebbe molto costoso. La soluzione è quella di utilizzare il MAC address della NIC (scheda di rete) del proprio PC, modificando l'ultimo byte dei 6 componenti il MAC e sperando che sia l'unico MAC address della rete utilizzata. Si individua il MAC address del PC andando nella finestra di comando e al prompt scrivendo il comando "ipconfig /all"; apparirà l'informazione del MAC nel formato di 6 byte in notazione esadecimale, dopodiché si cambierà l'ultimo byte. Il Wiznet W5100 supporta fino a quattro canali (so-

Inizializzazione SPI del PIC

```

.....
//Imposto le porte PORTA e PORTC per la periferica SPI del PIC18F26K20
TRISAbits.TRISA5 = 0; // RA5/SS - Output (Chip Select)
TRISCbits.TRISC5= 0; // RC5/SDO - MOSI, Output (Serial Data Out)
TRISCbits.TRISC4= 1; // RC4/SDI - MISO, Input (Serial Data In)
TRISCbits.TRISC3= 0; // RC3/SCK - Output (Clock)
.....

/*Inizializzo la periferica SPI
*/
// SPI in Master Mode 0: CKE=1, campionamento dato in ingresso a fine
tempo del dato in uscita: SMP=0
SSPSTAT = 0x40;

//SPI in Master Mode 0: CKP=0, Fosc/16: SSPM<3:0>=0000,SSPEN=1: abilita
i pins SCK,SDO,SDI e SS per SPI.
SSPCON1 = 0x21;
.....

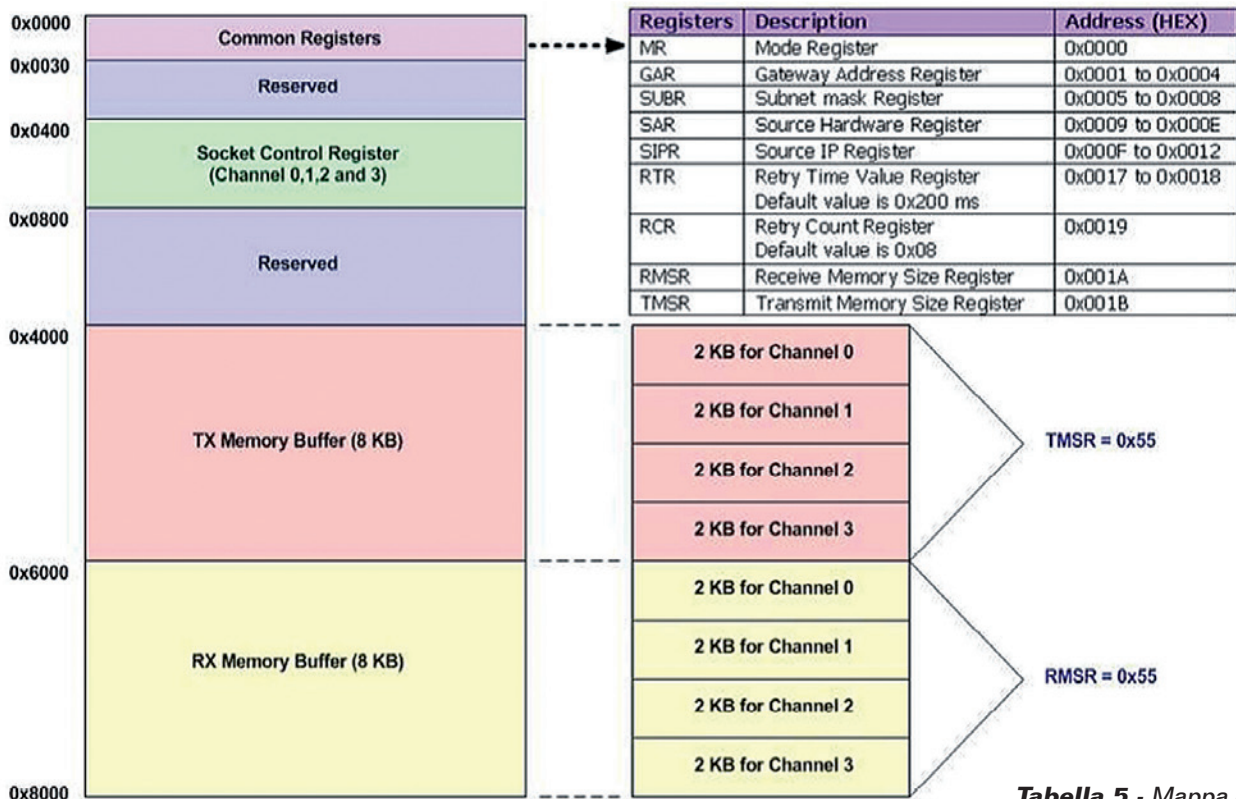
```

cket) simultaneamente, ogni canale ha propri registri per il controllo delle operazioni e ha a disposizione un buffer fino a 8 kB di memoria sia per RX, sia per TX. In questo progetto la dimensione di memoria RX e TX viene impostata a 2 kB mediante scrittura del codice 0x55 nei registri RMSR e TMSR:

```

/*Imposto a 2kB la dimensione di memoria del buffer RX e TX del W5100.*/
SPI_Write(RMSR,0x55);
SPI_Write(TMSR,0x55);

```



The Wiznet W5100 Memory Map

Tutta la procedura di inizializzazione è implementata nella funzione **W5100_Init()** caratterizzata dalla scrittura nei registri di controllo mediante la funzione **SPI_Write ()** per l'impostazione dei parametri di rete. L'assegnazione del protocollo TCP/IP e l'apertura della porta 80 è realizzata mediante la funzione **socket()**:

```

.....
// Imposto il Socket 0 in modalità ethernet
SPI_Write(S0_MR,eth_protocol);

//Assegno la porta TCP 80
SPI_Write(S0_PORT,((tcp_port & 0xFF00) >> 8 ));
SPI_Write(S0_PORT + 1,(tcp_port & 0x00FF));
// Ora apro il Socket 0
SPI_Write(S0_CR,CR_OPEN);
.....

```

Tabella 4 - Impostazioni Master Mode.

SPI Bus Mode	Control Bits States		Description
	CKP	CKE	
0	0	1	Clock Idle State = High, Transmit occurs on transition from active to Idle clock state
1	0	0	Clock Idle State = High, Transmit occurs on transition from Idle to active clock state
2	1	1	Clock Idle State = Low, Transmit occurs on transition from active to Idle clock state
3	1	0	Clock Idle State = Low, Transmit occurs on transition from Idle to active clock state

Tabella 5 - Mappa di memoria del W5100.

Dopo l'apertura del canale il registro di stato (**S0_SR**) sarà nello stato **SOCK_INIT**. Successivamente il server si porterà nello stato di ascolto mediante l'invio del comando **CR_LISTEN** attuato con la funzione **listen()** e il registro di stato sarà **SOCK_LISTEN**. Da questo momento il server W5100 è pronto per ricevere una richiesta dal client (browser). Stabilita la connessione con il server, il client invierà la richiesta standard HTTP al server; il registro di stato sarà **SOCK_ESTABLISHED**. Successivamente si esaminerà la dimensione dei dati ricevuti nel buffer RX del socket 0 mediante la funzione **recv_size()**:

```
unsigned int recv_size(void) {
```

```
//Questa funzione ritorna il valore della dimensione dei dati ricevuti nella memoria RX.
return ((SPI_Read(S0_RX_RSR) & 0x00FF) << 8) + SPI_Read(S0_RX_RSR + 1);
```

```
}
```

L'esistenza di dati nel buffer RX del W5100 consente la lettura degli stessi e il relativo inserimento nella memoria RAM (buf) del PIC mediante la funzione **recv()**. Dopo aver letto tutti i dati, viene aggiornato il puntatore della memoria RX del W5100 per predisporre il banco di memoria per una successiva ricezione di dati, dopodiché viene dato il comando **CR_RECV** per completare il processo di ricezione.

Il server esaminerà i dati ricevuti e predisporrà la relativa risposta HTML al client con i dati di temperatura ambiente e umidità assoluta del terreno e la selezione del comando di accensione o spegnimento della centralina di irrigazione.

Infatti, dopo aver copiato il testo HTML nella memoria RAM del PIC (buf), i dati vengono inviati al client mediante la funzione **send()** che invierà i dati nella memoria TX del W5100 verificandone preventivamente lo spazio disponibile nel registro TX. Come per il puntatore RX, anche il puntatore TX sarà aggiornato per predisporre una successiva scrittura nel banco TX. Il successivo comando **CR_SEND** completerà il processo di scrittura.

Dopo l'invio della risposta al client, il server mediante la funzione **disconnect()** chiuderà la connessione verso il client.

Durante il loop infinito nel "**main**" il server aprirà il socket rimanendo in ascolto per una eventuale nuova richiesta del client.

Per l'acquisizione dei valori di temperatura e umidità il software prevede l'utilizzo della subroutine **High_Int_TMR0** che, mediante l'interrupt di overflow del **Timer0** del PIC, ogni 10 ms legge e converte i valori di temperatura e umidità generati dai sensori.

Di seguito viene riportata la descrizione dell'impostazione

del modulo **ADC** e del **Timer0** del PIC. Vediamo ora come sono stati impostati i due importanti moduli **ADC** e **Tmr0** utilizzati all'interno della subroutine **High_Int_TMR0**.

Impostazione del modulo **Timer0** del PIC 18F26K20

Come detto precedentemente, con questo progetto si vogliono rilevare ad intervalli regolari e prestabiliti i valori di temperatura ambiente e di umidità assoluta del terreno ove sono localizzate piante, fioriere, giardini, eccetera, e conseguentemente decidere l'attivazione da remoto di una centralina di irrigazione. Questi intervalli di misurazione vengono attivati mediante interrupt prodotto ogni **10ms** dall'overflow di conteggio del modulo **Timer0** di cui è dotato il PIC. Il **Timer0** viene impostato con clock esterno (quarzo da 20 MHz), registro di conteggio a 8 bit e prescaler a 256. Per le impostazioni del modulo **Timer0** viene utilizzata la libreria "timers.h" del PIC e l'inizializzazione del **Timer0** viene effettuata dalla funzione **Timer0_init()**:

La funzione **SPI_Write()** consente al PIC di scrivere nella memoria TX del W5100.

```
/******SPI_Write******/

void SPI_Write(unsigned int addr,unsigned char data)

{
// Abilito il W5100 per l'attività SPI
// SS=0:imposto a 0 la posizione RA5 di PORTA
LATAbits.LATA5=0;

// Avvio trasmissione dell'OPCODE di scrittura del Wiznet W5100
SSPBUF = WIZNET_WRITE_OPCODE;

// Attendo trasmissione completa
while (!DataRdySPI());

// Avvio la trasmissione del byte alto dell'indirizzo del Wiznet W5100
SSPBUF= (addr & 0xFF00) >> 8;

// Attendo trasmissione completa
while (!DataRdySPI());

// Avvio la trasmissione del byte basso dell'indirizzo del Wiznet W5100
SSPBUF= addr & 0x00FF;

// Attendo trasmissione completa
while (!DataRdySPI());

// Avvio la trasmissione del dato
SSPBUF= data;

// Attendo trasmissione completa
while (!DataRdySPI());

// Disattivo il W5100
LATAbits.LATA5=1;

}
```

La funzione SPI_Read() consente al PIC di leggere nella memoria RX del W5100

```

/*****SPI_Read*****/
unsigned char SPI_Read (unsigned int addr)
{
    // Attivo il W5100
    LATAbits.LATA5=0;

    // Avvio trasmissione dell'OPCODE di lettura del Wiznet W5100
    SSPBUF = WIZNET_READ_OPCODE;

    // Attendo trasmissione completa
    while (!DataRdySPI());

    // Avvio la trasmissione del byte alto dell'indirizzo del buffer RX del W5100
    SSPBUF = (addr & 0xFF00) >> 8;

    // Attendo trasmissione completa
    while (!DataRdySPI());

    // Avvio la trasmissione del byte basso dell'indirizzo del buffer RX del W5100
    SSPBUF= addr & 0x00FF;

    // Attendo trasmissione completa
    while (!DataRdySPI());

    // Avvio la trasmissione del dato dummy per svuotare il registro SSPBUF
    SSPBUF= 0x00;

    // Attendo trasmissione completa
    while (!DataRdySPI());

    // Disattivo il W5100
    LATAbits.LATA5=1;

    //Ritorna il dato letto
    return (SSPBUF);
}

```

//Inizializzazione del Timer0

void Timer0_init(void) {

/*Inizializzo il timer0

**

**Contatore del Timer0 a 8 bit

**Clock interno

**Prescaler a 256

**Incremento del Timer0 sul fronte di discesa del clock.

*/

OpenTimer0(T0_8BIT&
T0_SOURCE_INT&
T0_PS_1_256&
T0_EDGE_FALL);

}

Di seguito la procedura di calcolo per la definizione del valore da impostare nel registro di conteggio TMR0 per ottenere un interrupt preciso ogni 10mS. Il periodo del clock utilizzato è pari a 0,05 µs ovvero:

$$T_{osc} = 1/F_{osc} = 1/(20 \cdot 10^6) = 0,05 \mu s$$

Dato che il tempo di esecuzione di un'istruzione è pari a quattro cicli di clock, il clock che va a incrementare il Timer0 è pari a 0,2 µs, ovvero:

$$T_{TIMER0} = T_{osc} \cdot 4 = 0,05 \cdot 4 = 0,2 \mu s$$

Il clock finale ritardato dal prescaler sarà:

$$T_{PRESCALER} = T_{TIMER0} \cdot 256 = 0,2 \mu s \cdot 256 = 51,2 \mu s$$

da cui ricaviamo il numero di conteggi che dovrebbe fare il Timer0 per creare l'overflow dopo 10mS:

$$T_{OVERFLOW} = T_{PRESCALER} \cdot N_{CONTEGGI} = 10mS \text{ da cui}$$

$$N_{CONTEGGI} = T_{OVERFLOW} / T_{PRESCALER} = 10 \cdot 10^{-3} / 51,2 \cdot 10^{-6} = 195$$

Ma il Timer0 inizia gli incrementi partendo da 0, quindi per fargli fare 195 passi anziché 256 fino all'overflow, si deve precaricare la differenza fra i conteggi totali e quelli necessari.

Conseguentemente il valore da caricare nei registri TMR0H e TMR0L sarà:

$$T_{TIMER0} = (2^8 + 1) - 195 = 62 \text{ corrispondente in hex a } 0x003E$$

Considerando i cicli delle ulteriori istruzioni nella subroutine si è ritenuto di impostare il codice di precarica a 0x003D.

//Carico il valore 003D nei registri TMR0H e TMR0L per ottenere //un overflow del Timer0 dopo 10mS.

TMR0H=0;

TMR0L=0x3D;

Impostazione del modulo ADC del PIC 18F26K20

Per un corretto impiego del modulo ADC è necessario considerare l'impostazione del prescaler del clock e del moltiplicatore del TAD in base al valore del clock esterno utilizzato dal PIC (in questo caso è FOSC=20MHz) quindi occorre fare delle considerazioni.

Il convertitore ADC è a 10 bit, per convertire un livello di tensione del segnale analogico d'ingresso in forma numerica operando la conversione ad approssimazioni successive, impiega un tempo TADx10bit, essendo TAD il clock del modulo ADC pari a 1/FOSC con prescaler=1; il clock in realtà sarà TADx12 considerando un tempo pari a 2 clock necessari per la scarica del condensatore CHOLD interno al modulo per ogni canale ADC (Fig. 6). Per le impostazioni dei registri dell'ADC, il software prevede l'utilizzo della libreria del PIC "adc.h" e l'inizializzazione dell'ADC viene effettuata dalla funzione ADC_init():

FIGURE 19-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

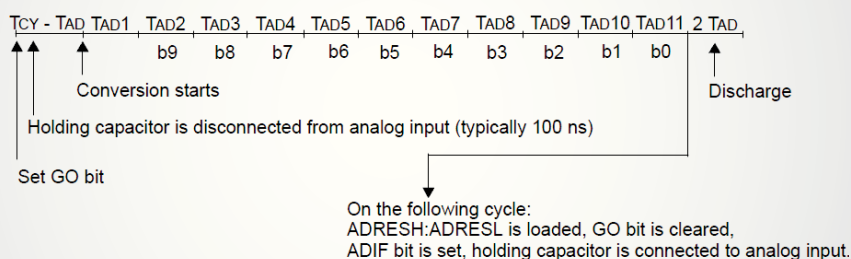


Fig. 6
Conversione A/D
con TACQ=0.

```
//Inizializzazione dell'ADC
void ADC_init(void) {
```

```
/* Apertura del modulo ADC:
```

```
**
```

```
** Fosc / 16, Fosc=20MHz -> FADC=1,25MHz (1/FADC deve essere
```

```
** maggiore del TAD min =0,7uS, vedi datasheet del PIC).
```

```
**Giustificazione a destra, tempo di acquisizione 20TAD.
```

```
**interrupt ADC disattivato,
```

```
**tensione di riferimento: VREF+ esterno (pin RA3) e VSS.
```

```
*/
```

```
OpenADC(ADC_FOSC_16&
```

```
ADC_RIGHT_JUST&
```

```
ADC_20_TAD,
```

```
ADC_CH0&
```

```
ADC_INT_OFF&
```

```
ADC_REF_VREFPLUS_VSS,
```

```
ADC_2ANA );
```

```
}
```

TADC è il tempo totale di acquisizione e conversione di un campione analogico in misurazione all'ingresso di un canale ADC; si calcola con l'equazione

$$T_{ADC} = T_{ACQ} + T_{CNV}$$

dove Tacq (Acquisition Time) è il tempo minimo necessario all'ADC per l'acquisizione dell'informazione analogica. L'equazione di calcolo è riportata nel paragrafo 19.3 del datasheet del PIC. Considerando il caso peggiore di funzionamento del modulo ADC, ossia temperatura di lavoro di 50° e impedenza della sorgente esterna di 10 kohm, il TACQ risulta 7,45 µs. (Per il buon funzionamento dell'ADC è necessario che l'impedenza della sorgente applicata all'ingresso non superi 10 kohm, come prescritto dal datasheet).

TABLE 26-25: A/D CONVERSION REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
130	TAD	A/D Clock Period	0.7	25.0 ⁽¹⁾	µs	TOSC based, VREF ≥ 3.0V, -40°C to +85°C
			0.7	4.0 ⁽¹⁾	µs	TOSC based, VREF ≥ 3.0V, +125°C
			TBD	TBD	µs	A/D RC mode
131	TCNV	Conversion Time (not including acquisition time) (Note 2)	11	12	TAD	
132	TACQ	Acquisition Time (Note 3)	1.4	—	µs	-40°C to +85°C
			TBD	—	µs	0°C ≤ to ≤ +85°C
135	TSWC	Switching Time from Convert → Sample	—	(Note 4)		
136	TDIS	Discharge Time	0.2	—	µs	

Legend: TBD = To Be Determined

Note 1: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

2: ADRES register may be read on the following TCY cycle.

3: The time for the holding capacitor to acquire the "New" input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (RS) on the input channels is 50 Ω.

4: On the following cycle of the device clock.

Fig. 7 - Requisiti temporali dell'ADC.

Funzione High_Int_TMR0() attivata da interrupt causato dall'overflow del Timer0

```
/******High_Int_TMR0*****  
**  
**La seguente funzione High_Int_TMR0 è attivata dall'interrupt di overflow  
**di TMR0 del PIC 18F26K20 che **avviene ogni 10ms. Dopo 20 interrupt, quindi ogni 200ms,  
**effettua la lettura della temperatura ambiente e dell'umidità del terriccio della pianta.  
**  
*****/  
#pragma interrupt High_Int_TMR0  
  
void High_Int_TMR0(void)  
{  
  
static unsigned char tenms=1;  
  
// Leggo l'ADC ogni 200 ms (20 cicli interrupt x 10ms).  
tenms++;  
  
if (tenms >= 20) {  
  
INTCONbits.GIEH = 0; // Disabilito tutti gli Interrupt  
  
//Seleziono il canale CH0 (pin RA0) per convertire la tensione (10mV/°C) dal LM35Z  
SetChanADC(ADC_CH0);  
  
//Avvio la conversione  
ConvertADC();  
  
//Attendo finché la conversione non è completa  
while (BusyADC());  
  
//Leggo il risultato della conversione  
tempvalue = ReadADC();  
  
// ADC = (Vin x 1024) / Vref, Vref = 1 Volt, LM35DZ Out = 10mV/°C  
//Per il valore Tmax=100°C, ADC=1024mV, quindi dividendo tempvalue per 10,24  
// avremo tempvalue = 100°C.  
tempvalue = (int)(tempvalue) / 10.24;  
  
Delay10TCYx (5); //Attendo 10uS prima della prossima conversione  
  
//Seleziono il canale CH1 (pin RA1) per convertire la tensione dal sensore di umidità  
SetChanADC(ADC_CH1);  
  
//Avvio la conversione  
ConvertADC();  
  
//Attendo finché la conversione non è completa  
while (BusyADC());  
  
//Leggo il risultato della conversione  
umidvalue = ReadADC();  
  
//Stabilire v/% del sensore di umidità  
//ADC = (Vin x 1024) / Vref, Vref = 1 Volt, sensore 10mV/% umidità (ipotizziamo 10mV per 1% di umidità)  
//Per il valore Umax=100%, ADC=(10mVx100x1024)/1000mV=1024mV, quindi dividendo umidvalue per 10.24  
//avremo umidvalue = 100%.  
umidvalue = (int)(umidvalue) / 10.24;  
  
//Resetto il contatore delle interruzioni  
tenms=1;  
  
// Riabilito l'interrupt  
INTCONbits.GIEH = 1; // Riabilito tutti gli Interrupt  
  
}  
  
//Ricarico il contatore del Timer0 per l'overflow dopo 10mS  
TMR0H=0;  
TMR0L=0x3D;  
  
//Resetto TMR0IF (TMR0 Overflow Interrupt Flag) per una successiva interruzione  
INTCONbits.TMR0IF=0;  
  
}.....
```

FIGURE 19-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)

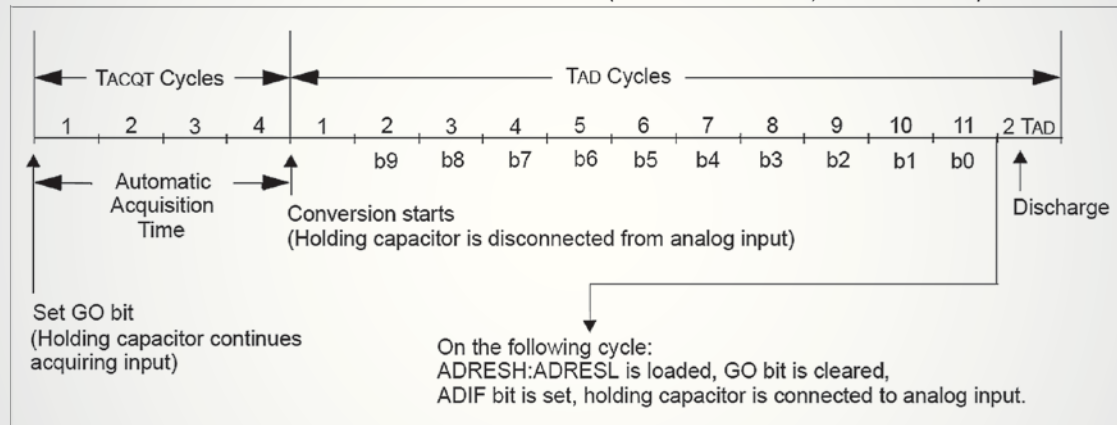


Fig. 8
Conversione
A/D con
TACQ=4TAD.

$T_{CNV} = \text{Conversion Time}$

è il tempo impiegato per la conversione di tutti i 10 bit dell'ADC (escluso il TACQ). TAD è il periodo del clock dell'ADC pari a $1/(\text{FOSC}/\text{valore del prescaler})$.

Nella Fig. 7, ricavata dal datasheet del 18F26K20, il TACQ min. vale $1,4 \mu\text{s}$ per un'impedenza della sorgente di 50 ohm, mentre il T_{CNV} max vale $12TAD$, quindi con $\text{FOSC}=20\text{MHz}$, impostando ad es. il prescaler a 4 e, dalla libreria ADC del PIC, selezionando ADC_12_TAD, si ha

$$TAD = 1/(\text{FOSC}:4) = 0,2 \mu\text{s} \quad TACQ = TAD \times 12 = 0,2 \mu\text{s} \times 12 = 2,4 \mu\text{s}$$

Come si osserverà, si avrebbe così un periodo TAD dell'ADC che consente un TACQ che già coprirebbe il tempo di acquisizione minimo del datasheet $TACQ=1,4 \mu\text{s}$, ma questo TAD già non rispetta il requisito del datasheet che prevede un TAD minimo di $0,7 \mu\text{s}$ (Fig. 8), inoltre, considerando il caso peggiore di TACQ prima calcolato di $7,45 \mu\text{s}$, per stare sicuri dobbiamo innanzitutto aumentare il valore del prescaler a 16 con la funzione ADC_FOSC_16 e poi il valore massimo del moltiplicatore di TAD a 20 con la funzione ADC_20_TAD; quindi si avrà:

$$TAD = (1/(\text{FOSC}:16)) = 0,8 \mu\text{s} \quad \text{da cui} \\ TACQ = TAD \times 20 = 0,8 \mu\text{s} \times 20 = 16 \mu\text{s}$$

Pertanto il TACQ sarà più del doppio del TACQ peggiore con ampio margine di sicurezza. Il tempo totale di conversione dei 10 bit dell'ADC sarà quindi:

$$T_{CNV} = 0,8 \mu\text{s} \times 12 = 9,6 \mu\text{s}$$

Mentre il tempo totale di acquisizione e conver-

sione sarà:

$$TADC = TACQ + T_{CNV} = 16 \mu\text{s} + 9,6 \mu\text{s} = 25,6 \mu\text{s}$$

quindi il segnale analogico presente in ingresso al canale ADC verrà acquisito e convertito ogni $25,6 \mu\text{s}$; il TADC è definito periodo di campionamento da cui la frequenza di campionamento $f_c = 1/TADC = 39 \text{ kHz}$. In questo progetto il modulo ADC deve campionare segnali analogici di bassissima frequenza, generati dalle eventuali variazioni dei livelli di tensione provenienti dai sensori di temperatura e umidità, quindi non ha importanza se la frequenza di campionamento f_c di 39 kHz risulta piuttosto bassa. Ricordiamo che un corretto campionamento di segnali analogici veloci prevede una f_c doppia rispetto alla frequenza del segnale sotto misura. Si può ottenere una più alta frequenza di clock dell'ADC agendo sul TACQ minimizzandolo, ovvero riducendo quanto più possibile l'impedenza della sorgente esterna applicata all'ingresso del canale ADC.

Nella Fig. 8 dal datasheet un esempio di un ciclo di conversione A/D con tempo di acquisizione TACQ impostato per una durata di 4 clock TAD.

REALIZZAZIONE PRATICA

Per la realizzazione dei circuiti stampati potete scaricare le tracce lato rame dal nostro sito www.elettronica.in.it. Vi consigliamo poi di utilizzare un buon saldatore o, meglio ancora, una stazione saldatrice con punta sottile.

Procedete con il montaggio dei componenti in ordine di altezza: prima i diodi e le resistenze, poi induttanze, circuiti integrati, zoccoli, condensatori (prima quelli ceramici e in poliestere e poi gli elettrolitici facendo attenzione alla polarità) transistor, morsettiere e connettori.

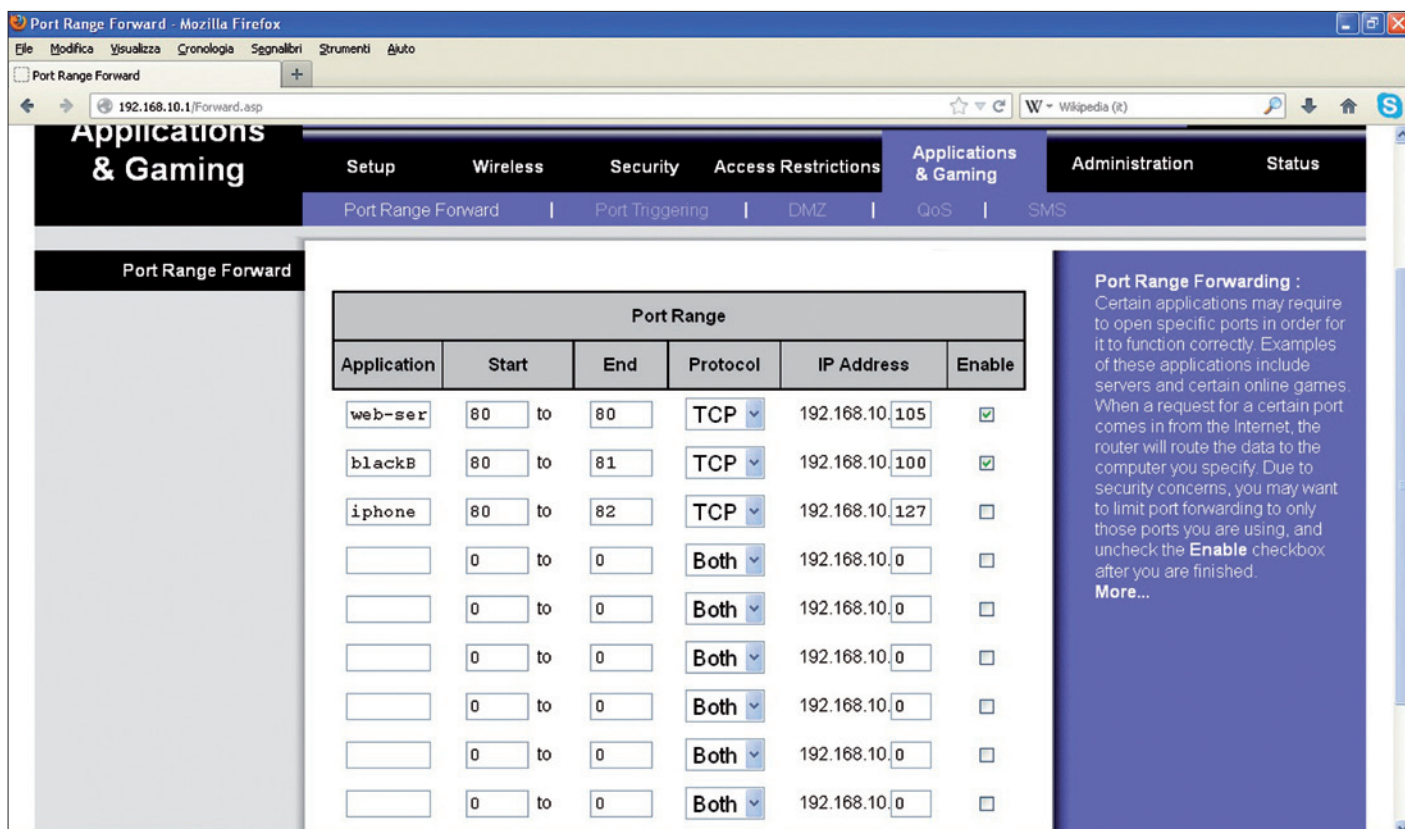


Fig. 9 - Setup Port Forwarding.

Completate il montaggio piazzando il modulo WIZ811MJ inserendo i pin nei connettori a 10 posti disposti su due file per lato. Realizzate la sonda di umidità saldando i due conduttori di uno spezzone di cavo UTP a due barrette metalliche di lunghezza di 5 cm circa, proteggendo le saldature con tubetto termorestringente. Terminato il montaggio dei componenti sul c.s. si passa al collegamento dei componenti esterni alle rispettive morsettiere: la clip portabatteria alla morsettiera CN2 e il cavo UTP relativo al sensore di umidità alla morsettiera CN1. Per il collegamento verso una centralina o qualsiasi altro dispositivo di attuazione è possibile utilizzare una delle linee I/O disponibili al connettore CN5, in questo progetto è attiva la linea PB0 il cui stato è impostato dalla selezione dei pulsanti WATER OFF (PB0=0V) e WATER ON (PB0=3V); alla PB0 si potrà collegare un driver e relativo relè considerando che le linee I/O da PB0 a PB5 possono erogare fino a 20mA.

COLLAUDO DEL WEB SERVER BOTANICO

Collegate l'alimentatore esterno a 12V, in grado di erogare una corrente continua di almeno 200mA, al connettore di ingresso SK1 della scheda e posizionate su ON l'interruttore SW1; si accenderanno i LED dei +3,3V e il LED del clock SPI del modulo WIZ811MJ.

Collegate il positivo del tester al pin centrale del trimmer RV1 e il negativo a massa, quindi regolate il trimmer fino ad ottenere precisamente una tensione continua di 1V. Collegate il cavo Ethernet al connettore RJ45 della scheda e ad una delle porte Ethernet del vostro router.

Ora, per far sì che il router instradi la connessione verso il server, nella sezione Port Forwarding occorre inserire l'IP del server che sceglierete nel range degli IP della rete del router, ad es. 192.168.10.105, selezionando anche il protocollo TCP/IP e la porta 80, spuntate infine il pulsante di abilitazione (**Fig. 9**). Fate riferimento alla documentazione del vostro router per scoprire come intervenire sulle impostazioni specifiche per il port forwarding; è probabile che le troviate cercando anche la voce "server", dato che è appunto nostra intenzione attivare un server all'interno della rete locale con indirizzi privati. Potrebbe anche essere necessario disabilitare il firewall interno al router per consentire al traffico entrante di raggiungere il nostro server botanico, oppure potete scegliere di realizzare una DMZ (DeMilitarized Zone) che è una "zona franca" della vostra rete privata, limitata a precisi indirizzi IP, dove non vengono applicate restrizioni nel traffico in entrata e in uscita.

A questo punto per il collaudo della scheda occorre assegnare gli altri parametri di rete al Web Server:

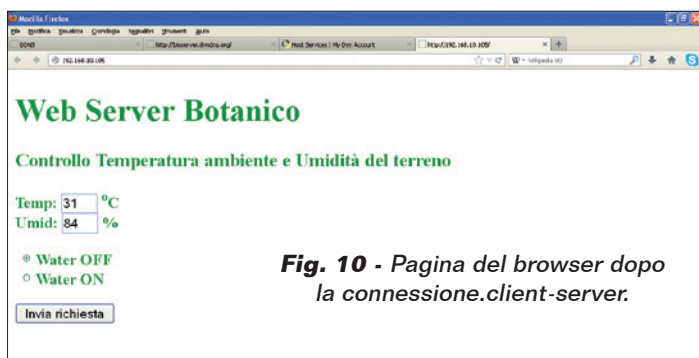


Fig. 10 - Pagina del browser dopo la connessione client-server.



Fig. 11 - Setup DNS.

MAC Address, IP gateway e subnet mask. Nel firmware del PIC forniti troverete già impostati tutti i parametri.

IP address: 192.168.10.105

MAC Address: 0x00,0x0E,0x35,0x73,0x9FD8,0xEC

Questi parametri potrete comunque modificarli, se necessario, entrando nel firmware con il tool della Microchip ICD2 o equivalenti, nel qual caso, oltre agli IP che assegnerete al server, nel firmware programmato nel PIC avete IP Gateway: 192.168.10.1 e Subnet mask: 255.255.255.0, mentre per reperire il MAC Address andate al prompt dei comandi del SO in uso e digitate "ipconfig/all"; nella schermata appariranno i parametri di configurazione dei dispositivi in rete, individuate e annotate il MAC Address del vostro PC e cambiate l'ultimo bit dell'ultimo byte, questo sarà il MAC Address del Web Server; a questo punto avrete tutti i parametri da inserire nel server.

Ora con tutti gli apparati accesi avviate nuovamente il prompt dei comandi e digitate "ping 192.168.10.105" o l'IP che avrete scelto e verificate che la scheda server sia raggiungibile in rete locale. Se anche questo test è andato a buon fine, potete iniziare il collaudo finale testando il reale funzio-

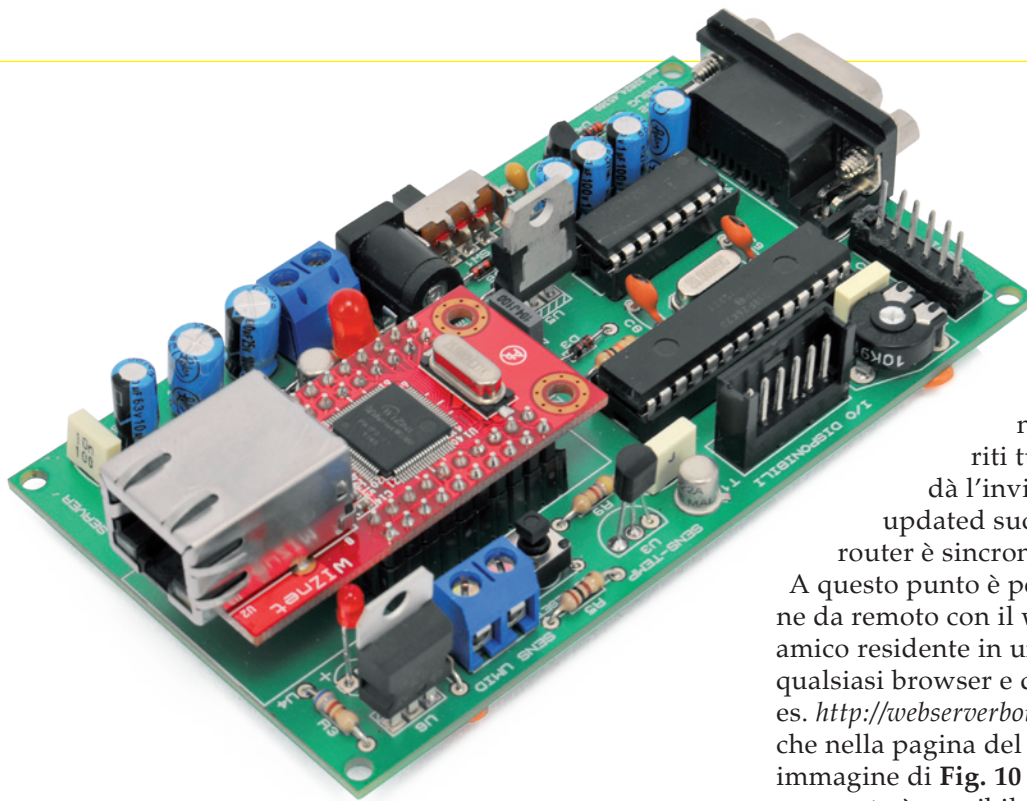
namento del server. Collegate il cavo seriale RS232 al PC e alla scheda server e aprite l'applicazione di terminale del vostro Sistema Operativo (Hyperterminal nel caso di Windows) configurando la porta seriale a 19200 bit/s, 8 bit, 1 bit stop; inserite in profondità le barrette metalliche della sonda del sensore di umidità nel terriccio di una pianta qualsiasi; dal vostro PC avviate il browser e digitate nella barra degli indirizzi l'indirizzo IP del server "http://192.168.10.105"; se tutto sta funzionando correttamente, dovrete vedere apparire nella pagina del browser l'immagine riportata nella Fig. 10 ove sono indicati in tempo reale i valori di temperatura ambiente, l'umidità assoluta del terriccio della pianta sotto monitoraggio (da 0% per terra asciutta fino al 100% per terra irrigata) e i pulsanti di accensione e spegnimento dell'eventuale centralina di irrigazione che vorrete utilizzare.

Al primo avvio dell'applicazione di default risulterà selezionato il pulsante "WATER OFF" e nella pagina dell'hyperterminal apparirà la scritta "CENTRALINA OFF"; ora selezionate "WATER ON" e inviate la richiesta, dopo pochi istanti vedrete il refresh della pagina del browser con il nuovo stato del pulsante selezionato "WATER ON" e nella pagina dell'hyperterminal apparirà la scritta "CENTRALINA ON". Ripetete più volte la selezione dei pulsanti modificando di volta in volta artificialmente lo stato dei sensori; dovrete veder variare i valori riportati nella pagina del browser a conferma della correttezza e stabilità dell'applicazione. Provatene a fare le stesse operazioni utilizzando il browser di un telefono cellulare BlackBerry, iPhone o di un tablet di cui avrete prima effettuato il collegamento con la vostra rete wireless; dovrete poter effettuare normalmente le stesse operazioni con successo com'è avvenuto via cavo con il PC.

Dai test effettuati sui prototipi si è rilevato il corretto funzionamento del server con i browser I.E., Mozilla Firefox, Chrome, Safari e Opera.

Sin qui si è visto come avviene la connessione di un client al server in rete locale, ma se volessimo monitorare ed eventualmente irrigare da remoto le nostre piante?

Per poterci collegare via web con il web server botanico da qualsiasi parte del mondo dobbiamo far sì che quando digitiamo un indirizzo web del tipo <http://webserverbotanico.dyndns.org> possiamo raggiungere il nostro server. Ovviamente non possiamo usare l'indirizzo 192.168.10.105 in quanto si tratta di un indirizzo IP privato, per cui avremo la necessità che il nostro provider di accesso a Internet ci fornisca un IP statico da associare al server che,



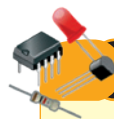
però, solitamente è previsto solo per contratti di tipo business. Nel caso di contratti di tipo privato il provider cambia dinamicamente l'IP della connessione Internet non consentendo la possibilità di realizzare una connessione diretta tramite IP con il server. La soluzione ci viene offerta da alcuni servizi quali dyndns.it, dyndns.org ecc... In pratica, previa registrazione e abbonamento o periodo di prova gratuito, è possibile che ci venga assegnato un "host name" - cioè una sorta di indirizzo mnemonico - che viene associato all'indirizzo dinamico assegnato dal provider di modo che, anche se ogni volta cambia l'IP di connessione a Internet, il sito si occupa di mantenere il collegamento con il nostro server aggiornando la corrispondenza fra host name e indirizzo IP. Occorre però che il router sia dotato della funzionalità gestione di questi servizi di DDNS e che faccia l'update dell'IP Internet assegnato ogni volta che questo cambia. Nei test sui prototipi si è utilizzato il router Linksys WRT54G che ha, appunto, la funzionalità DDNS. Dopo esserci procurati un host name dal sito dyndns.org, riprendiamo quindi il banco di test precedente, con gli apparati accesi e accendiamo questa volta anche il modem ADSL; entriamo nella pagina di setup del router (1) e nella sezione DDNS (2) selezioniamo il servizio *DynDns.org* (3).

Come mostrato nella **Fig. 11**, selezionato il servizio DynDns.org, occorre inserire user name e password già utilizzati nella registrazione del sito dyndns.org, poi l'host name che si è scelto sempre dal sito dyndns.org; sottostante l'host

name è indicato l'Internet IP dell'attuale connessione Internet stabilita quando avete collegato il modem che, essendo dinamico, può cambiare nel tempo. Una volta inseriti tutti i parametri richiesti si dà l'invio e il buon esito "DDNS is updated successfully" confermerà che il router è sincronizzato con il sito dyndns.org.

A questo punto è possibile testare la connessione da remoto con il web server; chiedete ad un amico residente in un'altra località di aprire un qualsiasi browser e digitare l'host name scelto, ad es. <http://webserverbotanico.dyndns.org> verificando che nella pagina del suo browser appaia la stessa immagine di **Fig. 10** a conferma che da quel momento è possibile raggiungere il nostro Web Server botanico da qualsiasi parte del mondo ci troviamo. In alternativa, potete usare il vostro smartphone disabilitando il WiFi e utilizzando la rete dati 3G/UMTS.

È da tenere comunque presente che per la garanzia di funzionamento ininterrotto del sistema occorre che modem ADSL, router e scheda del server siano alimentati mediante un gruppo di continuità; l'eventuale interruzione dell'alimentazione del modem o del router richiederà di risincronizzare il DNS del router con il sito che ha rilasciato l'host name, reinserendo i dati come descritto nella **Fig. 11**. ■



per il MATERIALE

Tutti i componenti necessari alla realizzazione di questo progetto sono facilmente reperibili in commercio. Dal sito della rivista (www.elettronica.in.it) è possibile scaricare il master del circuito stampato con cui realizzare il PCB, il firmware col quale programmare il microcontrollore PIC ed il software applicativo. Il modulo network con chip Wiznet W5100 (cod. 7100-WIZNET) può essere acquistato presso Futura Elettronica al prezzo di 35,00 Euro IVA compresa così come il microcontrollore PIC18F26K20 già programmato (cod. MF1085, Euro 14,00).

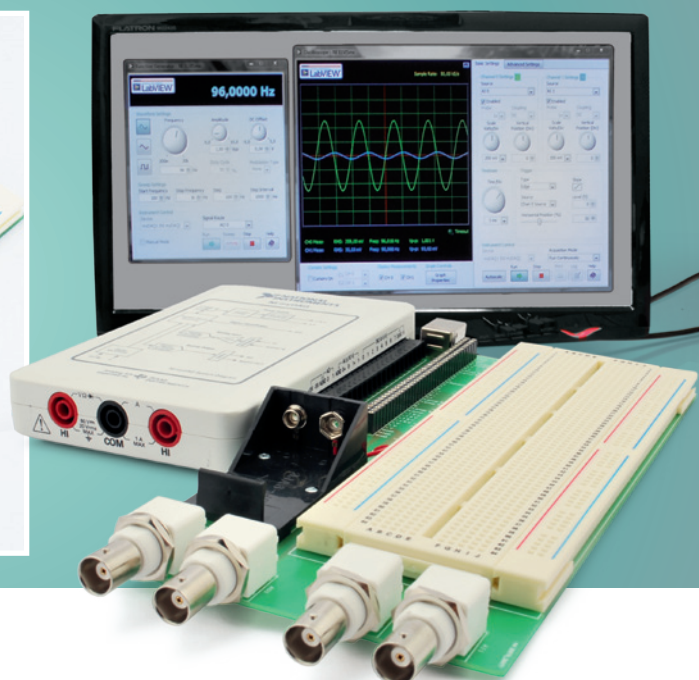
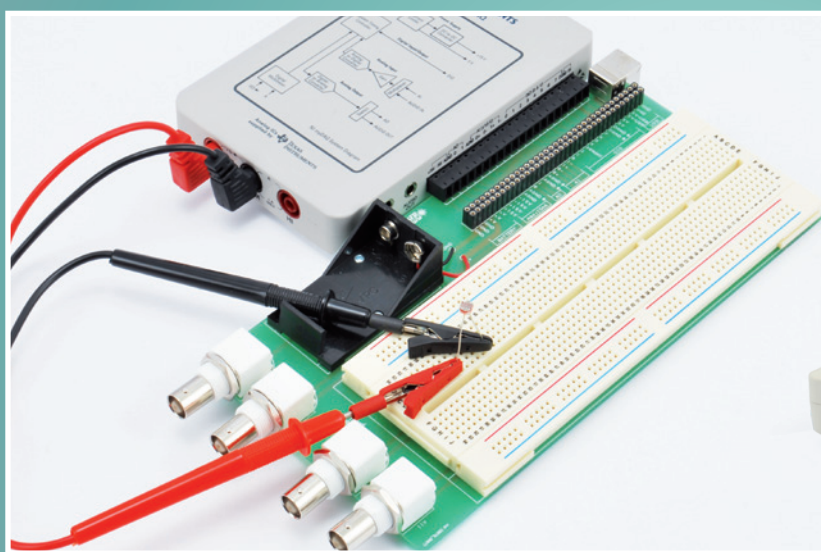
Il materiale va richiesto a:

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

myKIT

MISURA DI ILLUMINAMENTO CON FOTORESISTENZA

di DAVIDE SCULLINO



Questo articolo spiega come effettuare una misura di illuminamento utilizzando una fotoresistenza collegata alla scheda myDAQ. Il valore della resistenza del sensore sarà acquisita tramite DAQ Assistant che viene installato in LabVIEW con il driver NI DAQmx e sarà convertita nel valore di illuminamento corrispondente utilizzando la programmazione in LabVIEW. L'obiettivo è quello di effettuare una misurazione

di illuminamento utilizzando una fotoresistenza CdS GL5528 mediante i terminali DMM (digital multimeter) della scheda myDAQ con il software LabVIEW.

La fotoresistenza è un componente elettronico la cui resistenza è inversamente proporzionale all'intensità della luce che riceve. Le fotoresistenze sono elementi a semiconduttore in cui la radiazione luminosa assorbita viene convertita in una variazione di

Un altro esperimento con LabVIEW e myDAQ: questo mese realizziamo un software in grado di misurare il valore di una fotoresistenza in funzione del livello di illuminazione. Un esercizio che sarà molto utile per quanti, successivamente, dovranno realizzare applicazioni in questo campo.

Fig. 1 - Fotoresistenza.



Fig. 2 - Misure di resistenza effettuate con il Digital Multimeter fornito dal NI ELVISmx.



resistività del materiale stesso. Infatti la creazione di coppie elettrone-lacuna rende disponibili un maggior numero di portatori di carica che pertanto diminuiscono la resistività del materiale. In genere il materiale utilizzato è il CdS (solfuro di cadmio) oppure il PbS (solfuro di piombo), depositato in uno strato sottile su un supporto isolante. Il principale difetto delle fotoresistenze al CdS consiste nella scarsa velocità di ricombinazione delle coppie lacuna-elettrone generate dalla radiazione elettromagnetica, che se da un lato favorisce la loro sensibilità, dall'altro non consente una rapida risposta. Affinché il sensore abbia un'elevata sensibilità, deve essere in grado di catturare un numero elevato di fotoni e pertanto si tende a realizzare una superficie sufficientemente ampia. La fotoresistenza è utilizzata principalmente per la realizzazione di interruttori *crepuscolari* (circuiti che permettono di accendere una o più luci al calare del sole). Dalle specifiche fornite dal costruttore del sensore si ricava che la resistenza per un illuminamento di 0 lux ha un valore di almeno di 1 MΩ, mentre a 10 lux può avere un valore compreso tra 8 e 20 kΩ. Per comprendere meglio questi dati è utile introdurre la definizione di lux. Il lux (simbolo lx) è l'unità di misura dell'illuminamento adottata dal Sistema Internazionale. Un lux è pari a un lumen su metro quadrato; il lumen è l'unità di misura del flusso luminoso.

Di seguito sono elencati alcuni esempi di illuminamento espressi in lux:

- la luce del sole mediamente varia tra i 32.000 lx (32 klx) e i 100.000 lx (100 klx);
- sotto i riflettori degli studi televisivi si hanno circa 1.000 lux (1 klx);
- in un ufficio luminoso si hanno circa 400 lx;
- la luce della luna è pari a circa 1 lx.

Per avere un'idea più chiara sul funzionamento della fotoresistenza, può essere utile effettuare qualche misura di resistenza utilizzando il Digital Multimeter fornito da NI ELVISmx. In Fig. 2 sono riportate alcune

misure effettuate riferite a valori di illuminamento crescente. Come si nota, la resistenza è inversamente proporzionale all'illuminamento. Nel data-sheet del sensore si trova il grafico che esprime l'andamento della resistenza, espressa in kΩ, in funzione dell'illuminamento, espresso in lux (Fig. 3). 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:

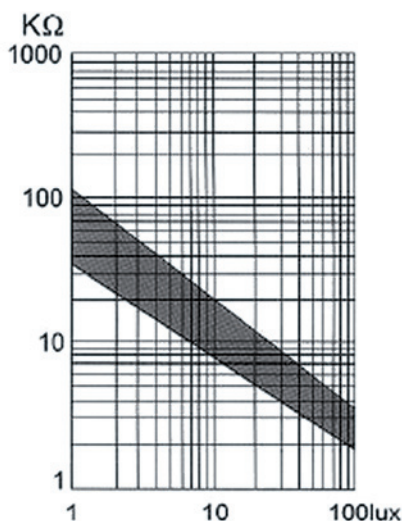


Fig. 3 - Grafico luminosità-resistenza fornito dal costruttore del sensore.

$$\log R = -a \times \log IL + \log R_1$$

In questa formula:

- R = resistenza della fotoresistenza;
- a = pendenza della retta chiamata

valore Gamma (valore fornito dal costruttore);

- IL = illuminamento espresso in lux;
- R₁ = resistenza della fotoresistenza in condizioni di illuminamento IL unitario (ricavabile dal grafico).

Da questa equazione è semplice ricavare il valore di illuminamento IL espresso in lux, in funzione degli altri dati. Si ottiene così:

$$\log \frac{R}{R_1} = \log IL^{-a} \rightarrow IL = \sqrt[a]{\frac{R}{R_1}}$$

Questa sarà la formula che utilizzeremo nel software per ricavare il valore di illuminamento IL.

ISTRUZIONI PER IL CABLAGGIO

La fotoresistenza è collegata al circuito come una

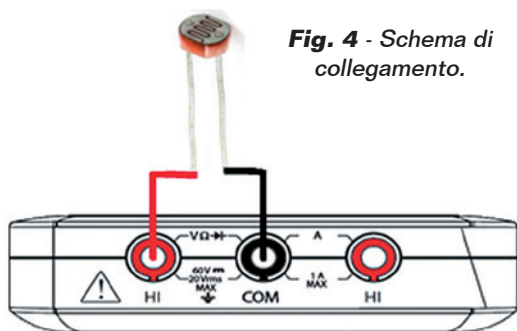


Fig. 4 - Schema di collegamento.

normale resistenza con un terminale fissato all'ingresso positivo (HI) ed uno all'ingresso negativo (COM) senza curarsi dell'orientamento.

PANNELLO FRONTALE LabVIEW

L'interfaccia utente che creeremo mostra i valori di illuminamento rilevati su un grafico e su due indicatori grafici, oltre a un indicatore numerico che riporta il valore di resistenza acquisito dal sensore. Il grafico dell'illuminamento memorizza i valori precedenti in modo che si possano vedere le variazioni di illuminamento nel tempo. Inoltre sono presenti due controlli numerici per inserire la resistenza R1 (resistenza della fotoresistenza in condizioni di illuminamento unitario) e il valore gamma (pendenza della "retta") fornito dal costruttore del componente. Questi dati permetteranno la risoluzione dell'equazione vista in precedenza, che consentirà di ricavare il valore di illuminamento in lux. Impostiamo tali valori tramite dei controlli invece che costanti, perché così il software si potrà utilizzare anche per fotoresistenze con specifiche diverse.

CODIFICA DELLA STRATEGIA

In LabVIEW abbiamo bisogno di acquisire il segnale proveniente dalla fotoresistenza che va da 0 a più di 1MΩ. Per avere una migliore risoluzione di misura, però, dobbiamo limitarci ad acquisire la resistenza da 0 a 20 kΩ. Secondo le specifiche del dispositivo myDAQ, questa scelta garantisce una risoluzione di 0,01 kΩ. Lo svantaggio sta nel fatto che non potremo acquisire valori di resistenza superiori a 20 kΩ, e quindi valori di illuminamento inferiori a 7 lux (valore calcolato con $R1 = 70 \text{ k}\Omega$ e valore gamma=0,7). Il valore di resistenza acquisito sarà poi convertito in un valore di illuminamento utilizzando l'equazione vista in precedenza, che richiede la conoscenza di alcune specifiche del sensore utilizzato.

$$IL = -a \sqrt{\frac{R}{R1}}$$

Infine, forniremo in uscita il risultato a due indicatori grafici e a un grafico sul Pannello Frontale.

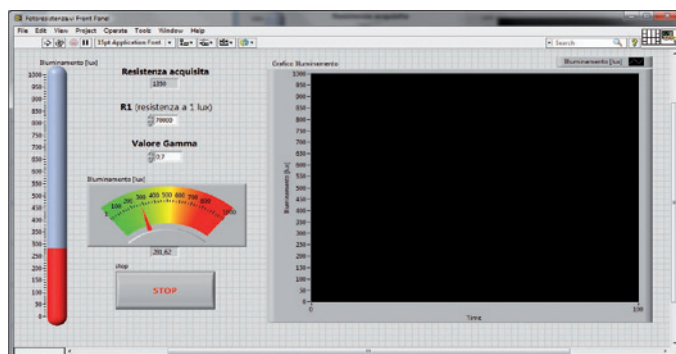


Fig. 5 - Pannello frontale LabVIEW.

Il diagramma a blocchi di LabVIEW è molto simile allo schema a blocchi di codifica.

COME FUNZIONA

All'interno del ciclo *While*, sulla sinistra è situato DAQ Assistant, configurato per acquisire un singolo valore



Fig. 6 - Schema a blocchi di codifica.

di resistenza dai terminali DMM myDAQ ogni volta che viene eseguito. Utilizzando la fotoresistenza, una volta che il valore di resistenza è stato acquisito, viene passato a un indicatore numerico e all'interno del *Formula Node*. Il valore viene poi convertito in un valore di illuminamento usando l'equazione vista in precedenza e utilizzando i parametri forniti dal costruttore del sensore utilizzato, inseriti tramite controlli numerici. Infine, la temperatura viene visualizzata sui due indicatori grafici e sul grafico del pannello frontale. Tutto il codice presente nel ciclo while continua a funzionare fino a che non viene premuto il pulsante Stop sul Pannello Frontale. Tenete presente che le misure effettuate

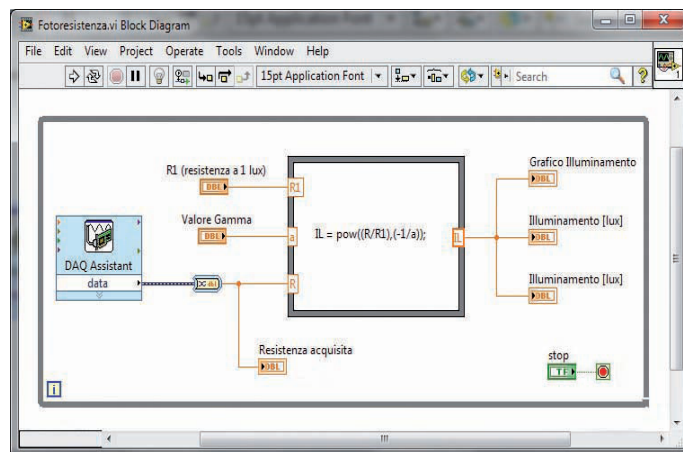


Fig. 7 - Diagramma a blocchi LabVIEW.

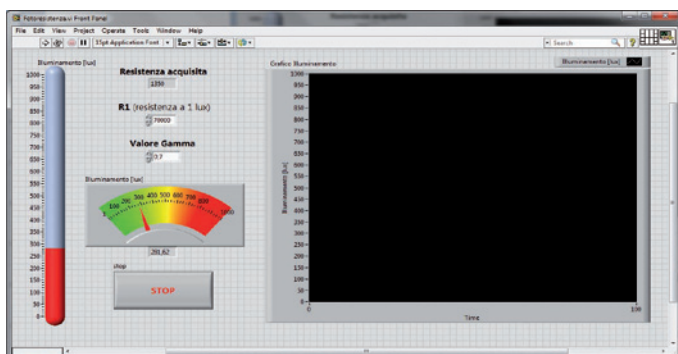


Fig. 8 - Pannello frontale LabVIEW completato.

tramite gli ingressi DMM sono temporizzate dal software, quindi la velocità di aggiornamento è influenzata dal carico sul computer e dall'attività sull'USB.

COSTRUZIONE DEL PANNELLO FRONTALE

Implementeremo ora il Pannello Frontale. Aprite un nuovo file LabVIEW e andate sul Pannello Frontale, quindi cliccate col pulsante destro del mouse per fare comparire la palette *Controls* dove sono elencati tutti i tipi di controllo e di indicatore, ma anche decorazioni, tabelle, ecc. Per comodità, potete bloccare la palette sullo schermo in modo da non doverla richiamare continuamente: allo scopo basta fare clic sulla "puntina" in alto a sinistra. Nella 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 in "Illuminamento [lux]" (**Fig. 8**). 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*. Ora, sempre alla voce *Modern*, selezionate e inserite due *Numeric Control* e un *Numeric Indicator* e, utilizzando gli strumenti *Positioning* e *Labeling*, posizionate e rinominateli come in **Fig. 8**. 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 un pulsante di Stop che trovate sempre sulla palette *Controls* ma alla voce *Modern»Boolean*, e un *Waveform Chart*, situato alla voce *Modern»Graph*. Disponeteli e ingranditeli a vostro piacimento con lo strumento *Positioning* e rinominateli con lo strumento *Labeling* come in **Fig. 8**. Utilizzando il medesimo strumento rinominate anche l'etichetta dell'asse Y e la

legenda in alto a destra del grafico in Illuminamento [lux], poi deselectionate la voce *Autoscale Y* dal menu a tendina che comparirà premendo il 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 i valori 0, +1000 con lo strumento *Labeling* (se durante le misure che effettuate il valore di illuminamento supera i 1000 lux inserite una soglia più alta così potrete visualizzarlo). Per finire inserite i seguenti valori sui controlli numerici *R1* e *Valore Gamma*: *R1*=70000, *Valore Gamma*=0,7. Il valore gamma è fornito dal costruttore del sensore, mentre il valore di *R1*(resistenza a 1 lux) potete ricavarlo dal grafico in **Fig. 3**.

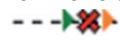
Successivamente, dal menu edit, impartite il comando *Make Current Values Default*: in questo modo impostate come predefiniti tali valori ogni qualvolta aprirete e utilizzerete questo VI (ovviamente se in futuro utilizzerete fotoresistenze diverse con parametri *Valore Gamma* e *R1* diversi, inserite i valori corretti e ripetete questa operazione).

COSTRUZIONE DEL DIAGRAMMA A BLOCCHI

Ora ci occuperemo 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 dovrete già visualizzare i controlli numerici e il pulsante Stop con le stesse etichette che avevate messo sul Pannello Frontale. Inoltre dovrete visualizzare i terminali, ad esempio quello del pulsante Stop, con questa icona: . È preferibile però visualizzarlo così: . Questo riduce lo spazio che occupa sul diagramma a blocchi. Per fare ciò selezionate *Tools»Options* e, dalla finestra che si aprirà, selezionate sulla sinistra *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 impostato come controllo. Prestate attenzione ai collegamenti che effettuerete sul diagramma a blocchi: essi permettono di trasferire dati tra oggetti del diagramma a blocchi; ogni collegamento parte da una sorgente di dati e la collega ad uno o più VI e funzioni che leggono i dati stessi. I collegamenti hanno differenti colori, stili e spessori, in funzione del tipo di dati. Un collegamento spezzato  appare come una linea nera tratteggiata con una X rossa in mezzo.

I collegamenti possono rompersi per svariati motivi: ad esempio se si tenta di collegare due oggetti con tipi di dati tra loro incompatibili.

La **Tabella 1** illustra i tipi di collegamento più comuni. I collegamenti possono essere realizzati solamente tra ingressi e uscite che veicolino dati compatibili tra loro. Nel diagramma a blocchi, facendo clic con il pulsante destro del mouse compare la palette *Functions*, che contiene i VI, le funzioni e le costanti da usare per creare il diagramma a blocchi. Se volete bloccare la palette *Functions* sullo schermo, fate clic sulla puntina che si trova in alto a sinistra.

Ora potete iniziare la costruzione del diagramma a blocchi. Per prima cosa dovete inserire il blocco DAQ Assistant per acquisire i valori di resistenza del componente. Le seguenti istruzioni seguono passo-passo la configurazione del DAQ Assistant.

1. Assicuratevi che il myDAQ sia collegato al PC;
2. Premete Ctrl-Spazio per aprire la finestra *Quick Drop* (ciò, al primo utilizzo, richiederà un minuto);
3. Cercate la voce DAQ Assistant e fate doppio clic su di essa quando appare nella lista;
4. Trascinate sul diagramma a blocchi;
5. Quando appare un nuovo riquadro per le attività di configurazione rapida, selezionate:
 - a. Acquire Signals;
 - b. Analog Input;
 - c. Resistance;
 - d. Dev 1 (NI myDAQ). Nota: Se si dispone di altri hardware NI installati, myDAQ non sarà Dev1;
 - e. dmm;
 - f. Finish;
6. Inserite in *Resistance Setup* il corretto *Signal Input Range*:
 - a. Max: 20k;
 - b. Min: 0;
7. Configurate l'eccitazione:

Tabella 1

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

- a. Internal;
- b. 1mA;

8. Cambiate il *Timing Settings*:
 - a. 1 Sample (On Demand);

9. Fate clic su OK.

Una volta terminata la configurazione, andate sulla palette *Functions*, selezionate *Programming»Structures* e poi *Formula Node*, quindi disponete il *Formula Node* nel diagramma a blocchi. Con lo strumento *Positioning* ingranditelo un po' e, cliccando con il pulsante destro del mouse sul bordo sinistro in alto, selezionate *Add Input*. Comparirà un riquadrino sul bordo del *Formula Node*; usando lo strumento *Labeling* nominatelo con R1. Ora, con lo strumento *Wiring* collegate il controllo numerico R1(resistenza a 1 lux) all'ingresso del *Formula Node* appena creato. Ripetete quest'operazione altre due volte rinominando i riquadri in "a" ed "R" per poter collegare al *Formula Node* i restanti ingressi *Valore Gamma* e la resistenza acquisita (**Fig. 10**), facendo attenzione che per collegare l'uscita data del DAQ Assistant all'ingresso "R" bisogna inserire il blocco *Convert from Dynamic Data* situato sulla palette *Functions* alla voce *Express»Signal Manipulation*. Questo permette all'espressione che inserirete nel *Formula Node* di avere un riferimento ad ogni ingresso utilizzato. Appena

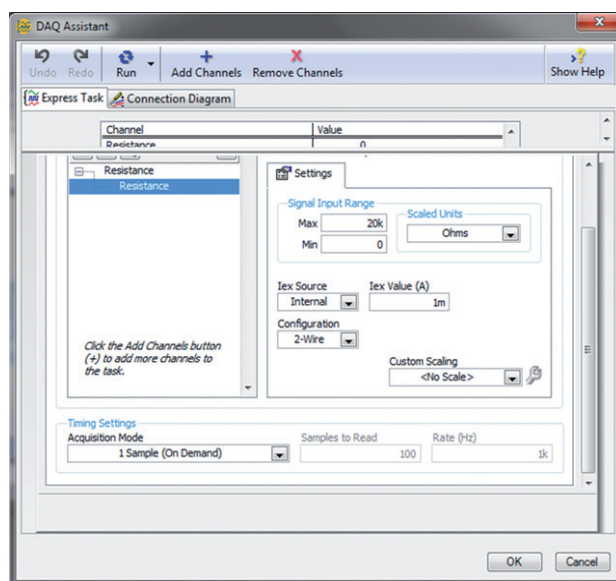


Fig. 9 - Configurazione DAQ Assistant

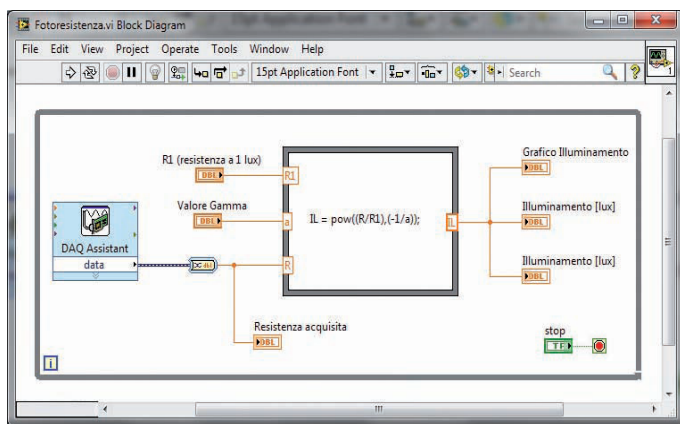


Fig. 10 - Diagramma a Blocchi LabVIEW.

inserirte 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 andate sul bordo destro del *Formula Node*, cliccate col tasto destro del mouse, selezionate *Add Output* e nominate come "IL" l'uscita così creata. Usando lo strumento *Wiring* collegate questa uscita ai due indicatori grafici *Illuminamento [lux]* e all'indicatore *Grafico Illuminamento* (Fig. 10). Per inserire l'espressione nel *Formula Node* utilizzate lo strumento *Labeling* e scrivete:

$$IL = \text{pow}((R/R1), (-1/a));$$

Se volete visualizzare l'espressione nel centro del *Formula Node*, premete più volte *Invio* così da andare a capo di una nuova riga. Come vedete, nell'espressione sono presenti diversi termini che avete già inserito come ingressi sul bordo sinistro del *Formula Node*. Inoltre il risultato, che è il valore di illuminamento, viene passato come valore in uscita all'output IL presente sul bordo destro del *Formula Node*. Per avere anche un'indicazione sul valore della fotoreistenza, collegate l'indicatore *Resistenza acquisita* all'uscita *Data di DAQ Assistant* dopo che è stata convertita dal blocco *Convert from Dynamic Data* dal tipo di dato "dinamico" al valore di tipo "double" (Fig. 10). 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*.

Quando avrete finito di costruire il vostro VI andate su *File»Save as* e scegliete un nome e una destinazione con cui salvarlo. Ora cliccate sul pulsante *Run* nella barra degli strumenti per mettere il VI in esecuzione. Mentre il VI è in esecuzione, l'icona del pulsante *Run* si trasforma nella seguente: . Una volta terminata l'esecuzione, il pulsante *Run* torna al suo stato originale e gli indicatori del Pannello Frontale mantengono i

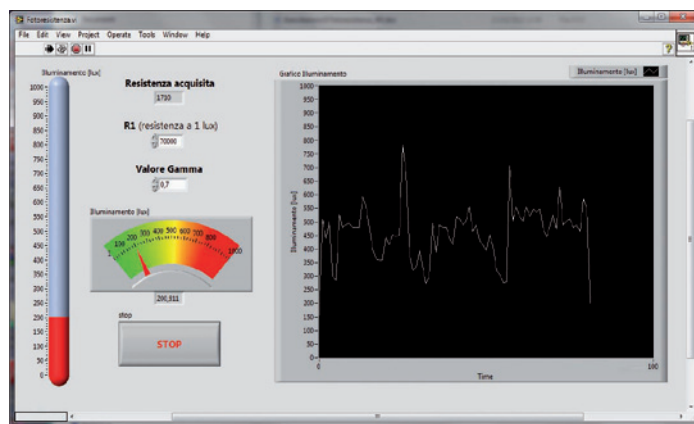
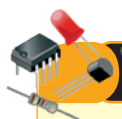


Fig. 11 - VI in esecuzione.

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.



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).

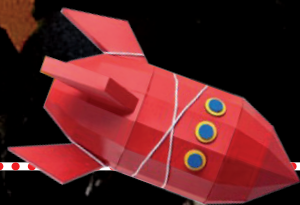
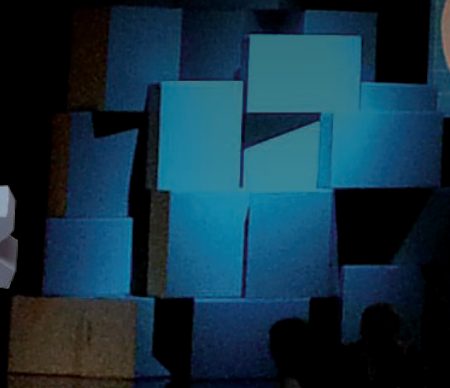
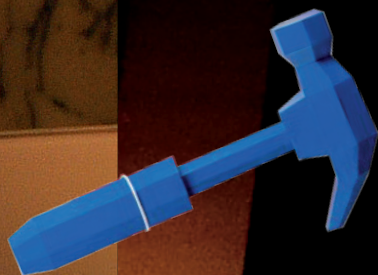
Il materiale va richiesto a:
Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
www.futurashop.it



ROME

Maker Faire®

UN SUCCESSO ANNUNCIATO



di ARSENIO SPADONI

Creatività, innovazione e tante, tantissime persone hanno animato la prima edizione europea della Maker Faire che si è tenuta a Roma lo scorso ottobre. Una ventata d'aria fresca ha investito la severa struttura del Palazzo dei Congressi dell'EUR che si è riempita fino all'inverosimile di visitatori e di maker provenienti da tutta Europa.

Sicuramente in molti avrete già letto il comunicato finale della manifestazione con tutti i numeri del caso: quasi 35.000 presenze nei quattro giorni, 1.500 persone che hanno seguito la Conferenza di giovedì, 6.000 studenti che hanno riempito i padiglioni del Palazzo dei Congressi dell'EUR il venerdì e oltre 25.000 visitatori il sabato e la domenica.

Al di là dei numeri, che hanno in ogni caso il loro peso, il successo della manifestazione, fortemente voluta da Massimo Banzi e





Riccardo Luna, sta principalmente nell'altissima qualità della proposta messa in campo, che per molti versi ha superato l'originale americano. Non a caso abbiamo visto Dale Dougherty – l'ideatore delle Maker Faire e fondatore della rivista Make – seguire attentamente tutti gli speech e aggirarsi tra gli stand dei makers alla ricerca di nuove idee e nuovi spunti.

Quanti hanno partecipato a questa prima edizione europea della Maker Faire hanno sicuramente avuto modo di apprezzare la Conference del primo giorno: un susseguirsi di presentazioni, filmati e demo particolarmente coinvolgenti, all'insegna dell'*Inspiring the Future*.

Come si fa, infatti, dopo aver visto un ragazzino di 15 anni riuscire nell'intento di preparare un test per il tumore al pancreas a non pensare di poter/dover fare qualcosa di simile? Oppure come non appassionarsi al racconto di Alice Taylor che insegna a

bambini e ragazzi a costruire con le nuove tecnologie il giocattolo dei propri sogni, unico e assolutamente diverso da qualsiasi altro?

E che dire di chi, come Nina Tandon, sta realizzando parti di ricambio per il corpo umano integrando un bioreattore con tecnologie mutate dalla stampa 3D?

Inspiring the Future, appunto, con un filo conduttore unico: migliorare il nostro mondo, a piccoli passi, come suggerito dallo scrittore David Gauntlett, condividendo le conoscenze e le esperienze. Un esempio che parte dal basso e che a poco a poco arriverà fino a quanti hanno in mano le sorti del nostro mondo. Almeno questa è la speranza.

Chi ha seguito fino in fondo la Conference ha avuto la netta sensazione che già con l'attuale tecnologia sarebbe possibile cambiare radicalmente la nostra società sconfiggendo l'ignoranza, la disoccupazione e la povertà, all'insegna del

How to reMake the World, come recita, appunto, il titolo della Conference; e di esempi ne abbiamo visti tantissimi.

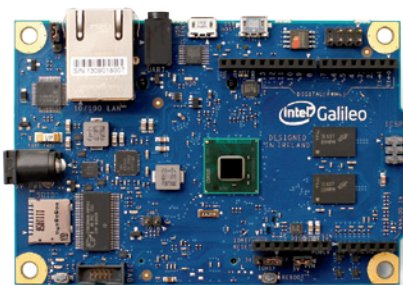
Se da un lato Riccardo Luna ha condotto con grande partecipazione la Conferenza, il vero mattatore della prima giornata è stato



Board Embedded compatibile Arduino con sistema operativo Linux o Android. Dual e Quadri-Core. Completamente open-source nell'hardware e nel software.
www.udoo.com

**UD00,
board embedded
sponsorizzata
da Kickstarter**

I nuovi prodotti ARDUINO



Durante l'evento romano, il team di Arduino ha annunciato la nuova board Arduino TRE che sarà disponibile dalla primavera prossima e che è in fase di preparazione in collaborazione con la Fondazione BeagleBoard e con Texas Instruments. È stata anche presentata (e distribuita gratuitamente ai presenti) la nuova scheda Intel Galileo (vedi l'anteprima su questo stesso numero). È stato lo stesso CEO di Intel Brian Krzanich, insieme a Massimo Banzi, a presentare la board e ad annunciare l'accordo di collaborazione con Arduino.



Massimo Banzi che a sorpresa ha presentato la board Galileo, un progetto destinato sia al mondo dell'educational che a quello dei makers, frutto della collaborazione con Intel, una partnership siglata appena due mesi prima. In questo brevissimo lasso di tempo Intel e Arduino sono riusciti a realizzare una scheda embedded che combina il mondo di Arduino con la potenza di un sistema operativo basato su GNU/Linux. Certo, il processore scelto utilizza un'architettura CISC con consumi tre volte superiori rispetto ai processori ARM, come hanno subito osservato in molti; dal canto loro, i diretti interessati fanno notare come questa sia solamente la prima di una serie di board che nasceranno dalla collaborazione tra Arduino e Intel. Per sottolineare l'importanza di questo accordo, alla presentazione della scheda Galileo è intervenuto Brian Krzanich, CEO di Intel, società che ha anche sponsorizzato la Rome Maker Faire e che era presente a Roma con uno stand molto grande.

Un migliaio delle nuove board Galileo sono state distribuite gratuitamente ai presenti ed altre 50.000 verranno donate a università di tutto il mondo. La massiccia partecipazione di un colosso come Intel e la presenza addirittura di Brian Krzanich hanno fatto pensare a qualcuno che Massimo Banzi potesse seguire le orme di Bre Pettis, il fondatore di MakerBot Industries recentemente passato armi e bagagli alla multinazionale Stratasys per 600 milioni di dollari.

"Non scherziamo" mi dice Massimo "Arduino è una realtà molto più complessa e articolata



Dopo aver perso la mano in un incidente, Nicolas ha costruito da solo la sua mano bionica stampata in 3D e controllata da Arduino.

<http://www.youtube.com/watch?v=JpOVeNjI4ks>



Bionico project, Nicolas Huchet e la sua mano bionica

con attività in molti paesi del mondo. La cosa più importante dell'accordo con Intel (e di cui siamo particolarmente orgogliosi) è di essere riusciti a fare produrre ad una delle più importanti società del mondo una scheda completamente open source sia nell'hardware che nel software, con una documentazione completa e che sarà di grande aiuto per migliaia di studenti nell'apprendimento dell'elettronica, dell'informatica e della robotica e che, oltretutto, verrà regalata in 50.000 esemplari". "Approfitto dell'occasione per ri-



**Comanda
l'erogazione del
caffè tramite uno
smartphone che
pubblica anche un
messaggio sulla tua
bacheca Facebook.
www.openpicus.com**

**Open Picus,
lo smartphone
che fa il caffè**



Intel Galileo è basata sul SoC Intel Quark X1000, il primo componente della famiglia di prodotti small-core e a basso consumo con tecnologia Intel Quark.

La tecnologia Quark estenderà l'architettura Intel ad aree in rapida crescita, dall'Internet delle cose ai computer indossabili del futuro. Progettato in Irlanda, il core di processore alla base del SoC Quark X1000 è una CPU Pentium single-core, single-thread a 32 bit compatibile con ISA (Instruction

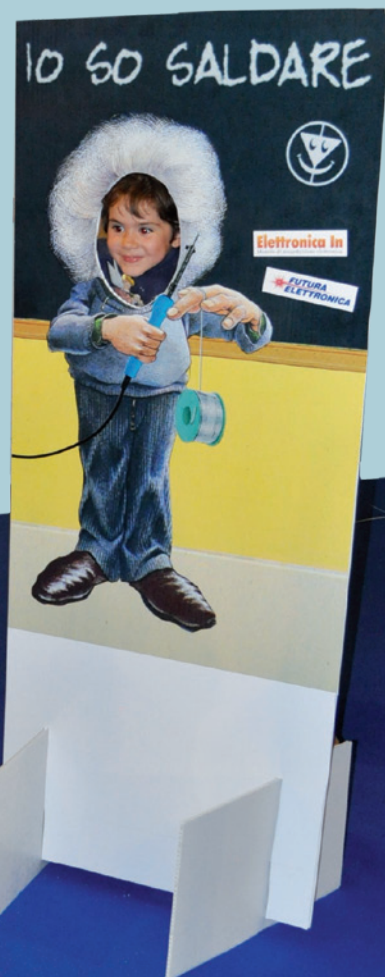
Set Architecture) che opera con velocità fino a 400 MHz.

Per quanto riguarda la disponibilità, Galileo sarà in vendita dalla fine di novembre ad un prezzo che dovrebbe essere compreso tra i 40 e i 50 dollari.

Sempre a proposito di quotazioni, qualcosa dovrebbe cambiare nel prezzo di vendita della Yún, "L'attuale prezzo è scaturito da un primo lotto di produzione piuttosto piccolo, adesso con numeri decisamente più consistenti contiamo

spondere a quanti in questi giorni ci hanno criticato per il fatto che Galileo e le altre nuove board (Yún e Arduino TRE) vengono prodotte negli Stati Uniti o a Taiwan." prosegue Massimo "La verità è che il nostro progetto si è caratterizzato sin dall'inizio per avere una valenza internazionale: io – pur essendo italiano – vivo e insegno da anni in Svizzera, David (Cuartielles) è spagnolo ma vive a Malmö, David Mellis lavora e insegna negli Stati Uniti... collaboriamo con tantissime società sparse in mezzo mondo per cui è normale che una board venga prodotta in Italia, l'altra negli Stati Uniti e così via." Ovviamente anche noi durante la Maker Faire di Roma siamo riusciti a mettere le mani su un paio di board Galileo e su questo stesso numero trovate le prime impressioni.

Intel Galileo combina le prestazioni della tecnologia Intel e la facilità dell'ambiente di sviluppo software Arduino. La scheda di sviluppo funziona col sistema operativo open source GNU/Linux e con le librerie software Arduino, consentendo così scalabilità e riutilizzo degli sketch; la scheda è inoltre compatibile con l'hardware e il software dell'ecosistema degli shield Arduino.



Makers, a story: from Leonardo to Arduino

2005

Make (or MAKE) is an American quarterly magazine published by Maker Media which focuses on do it yourself (DIY) and/or SHED (Do It With Others) projects involving computers, electronics, robotics, metalworking, woodworking and other disciplines. The magazine is marketed to people who enjoy making things and features complex projects which can often be completed with cheap materials, including household items. Make magazine is considered "a central organ of the maker movement lead by Dale Dougherty".

"We don't just live. We make".

2008

At the Maker Faire the Institute for the Future, a 40-year-old non-profit Stanford think-tank, presents an historical map about the future of Making, exploring how the stuff of our world may be researched, invented, designed, manufactured, and distributed in the next ten years.

"There is much to be learned from the maker mindset of collaborative creativity, and open access. Yet maker culture will not replace industrial industry. In the future, traditional manufacturers and makers will be closely linked, sometimes cooperating, sometimes competing, but frequently crossing the boundaries that separate

2009

Makers is a novel by Cory Doctorow. The book focuses on a near-future imagining of members of the maker culture. Doctorow imagines a world composed of "makers", basic living arrangements, even when they are down the town

2008

Arduino is a low-cost, open source, easy to use and easy to program single-board microcontroller to make using electronics in multidisciplinary projects more accessible. It was started as a project for students in at the Design Institute of Ivrea: the founders are Massimo Banzi, David Cuatrecasas, Tom Igoe, Gianluca Martino, David Mellis and Nicholas Zambetti. The official presentation on the blog is April 23, 2008: "It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments".

"Now you don't need anybody's permission to create something great".

HOW TO Make Revolution



di riuscire ad abbassare un po' il prezzo di vendita al pubblico." mi conferma Massimo. Oltre alla presentazione della scheda Galileo, la Maker Faire è stata l'occasione per l'annuncio della nuova board Arduino TRE che sarà disponibile nella primavera del 2014 e che è stata realizzata in collaborazione con la fondazione BeagleBo-

ard e con Texas Instruments. Anche questa scheda combina una sezione di interfacciamento col mondo fisico basata su Arduino con una sezione di elaborazione e connessione che utilizza un processore Sitara della Texas Instruments sul quale gira GNU/Linux. Al momento tutto quello che si sa di questa

scheda è quanto contenuto nello scarno comunicato ufficiale, anche perché la board è ancora in fase di sviluppo.

In nessun'altra Maker Faire del passato c'erano stati così tanti, importanti, annunci come qui a Roma, neppure a New York o a San Mateo, le due edizioni americane più importanti. Insomma, un vero trionfo per Massimo Banzi, da tutti riconosciuto come il grande protagonista di questa manifestazione. Ma il vero esame per la Maker Faire romana doveva ancora arrivare: quello del pubblico. Un'attesa piena di tensione, anche perché lo sforzo finanziario per organizzare l'evento era stato piuttosto consi-



Il nostro STAND

La Maker Faire è stata l'occasione per vedere in funzione alcuni nostri progetti di punta tra i quali la stampante 3Drag che ha "sforato" in continuazione oggetti di ogni tipo e il nostro veicolo OpenWheels, che i visitatori hanno potuto ammirare muoversi tra i padiglioni della fiera.

Molto interesse hanno anche suscitato i numerosi altri prototipi esposti, specie quelli legati al mondo Arduino e Raspberry Pi, nonché i libri ed i set dedicati a questi prodotti.



stente ed un insuccesso avrebbe significato stroncare sul nascere questa iniziativa. Personalmente ricordo la grande apprensione dei responsabili di Asset-Camera incontrati prima dell'estate, preoccupati di non riuscire a portare che poche migliaia di makers all'evento, in netto contrasto col mio ottimismo, basato sulla conoscenza della storia delle Maker Faire americane e sulla qualità della manifestazione che si stava organizzando. La mia preoccupazione, caso mai, era quella di capire se la struttura avrebbe retto l'impatto della folla che – ne ero certo – sarebbe arrivata a Roma. In effetti le mie previsioni si sono rivelate esatte, soprattutto il sabato, quando è addirittura dovuta intervenire la polizia per disciplinare il flusso di visitatori. In ogni caso, dopo la Conference di giovedì, una ventata di ottimismo l'aveva portata la giornata di venerdì – dedicata alle scuole – con l'invasione di quasi 6.000 studenti, i primi a vedere i progetti degli oltre 200 makers provenienti da tutta Europa che nella giornata di giovedì avevano allestito i pro-

pri stand. Gli studenti sono stati anche i primi ad animare i vari laboratori e workshop.

IL WEEK-END

Nonostante il forte temporale di sabato mattina, già dopo un paio d'ore dall'apertura era praticamente impossibile girare nei corridoi e all'interno degli spazi espositivi: tutti volevano vedere, toccare e capire come funzionavano i progetti esposti. Tanti giovani e tantissime famiglie con bambini, ma anche molti ragazzi provenienti da altri paesi europei. Un successo anche da questo punto di vista, una vera Eu-

ropean Maker Faire e non una manifestazione solamente italiana. E con l'inglese, naturalmente, come lingua ufficiale, dal momento che oltre la metà degli espositori proveniva dall'estero. Una dimostrazione sul campo, al di là di tanti discorsi retorici, della necessità di conoscere almeno l'inglese per essere partecipi della nuova era digitale. Anche l'Arduino Store (qualcosa di simile all'americano Maker Shed) – lo spazio dove era possibile acquistare i prodotti – è stato preso d'assalto, con lunghe code per riuscire a conquistare la cassa.

Per non parlare del nostro stand, letteralmente invaso per tutto il week-end da appassionati, curiosi

Unisce la tradizione del ricamo fatto a mano con la tecnologia digitale di Arduino per realizzare un completo sintetizzatore sonoro.

<http://afroditiptarra.com/index.php?on-going/lily/>

**Lilytronica,
sintetizzatore
con LilyPad
Arduino**



e anche da altre makers, intenti a valutare i prototipi esposti, in modo particolare le stampanti 3D che hanno “sforato” decine e decine di oggetti lavorando dalla mattina alla sera. Ma il progetto che ha riscosso maggior interesse è stato il nostro veicolo simil-Segway OpenWheels, che - condotto da Boris e Alessandro e da qualche visitatore coraggioso - si è mosso agilmente tra i padiglioni della fiera.

Alla fine, un successo su tutti i fronti che ha premiato quanti hanno creduto in questo progetto anche se, paradossalmente, il grande successo di pubblico e il numero fin troppo elevato di eventi hanno impedito di assaporare fino in fondo questa grande iniziativa. Nonostante i quattro giorni trascorsi a Roma non siamo riusciti a contattare molti dei makers presenti e a partecipare a tutti gli eventi in programma. Il Palazzo dei Congressi dell'EUR ha mostrato i suoi limiti, non certo per una carenza organizzativa ma per il grandissimo numero di visitatori.

Non è dello stesso parere Massimo Banzi “In effetti le nostre più ottimistiche previsioni prevedevano la presenza di 20.000 visitatori, un numero compatibile con la struttura, mentre quelli che realmente hanno varcato i cancelli nel weekend ci hanno un po' spiazzati. Capisco che ci sia stato qualche problema, anche i bar, mi hanno detto, sono rimasti a corto di panini e bevande. L'anno prossimo cercheremo di sfruttare meglio gli spazi a disposizione e di ottimizzare l'attuale struttura che offre ancora qualche possibilità in questo senso. Anche perché non vogliamo fermarci qui e il numero di presenze deve asso-

lutamente aumentare. Alla prima edizione della Maker Faire di New York sono arrivati, come qui a Roma, 35.000 visitatori mentre alla seconda edizione erano già 50.000: noi non vogliamo essere da meno. Ma c'è anche un altro motivo per il quale vogliamo restare a Roma: non intendiamo cambiare nulla perché con Asset-Camera abbiamo lavorato bene e loro hanno creduto fino in fondo a questo progetto facendosi carico non solo di un



grande rischio finanziario ma mettendo in gioco più di una poltrona che sarebbe sicuramente saltata in caso di flop. Non pensiamo ad altre soluzioni per i prossimi anni”. Un'altra critica che qualcuno ha mosso alla Rome Maker Faire è stata quella di essere troppo Arduino oriented. In molti hanno evidenziato l'assenza di alcuni importanti protagonisti di questa nuova ondata di innovatori, prima fra tutti la fondazione Raspberry Pi con la relativa board che ad oggi è stata venduta in circa due

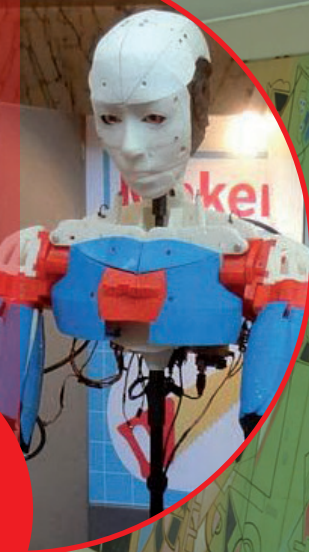
milioni di pezzi... "Gli organizzatori non hanno posto veti né impedito a qualcuno di partecipare" risponde Banzi, "se la Fondazione Raspberry Pi voleva essere presente tra i makers poteva rispondere alla Call e sicuramente sarebbe stata accettata; se invece voleva partecipare come Sponsor poteva seguire la trafilata di tutti gli altri, contribuendo finanziariamente o fornendo in cambio qualcosa di significativo ed attinente all'evento. Probabilmente anche loro, come molte altre importanti società, non hanno creduto in questo progetto e si sono defilati."

Chiediamo a Banzi, a manifestazione terminata, quale è stato l'aspetto della prima edizione della Maker Faire romana che gli ha fatto più piacere.

"Sicuramente la presenza di tantissime famiglie con i bambini che trascinavano i genitori da uno stand all'altro, in perfetta sintonia con quello che è lo spirito delle Maker Faire. D'altra parte tutto quello che stiamo facendo oggi lo facciamo proprio per loro."

**Il primo
umanoide
in dimensioni reali
stampato in 3D
completamente
open-source tende la
mano ai visitatori
della Maker Faire.
www.inmoov.fr/**

**InMoov,
robot umanoide
in dimensioni
reali stampato
in 3D**



**Struttura
semplicissima
che consente
di scrivere
o disegnare
su qualsiasi
muro.
[www.
carloratti.
com/project/
openwall/](http://www.carloratti.com/project/openwall/)**

**Vertical Plotter,
per scrivere
e disegnare
sui muri**

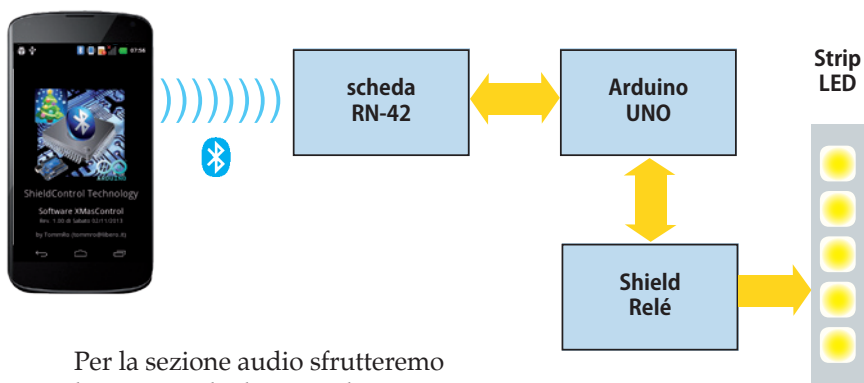
MUSICXMAS: RITMO E COLORE PER IL TUO NATALE

Ingg. TOMMASO E ALESSANDRO GIUSTO

Tutti gli anni a Natale cerchiamo qualche novità per l'illuminazione dei nostri addobbi, in modo da lasciare a bocca aperta i parenti e gli amici che vengono a trovarci e a farci gli auguri. Quest'anno vi proponiamo qualcosa di un po' insolito: si tratta non di un classico gioco di luci o di un carillon natalizio, ma di un sistema che gestisce l'illuminazione a seconda del ritmo della vostra canzone preferita.

Realizziamo un'animazione luminosa che fa pulsare 8 luci al ritmo del brano musicale riprodotto da uno smartphone Android collegato via Bluetooth.





Per la sezione audio sfrutteremo la capacità degli smartphone Android di riprodurre ed analizzare in tempo reale un file audio (sono supportati i formati MP3 e WAV); per la sezione hardware utilizzeremo, invece, un sistema Arduino munito di opportuni shield di espansione. La comunicazione fra le due unità avverrà tramite Bluetooth.

Il funzionamento del sistema nel suo complesso è questo: mediante un'App Android si sceglie e si riproduce un file audio; la stessa applicazione analizza, in tempo reale, la composizione dello spettro del segnale BF corrispondente (tra 0 e circa 20.000 Hz) e, a seconda che il livello rilevato per alcune bande di frequenza sia superiore o inferiore ad una soglia configurabile, attiva o disattiva le luci che compongono il nostro sistema di illuminazione.

Lo spettro udibile viene suddiviso in otto bande, ciascuna dell'ampiezza di circa 2.500 Hz ($20.000/8=2,5$ kHz); per ogni banda si calcola l'ampiezza corrispondente e, confrontandola con la soglia impostata manualmente sullo smartphone, si determina

un valore binario (un byte per il sistema complessivo) che indica se la luce corrispondente debba essere accesa o spenta.

L'hardware consiste in un sistema "Arduino Powered" composto da una scheda Arduino Uno (la Rev. 3, in particolare...) cui verranno aggiunti uno shield denominato I²C I/O Expander (che analizzeremo a breve) e lo shield Bluetooth RN-42 che già conosciamo.

Il primo shield sarà utilizzato per la gestione fisica dell'accensione delle luci (in particolare, il sistema è munito di 8 uscite a relé che andremo a collegare opportunamente per accendere o spegnere le file di luci tradizionali o LED con cui realizzare il gruppo di illuminazione) mentre il secondo shield verrà utilizzato per realizzare un link Bluetooth tra lo smartphone e Arduino, in modo che lo smartphone passi a quest'ultimo i dati per comandare le luci.

Notate che la decisione di ripartire lo spettro audio in 8 bande di frequenza è dettata dal fatto che lo shield Expander dispone di soli 8 relé. Quanto al Bluetooth, l'abbia-

mo scelto perché tutti gli smartphone Android dispongono di tale connessione wireless, perché per funzionare non necessita di alcun sistema aggiuntivo (in particolare router Wi-Fi in caso di rete TCP/IP wireless) e perché tutto sommato permette una velocità di comunicazione sufficiente per ottenere una risposta immediata (in tempo reale) per la nostra applicazione.

Riassumendo, questo è il funzionamento globale del sistema: da smartphone si avvia la App, si seleziona a quale dispositivo collegarsi e si sceglie il file audio da riprodurre. A questo punto il tutto viene gestito in automatico da software: viene analizzato il segnale sonoro per identificare quali uscite accendere o spegnere; tali risultati vengono inviati in continuazione e ad intervalli di circa 10 millisecondi (praticamente in tempo reale...) all'hardware Arduino che attiva o disattiva le uscite a relé.

Infine, nel caso in cui la App non sia in esecuzione o non ci sia il collegamento Bluetooth, il firmware Arduino si accorge di tale condizione e gestisce le uscite secondo sequenze casuali.

SCHEMA ELETTRICO I²C I/O EXPANDER SHIELD

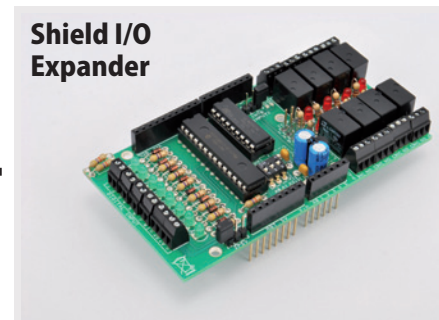
Iniziamo ad analizzare lo schema elettrico dello shield I²C I/O Expander, che è costruito attorno al componente U1, un I/O Expander

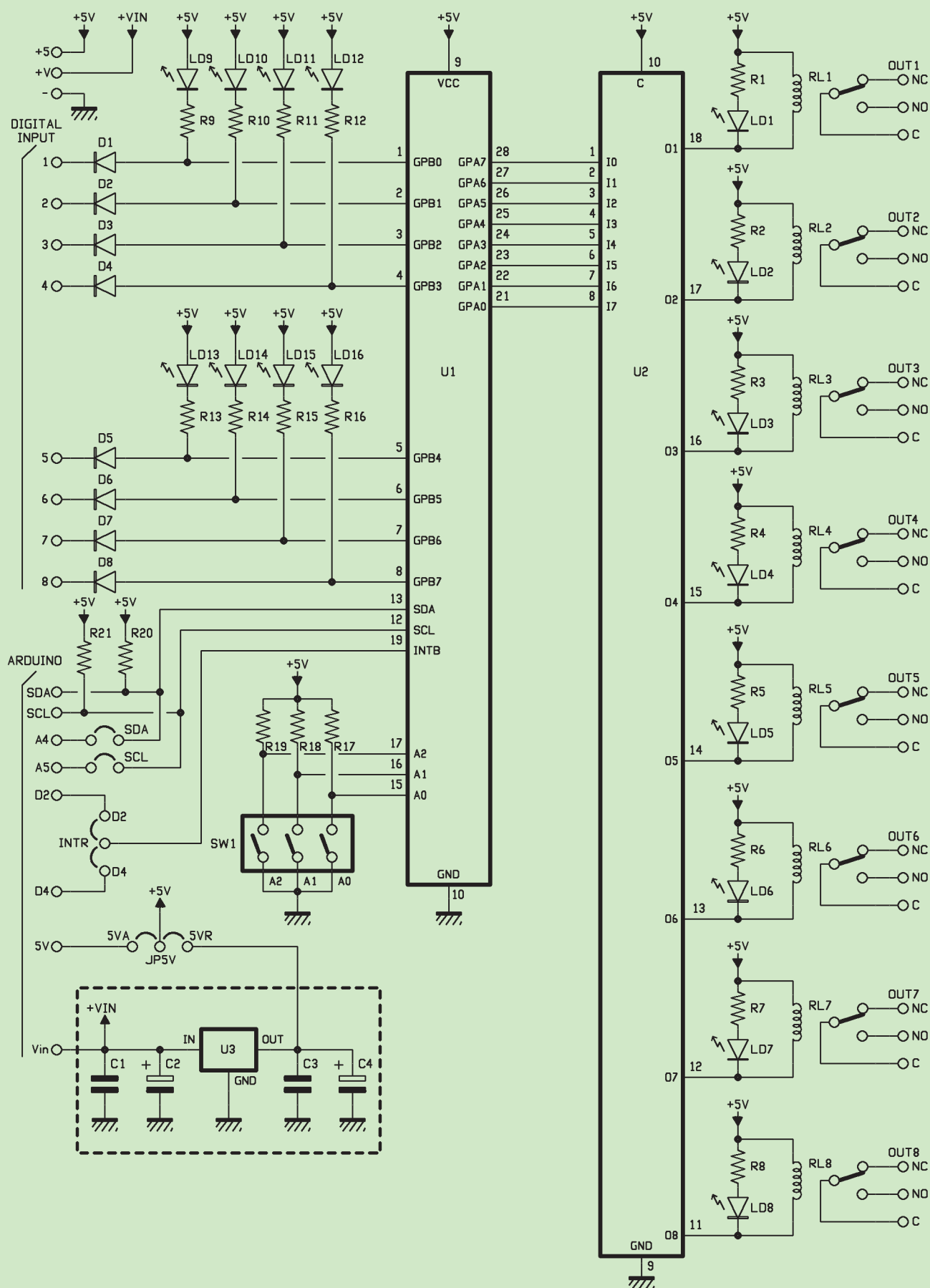


+



+





[schema elettrico dello SHIELD I/O EXPANDER]

Lo shield **Bluetooth**

La connessione con lo smartphone viene assicurata da un opportuno shield Bluetooth già descritto ampiamente nella 4^a puntata del Corso Android, pubblicata nel fascicolo n° 172 di Elettronica In (cui rimandiamo per gli approfondimenti del caso) 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 carat-

tere 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 a lato 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: 220 ohm (0805)
R2: 220 ohm (0805)
R3: 220 ohm (0805)
R4: 1 kohm (0805)
R5: 1 kohm (0805)
R6: 1 kohm (0805)
R7: 1 kohm (0805)
R8: 20 kohm (0805)
R9: 10 kohm (0805)
C1: 100 nF ceramico (0805)
U1: RN-42
RST: Microswitch
LD1: LED rosso (0805)
LD2: LED verde (0805)
LD3: LED giallo (0805)
SW1: Dip-Switch 4 vie

Varie:

- Strip M/F 3 vie (2 pz.)
- Strip M/F 6 vie
- Strip M/F 8 vie (2 pz.)
- Strip M/F 10 vie
- Strip Maschio 2 vie
- Strip Femmina 3 vie
- Strip Femmina 4 vie (2 pz.)
- Strip Femmina 3 vie (3 pz.)
- Jumper
- Circuito stampato

MCP23017 prodotto da Microchip. Come descritto nel datasheet, questo chip dispone in totale di 16 pin GPIO (suddivisi in due porte chiamate A e B) configurabili singolarmente come ingresso o uscita. Nel nostro schema abbiamo deciso di settare 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, un integrato progettato appositamente come driver per relé (al suo interno dispone di diversi stadi a transistor che permettono di pilotare senza problemi le bobine dei relé).

Gli ingressi, invece, (porta B) sono collegati ad apposite sezioni circuitali che realizzano gli input digitali; facciamo notare che nell'applicazione descritta in questo articolo non vengono utilizzati, ma per chiarezza li analizziamo lo stesso in quanto sicuramente saranno utili in progetti futuri. Ogni ingresso dispone di un diodo di

protezione, una resistenza di pull-up ed un LED. 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 alto il relativo LED è spento e l'I/O Expander indicherà che l'ingresso è "1".

La gestione dell'MCP23017 avviene tramite protocollo I²C Bus e scrittura/lettura di appositi registri interni.

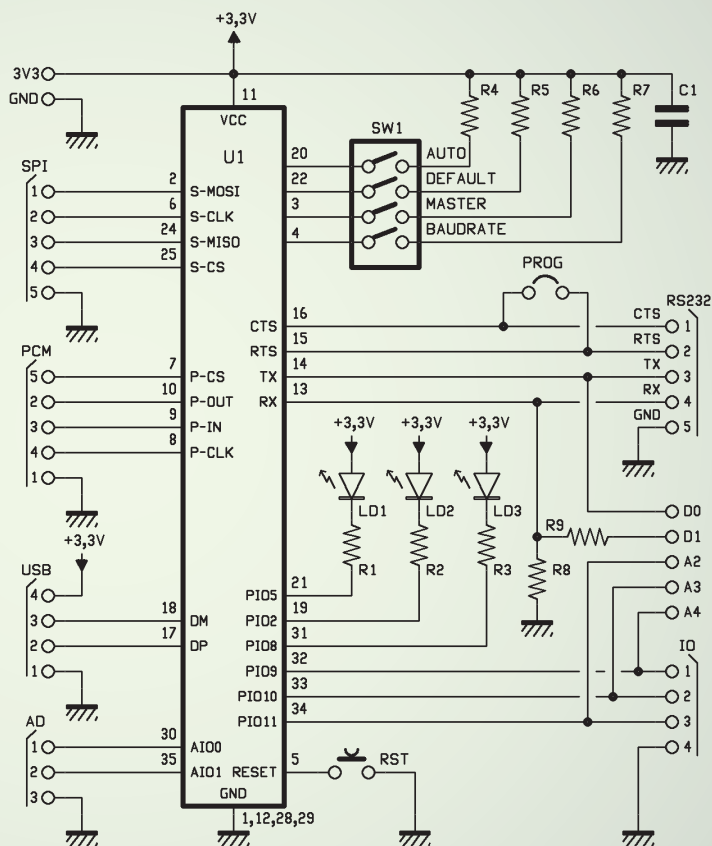
Come noto, il bus I²C è realizzato da 2 pin (SDA, ossia dati, ed SCL, cioè clock); sull'integrato sono disponibili sui pin 13 e 12 e, come da schema, vengono portati sui corrispondenti pin della scheda Arduino.

L'MCP23017 offre la possibilità di generare un interrupt a fronte della variazione di almeno un pin programmato come input; nel nostro caso non abbiamo sfruttato questa funzione, che però potrebbe essere utile per gestire

da Arduino gli input in modalità interrupt e non polling (il che assicura più velocità senza appesantire il bus I²C). Chiaramente, per sfruttare questa caratteristica è necessario scrivere opportunamente il firmware.

Nello shield abbiamo previsto la possibilità di portare (tramite ponticello) il pin di interrupt dell'MCP23017 (INTB, pin 19) in ingresso su due differenti pin di Arduino (D2 o D4) in modo da adattarlo il più possibile al reale utilizzo.

L'indirizzo I²C-Bus dell'MCP23017 viene selezionato tramite dip-switch SW1; in esso sono presenti 3 dip, pertanto risulta possibile collegare fino a 8 diversi shield senza che interferiscano tra di loro. Infine è presente una sezione di alimentazione composta dal regolatore U3 (integrato L78L05) che trasforma la tensione di ingresso Vin ai 5 volt necessari per far funzionare tutto il circuito.



Segnaliamo anche che tramite il ponticello JP5V è possibile scegliere di bypassare il regolatore ma utilizzare direttamente i 5V forniti dalla scheda Arduino.

LIBRERIA ARDUINO I²C I/O EXPANDER SHIELD

Per il progetto abbiamo sviluppato una libreria software Arduino che fornisce tutte le routine necessarie alla rilevazione degli shield collegati e alla relativa gestione degli I/O (potete scaricare tale libreria dal nostro sito www.elettronica.in.it).

La comunicazione con l'MCP23017 è basata sull'I²C-Bus, pertanto la libreria utilizza "Wire.h" di Arduino; è presente una funzione "begin(int i2cAddress)" per inizializzare il singolo shield identificato tramite indirizzo I²C, ed anche 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 in particolare ci interessano sono le "digitalRead(int pin)", "digitalWrite(int pin, int val)", "digitalWordRead()" e "digitalWordWrite(word w)" che, rispettivamente, permettono di leggere lo stato di un singolo pin di input, scrivere un singolo pin di output, leggere lo stato di tutti i pin di input (8 nel nostro caso) ed infine scrivere lo stato di tutti i pin di output (sempre 8).

Come anticipato in precedenza, la gestione degli input può avvenire tramite interrupt; è pertanto presente un'apposita funzione "pinDisableINT(int pin)" che permette di configurare il singolo pin come generatore di interruzioni o meno (in particolari applicazioni potrebbe essere utile avere la possibilità di non impostare tutti i pin di input come generatori di interrupt, ma farlo selettivamente).

SCHEMA ELETTRICO SHIELD BLUETOOTH RN-42

Il componente fondamentale dello shield Bluetooth è sicuramente il modulo RN-42, la cui descrizione è già stata fatta nel fascicolo n° 168; il modulo, oltre ad avere integrato un chip per la gestione dello stack e della comunicazione Bluetooth, per l'interfacciamento verso l'esterno dispone di diverse porte di comunicazione e di alcuni pin di input/output. Dallo schema elettrico dello shield si vede che la porta UART dell'RN-42 è localizzata sui pin compresi tra 13 e 16; questi vengono portati sia sui pin di Arduino che corrispondono alla porta hardware UART, sia su un connettore esterno di espansione. In teoria con shield inserito ed alimentato non dovrebbe essere possibile aggiornare il firmware Arduino; in pratica, come vedremo tra breve, abbiamo realizzato un piccolo trucco per consentirlo. Su CTS ed RTS è stato posizionato il jumper PROG, che cortocircuita i due segnali: serve perché nel normale funzionamento il controllo di flusso è disabilitato.

Sullo schema sono presenti tre LED di segnalazione gestiti direttamente dall'RN-42 tramite i GPIO 5, 2 ed 8 (un LED rosso di *segnalazione stato*, uno verde di *presenza alimentazione* ed uno, ancora verde, di *comunicazione in corso*).

Nello shield alcuni pin GPIO del modulo (GPIO9, GPIO10 e GPIO11) vengono portati verso l'esterno; per leggere o pilotare questi I/O da Arduino è necessario entrare nella modalità comandi. Esistono due distinti comandi per controllare le linee di I/O (per leggere o per scrivere). I pin vengono anche mappati su A4, A3 e A2 di Arduino.

Il connettore identificato nello shield con la scritta AD porta verso l'esterno 2 ingressi analogici; per esempio, possono essere

Il protocollo di comunicazione

Il protocollo di comunicazione tra scheda e smartphone è abbastanza semplice (anche per permettere elevate velocità di comunicazione). Come certo saprete, il Bluetooth prevede 2 distinte classi di dispositivi: master e slave. Della prima classe fanno parte tutti quei dispositivi che, di propria iniziativa, possono avviare una nuova connessione, mentre della seconda classe si trovano quelli che sono sempre in attesa di richieste di connessione (facendo un equivalente con la comunicazione TCP/IP, i master corrispondono ai client, mentre gli slave ai server). Nel nostro caso, lo smartphone si comporta da master mentre la scheda elettronica da slave.

Il protocollo prevede delle trame di lunghezza massima di 3 byte così strutturate:

- primo byte=identificativo tipo della risorsa. Sono previste 3 possibilità: uscite a relè (OUTRLE_CMD; byte 0x43); nome Bluetooth del sistema (NMEBTH_CMD; byte 0x44) ed infine pin code Bluetooth (PNCBTH_CMD; byte 0x45);
- secondo byte=stato uscite per il comando OUTRLE_CMD oppure indice carattere nome o PIN (per i comandi NMEBTH_CMD e PNCBTH_CMD);
- terzo byte=singolo carattere nome o PIN code.

Infine, il protocollo è basato su request/reply, perciò prevede che ad ogni richiesta di comandi proveniente dal master, lo slave risponda con pacchetti di conferma o risposta.

utilizzati dal modulo per acquisire il livello di tensione sulla batteria oppure per leggere un sensore di temperatura.

Il modulo dispone di una porta SPI (portato verso l'esterno sul relativo connettore) e tramite essa si può scrivere nella flash interna al modulo RN-42 in modo da aggiornarne o sostituirne il firmware; nel normale funzionamento, la connessione non dovrebbe mai essere utilizzata.

L'RN-42 dispone anche di una porta PCM (anch'essa portata su un apposito connettore esterno); tale interfaccia offre la possibilità di inviare via radio, ad un altro dispositivo Bluetooth, uno stream audio ad una velocità di trasferimento di 64 kbps. Per i nostri scopi, questa possibilità non serve. Infine, andiamo al connettore USB: collegandolo al PC tramite un cavo USB, è possibile utilizzare la modalità "HCI over USB" del modulo ed ottenere velocità maggiori (fino a 3 Mbps). Tuttavia, questo è possibile utilizzando solo la versione USB del modulo RN-42 (chiamata appunto RN-42-USB) o aggiornando opportunamente il firmware tramite il bus SPI. Passiamo ora agli altri elementi dello shield: il dip-switch SW1 può collegare alcuni pin (GPIO3,

GPIO4, GPIO6, GPIO7) del modulo a VCC (3,3V) tramite quattro resistenze; questi pin servono per impostare alcune modalità di funzionamento del modulo RN-42 e sono normalmente collegati a massa tramite resistenze di pull-down integrate nel modulo. Spostando il rispettivo dip verso la scritta "ON" viene collegato il pin del modulo a Vcc e quindi a livello logico 1, abilitando la relativa funzionalità. Tutti questi pin vengono interpretati al momento dell'avvio, quindi cambiando lo stato dei dip durante il funzionamento dello shield non si otterranno gli effetti desiderati.

Il pin AUTO è collegato al pin GPIO3 dell'RN-42; se abilitato, permette l'accoppiamento automatico con un altro dispositivo remoto Bluetooth.

DEFAULT è collegato al pin GPIO4 del modulo: permette di ripristinare tutti i settaggi interni al modulo ai valori iniziali di "fabbrica".

MASTER è collegato al pin GPIO6; per impostazione predefinita, il modulo Bluetooth si comporta come un dispositivo slave e quindi può venire rilevato da altri dispositivi Bluetooth e connettersi ad essi, ma non è in grado di iniziare per primo una connessione.

Portando a ON il rispettivo dip, il modulo lavorerà in modalità Master.

BAUDRATE è collegato al pin GPIO7; abilitandolo, viene forzata la velocità di comunicazione per dati e comandi a 9.600 bps., altrimenti verrà utilizzata la velocità desiderata, che avremo precedentemente scritto in flash tramite comando software. Per impostazione predefinita, questo valore è pari a 115.200 baud. Tramite software è possibile impostare anche velocità di trasferimento non standard.

Infine, è presente un pulsante di reset collegato al relativo pin del modulo RN-42. Quando il pulsante è premuto, il modulo viene posto in reset ed in particolare la porta seriale viene scollegata. In questo modo è possibile staccare l'RN-42 dalla porta seriale di Arduino permettendo quindi l'aggiornamento del software anche con shield Bluetooth inserito.

LIBRERIA ARDUINO BLUETOOTH RN-42 SHIELD

Anche per lo shield Bluetooth, abbiamo sviluppato una nostra libreria software Arduino. La comunicazione con il modulo RN-42 avviene tramite 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 presenti le seguenti funzioni:

- "begin()" che inizializza la porta seriale con baud-rate 115.200 ed 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;
- "Set_Operation_Mode_Slave_RN_42()" e "Set_Operation_Mode_Master_RN_42()" rispettivamente per impostare la modalità Bluetooth 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 il nome e il pin code Bluetooth;
 - "Enter_Command_Mode_RN_42()" e "Enter_Data_Mode_RN_42()" per entrare nella modalità comandi e dati.

Per i dettagli di tutte le funzioni, rimandiamo al datasheet dell'RN-42.

Oltre a queste routine di basso livello, quelle che più ci interessano per il nostro firmware Arduino sono "New_Connection_Request_RN_42()" e "Close_Connection_Request_RN_42()" che testano se dall'esterno è arrivata una richiesta, rispettivamente, di nuova o chiusura connessione. Inoltre la "Write_Buffer_Bluetooth_RN_42(char *Buffer_Bluetooth, int Buffer_Length)" e "Read_Buffer_Bluetooth_RN_42(char *Buffer_Bluetooth)" per scrivere o leggere un buffer di dati tramite link Bluetooth. Infine, la "Available_RN_42()" che usiamo per testare quanti byte di dati sono disponibili in ricezione.

SKETCH PER ARDUINO

Avendo definito in precedenza le librerie software per la gestione degli shield I/O Expander e Bluetooth, la realizzazione del software Arduino risulta più semplice e veloce.

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 porta seriale richieste dagli shield, viene utilizzato il solo pin 13 come output per gestire come segnalazione il LED presente sulla board Arduino.

Nella EEPROM, 36 byte sono utilizzati per memorizzare il nome e pin code Bluetooth da associare all'accensione al link wireless.

Nella funzione "setup()" viene configurato il pin digitale, inizializzata la EEPROM, avviato il bus I²C ed infine viene rilevato quale, tra gli 8 possibili shield I/O Expander, è collegato. Per realizzare ciò viene utilizzato il bus I²C ed in particolare l'ACK che gli slave rispondono ai diversi comandi.

Se lo shield risulta collegato, vengono impostati i versi di input e output dell'MCP23017 secondo lo schema elettrico.

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 ("5555"). In seguito si entra nel ciclo principale di esecuzione, in cui si gestisce la comunicazione Bluetooth: si testa la presenza di nuovi byte nel buffer di ricezione seriale ed in caso affermativo questi vengono analizzati. Viene verificata la condizione di richiesta di nuova connessione, di disconnessione, oppure i byte vengono processati secondo le regole del protocollo di comunicazione. In particolare,

esistono solo comandi di scrittura ovvero scrittura dello stato delle uscite a relé e programmazione della configurazione Bluetooth (nome e pin code).

Come già detto, per ottenere un funzionamento che sia assimilabile ad un real-time, l'invio dello stato delle uscite a relé deve avvenire ad un ritmo abbastanza sostenuto (di circa 10 msec); questo ci ha portati a definire un protocollo di comunicazione che sia il più semplice possibile e che permetta di ottenere tempi minimi di trasmissione ed analisi dei dati. In particolare, abbiamo definito

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

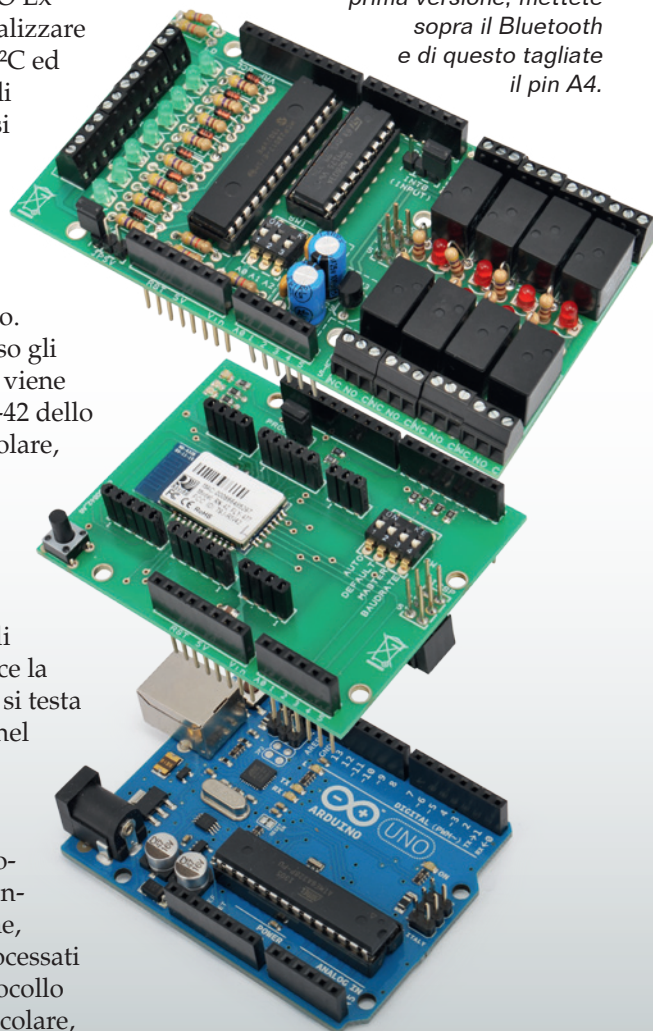


Fig. 1



Fig. 2

semplici pacchetti composti al massimo da tre byte, con un solo byte di informazioni di controllo ed il resto di dati reali.

Anche per il calcolo del time out di verifica della disconnessione Bluetooth, abbiamo usato un ritardo minimo (appena 5 msec) ma ripetuto più frequentemente.

SOFTWARE ANDROID

Il software sviluppato per smartphone Android ("XMasControl") permette il collegamento e la gestione da remoto del sistema. Una volta avviata ed in esecuzione, l'App è costituita dal thread principale più due ulteriori thread (che lavorano nascosti, in background) eseguiti in contemporanea ma sincronizzati tra loro.

Dei due processi aggiuntivi, il primo thread si occupa della riproduzione e dell'analisi delle frequenze audio (oltre ad aggiornare la grafica) mentre il secondo si fa carico della gestione della comunicazione Bluetooth. Per l'analisi dell'audio viene utilizzata la Trasformata di Fourier Veloce (la "famosa" FFT, Fast Fourier Transform) la quale, ricevuti

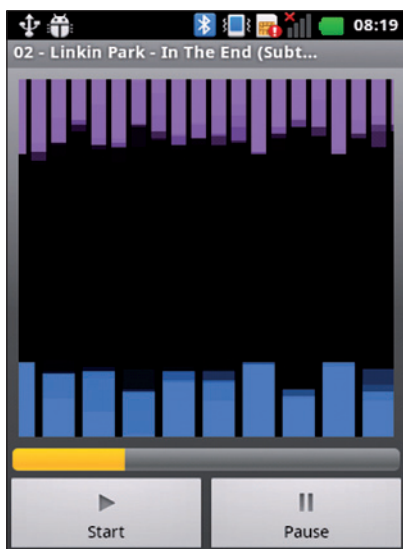


Fig. 3

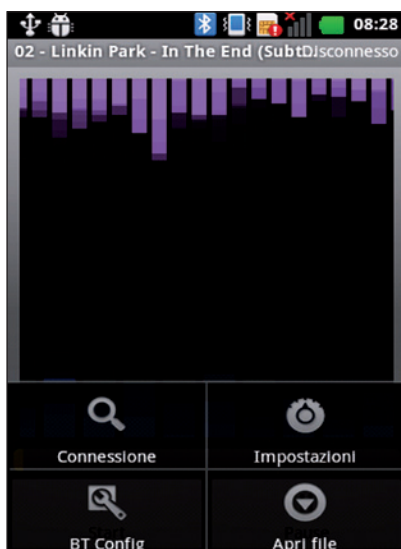


Fig. 4

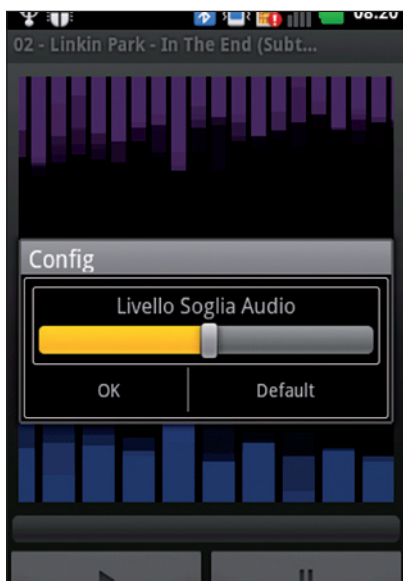


Fig. 5



Fig. 6

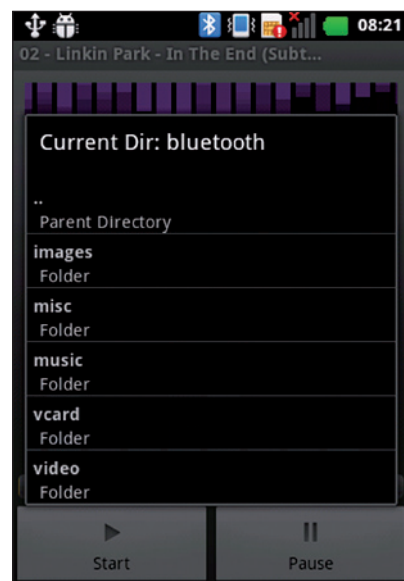


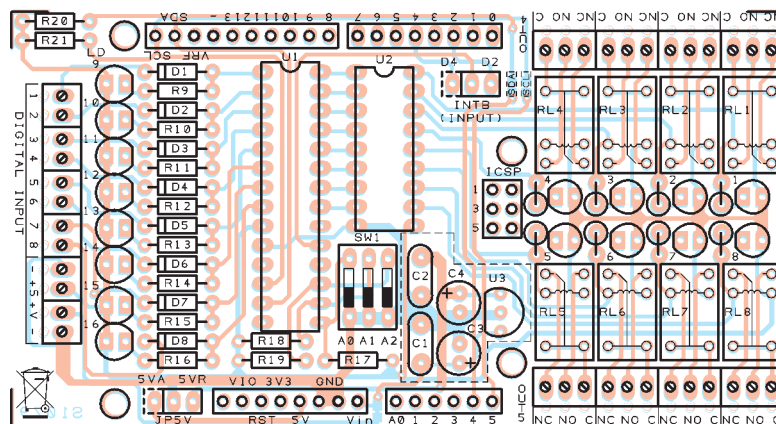
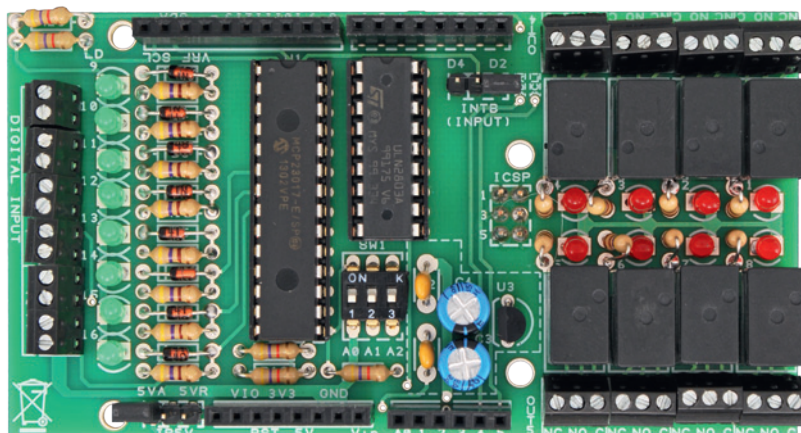
Fig. 7

Elenco Componenti:

R1÷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



in ingresso diversi campioni PCM dell'audio (in particolare 1.024 nel nostro progetto), fornisce in uscita una sequenza di byte (anche in questo caso 1.024) in cui ogni dato indica l'ampiezza del segnale audio relativo a una determinata gamma di frequenze.

Originariamente lo spettro sonoro è quindi suddiviso in 1.024 piccole bande audio; il nostro codice riceve in ingresso questi 1.024 byte e li raggruppa (sommando l'ampiezza) in blocchetti da $1.024/8=128$ byte in modo da ottenere l'ampiezza di otto bande più ampie. In seguito tali livelli di segnale vengono confrontati con un livello di soglia (modificabile dall'utente) in modo da ottenere 8 bit (un byte) in cui ciascun bit indica se l'uscita hardware abbinata alla frequenza

debba essere attiva o disattiva. Tutti questi byte vengono passati al secondo thread (comunicazione Bluetooth) che in sequenza li invia fisicamente all'hardware Arduino. Con questa tecnica si riesce ad ottenere l'invio dei campioni ad un ritmo di circa 15 ms, riuscendo quindi a realizzare una trasmissione praticamente in tempo reale. All'avvio, dopo una schermata iniziale di presentazione del software, viene avviata la scansione e inizia il rilevamento di tutti i dispositivi Bluetooth presenti all'interno della zona di copertura dello smartphone. Nella lista visualizzata in **Fig. 1** vengono mostrati sia tutti i sistemi già accoppiati, sia i nuovi rilevati; cliccando sulla voce della scheda ("MusicXMasControl" nel no-

stro esempio) la si seleziona e si richiede l'avvio del collegamento. A schermo è inoltre presente un pulsante per richiedere una nuova esecuzione della scansione e l'aggiornamento della lista. A questo punto, se è la prima volta che il dispositivo viene rilevato, la App richiede la digitazione del pin code (**Fig. 2**) e in seguito viene avviato un MP3 campione (*Test.MP3*, contenuto all'interno dell'App), quindi si passa nella schermata principale del programma (**Fig. 3**). In questa, attraverso istogrammi, viene visualizzato lo spettro delle frequenze sonore (la banda completa è suddivisa in 19 sezioni per la visualizzazione superiore e in 10 per quella inferiore); inoltre, nella parte estrema superiore viene visualizzato il nome del file

Montaggio su Arduino UNO

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. Nella nostra applicazione, la CPU Arduino comunica con lo shield I/O Expander utilizzando 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 bloc-

cato e pertanto il software non viene eseguito correttamente. La soluzione a questo inconveniente consiste, durante il montaggio uno sopra l'altro degli shield, nell'inserire lo shield Bluetooth come ultimo (sopra a tutti, anche per evitare che il modulo RN-42 sia chiuso sopra e sotto e quindi la comunicazione RF venga ostacolata); inoltre, conviene lasciare flottante il pin che corrisponde ad A4 di Arduino (flottante significa non inserito nel corrispondente strip femmina, quindi piegato perché 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). Nessun problema, invece, se usate Arduino Uno Rev. 3.

attualmente in esecuzione. Nella parte inferiore è invece presente una barra di scorrimento che rappresenta l'esecuzione temporale del file audio, oltre a due pulsanti per avviare o mettere in pausa l'esecuzione.

Facendo clic sul pulsante *menu* di Android, in basso viene visualizzato il menu di scelta della configurazione (Fig. 4) composto da quattro pulsanti: "Connessione" per tornare nella sezione di scansione e selezione scheda hardware a cui collegarsi; "BT Config" per selezionare la configurazione Bluetooth della scheda; "Impostazioni" per la configurazione del livello di soglia audio; "Apri file" per selezionare quale file audio eseguire (la selezione avviene attraverso le cartelle e i file presenti nello smartphone).

Nella prima pagina di configurazione (Fig. 5) è presente una barra di selezione orizzontale attraverso cui scegliere il livello della soglia audio di confronto. Premendo "OK" il valore impostato viene confermato; "Default" ritorna alla configurazione predefinita. Nella seconda pagina di configu-

razione (Fig. 6) sono presenti due sezioni: "Nome sistema" e "Codice PIN"; abbinati ad ogni sezione sono presenti un campo di testo per impostare la configurazione ed un pulsante per richiedere l'invio e modifica della singola configurazione.

Nell'ultima pagina ("Apri file", Fig. 7) è possibile selezionare il file audio da riprodurre. Partendo dalla cartella root, viene visualizzato il contenuto della SD card inserita nello smartphone; cliccando su una cartella ("Folder") si entra nella sotto directory ("...", "Parent folder" per tornare alla cartella superiore). Invece, facendo clic su un file audio valido, questo viene selezionato e la pagina viene chiusa (la App torna alla finestra principale).

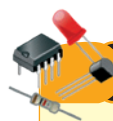
REALIZZAZIONE PRATICA

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

Arduino Uno è un prodotto che si compera già pronto; quanto allo shield Bluetooth, viene venduto da

Futura Elettronica praticamente già montato (quasi tutti i componenti sono in SMD e sono già saldati); rimangono da saldare solo il pulsante di reset, gli strip maschio/femmina di montaggio tipici di Arduino e il dip-switch a quattro poli.

Per quanto riguarda la saldatura di tali componenti, se si ha un minimo di dimestichezza con il saldatore non dovrebbero esserci particolari problemi. L'unico circuito da montare è lo shield I/O Expander (che volendo si può acquistare da Futura Elettronica in kit di montaggio), la cui realizzazione è alla portata di tutti perché richiede componenti a montaggio tradizionale. Charamente è necessario disporre di un saldatore; si può iniziare dai componenti a più basso profilo (resistenze e diodi) quindi procedere via-via con il dip-switch a 3 poli, i ponticelli, i LED, i condensatori ceramici ed elettrolitici ed infine il regolatore di tensione. In seguito



per il MATERIALE

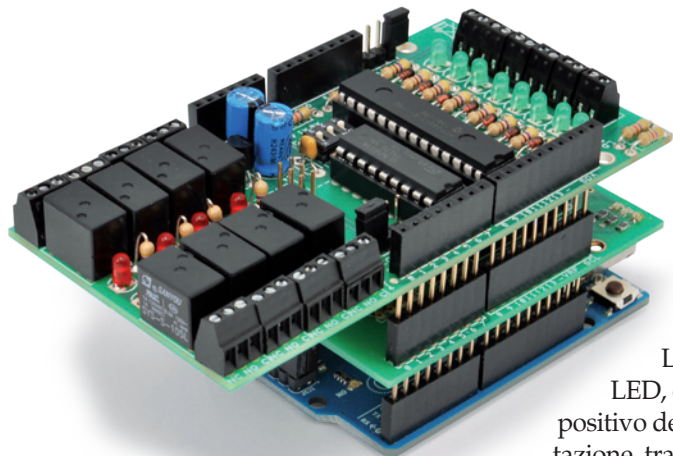
Tutti i componenti utilizzati in questo progetto sono di facile reperibilità; il master del circuito stampato dello shield I/O Expander per Arduino può essere scaricato dal sito della rivista. Questo shield (cod. FT1079K) è anche disponibile in kit al prezzo di 34,00 Euro; la scatola di montaggio comprende 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>



passate alle morsettiere; per gli input sono presenti sei morsettiere a due poli, mentre per le uscite ce ne sono otto a tre poli. Passate poi agli integrati: prima saldate gli zoccoli corrispondenti, poi inserite gli integrati facendo attenzione alla posizione della tacca che indica il pin 1; procedete con i relé ed infine inserite e saldate gli strip maschio/femmina di collegamento degli shield.

La posizione dei pin del dip-switch a tre poli non è significativa, in quanto, come abbiamo spiegato nelle pagine precedenti, questi assegnano l'indirizzo I²C al chip MCP23017, ma, essendo presente un solo slave, sicuramente non ci saranno problemi di incompatibilità. Se i pin dei dip-switch vengono impostati a ON il relativo pin dell'MCP23017 è collegato a massa e pertanto il corrispondente bit dell'indirizzo I²C è resettato a "0"; caso inverso se il pin è posizionato su OFF il bit dell'indirizzo sarà settato a "1".

La corrispondenza tra i singoli pin dei 3 e i bit A0, A1 ed A2 sono evidenziati sulla serigrafia dello stampato. Infine, per il corretto funzionamento del progetto, il dip-switch a quattro poli dello shield Bluetooth andrà configurato con tutti e quattro i pin in posizione OFF.

Consideriamo ora il collegamento delle luci. Come visto dallo schema elettrico dello shield I/O Expander, le otto uscite gestite sono disponibili come contatti puliti (cioè come

contatti C, NC ed NO di otto relé) pertanto le userete come interruttori opportunamente inseriti nei circuiti di alimentazione delle lampade o LED. Se comandate dei

LED, collegatene l'anodo al positivo della tensione di alimentazione, tramite una resistenza di valore opportuno, quindi connettete il contatto C del relé a massa e l'NO al catodo. Se usate strip di LED, è lo stesso, perché avrete liberi un anodo e un catodo. Per le lampadine la connessione è identica, ma non servono resistenze né il rispetto della polarità.

POSSIBILI ESPANSIONI

Il progetto presentato in queste pagine è stato ideato supponendo di utilizzare un solo shield I²C I/O Expander e pertanto di gestire al massimo 8 luci. Di conseguenza lo spettro audio è stato suddiviso in 8 bande corrispondenti agli 8 output. Tuttavia, dal momento che il sistema supporta fino ad un massimo di 8 shield Expander uno sopra l'altro (ognuno va identificato da un indirizzo I²C univoco) in teoria si possono gestire fino a 64 uscite. In questo modo si otterrebbe un effetto luminoso più preciso, in quanto si avrebbero più bande ma molto più strette (circa 300 Hz).

Chiaramente andrebbero riviste alcune caratteristiche del progetto: per quanto riguarda la App andrebbe aggiornato il calcolo delle potenze delle bande (che aumentano di numero) e volendo anche la visualizzazione grafica (operazione non necessaria, avendo funzione soltanto estetica, ma che aiuterebbe l'utente a verificare visivamente che il sistema ripeta l'andamento del brano riprodotto).

In parte va rivisto anche il protocollo di comunicazione, in quanto per ogni trasmissione bisognerebbe in-

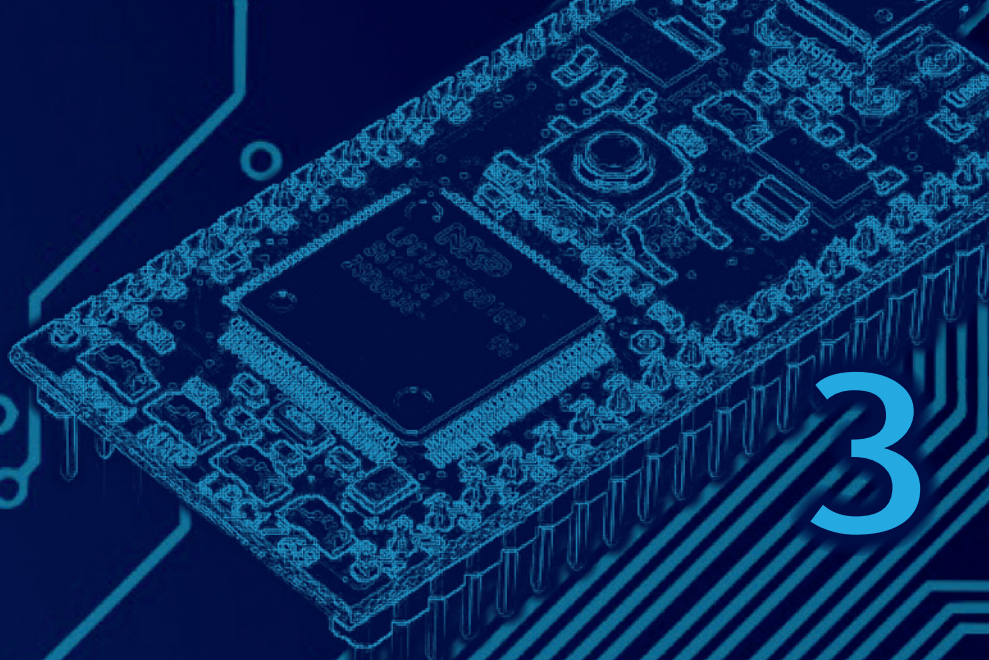
viare 8 byte di informazioni dati, invece di uno; per il software Arduino andrebbe gestita la comunicazione I²C con più I/O. Infine, per l'hardware Arduino andrebbero inseriti uno sopra l'altro gli 8 shield Expander, ciascuno andrebbe identificato con un indirizzo I²C diverso ed alle uscite di ciascuno di essi andrebbero collegate correttamente le luci. A future espansioni si prestano anche gli 8 input dello shield Extender, che attualmente non utilizziamo; un possibile impiego potrebbe essere pensare di gestirli per creare differenti effetti luminosi a seconda del verificarsi di particolari condizioni.

Download del software

Il modulo RN-42 del Bluetooth shield comunica con Arduino attraverso la porta hardware UART; la stessa porta è anche utilizzata dall'IDE Arduino per effettuare il download del firmware. Esiste quindi un accesso contemporaneo alla risorsa e chiaramente le due cose non possono funzionare insieme. Quindi, quando dovete programmare la scheda Arduino, una prima soluzione al problema potrebbe essere di rimuovere lo shield Bluetooth, effettuare il download, inserire nuovamente lo shield ed infine accendere la scheda.

Questa prima soluzione è abbastanza scomoda e poco pratica; un'alternativa più comoda consiste nell'usare il pulsante di reset presente sullo shield. La pressione del pulsante pone il modulo RN-42 nello stato di reset; di conseguenza il modulo viene disabilitato, ma soprattutto risulta scollegato dalla porta UART. Fintanto che il pulsante risulta premuto, Arduino ha disponibile la porta seriale, che pertanto può essere utilizzata per il download. A programmazione finita basta rilasciare il pulsante, riavviare la scheda e il tutto dovrebbe funzionare correttamente.

Utilizziamo la nostra piattaforma di prototipazione per pilotare un display LCD mediante uno shield Arduino in modo da visualizzare scritte a piacere e l'ora dell'RTC, ma anche per generare forme d'onda usando il DAC del LPC176. Terza puntata.



Conoscere e usare

mbed

dell'ing. Vincenzo Mendola

Nelle puntate precedenti abbiamo introdotto le schede di prototipazione mbed, mostrandovi le nozioni base per il loro utilizzo per aiutarvi a prendere confidenza con il relativo hardware; abbiamo inoltre introdotto un comodo circuito adattatore che ci permette di utilizzare con facilità gli shield di Arduino sulla piattaforma mbed a 32 bit. Con questo adattatore abbiamo iniziato a realizzare alcune applicazioni, descrivendo di volta in volta il relativo codice. In questa puntata prosegui-

mo la descrizione delle applicazioni e lo facciamo proponendo la gestione di vari display standard (a controllo parallelo) a cristalli liquidi, ma anche la generazione di segnali di forma d'onda più comune. Partiamo con il primo esempio applicativo, utilizzando l'LCD shield per Arduino, presentato sul numero 174 di marzo 2013 di Elettronica In (e commercializzato da Futura Elettronica con la sigla 7100-FT1030K) per mostrare qualche esempio di codice che permette ad mbed di interfacciarsi ad un comune LCD. Per prima cosa provvediamo alla

Listato 1

```
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p30, p24, p12, p22, p23, p27,
TextLCD::LCD16x2);

int main()
{
    lcd.locate(0,0);
    lcd.printf("MBED SHIELD");
    lcd.locate(0,1);
    lcd.printf("ELETTRONICA IN");
}
```

visualizzazione delle scritte "MBED SHIELD" e "ELETTRONICA IN" (**Listato 1**).

Com'è facile intuire, la libreria *TextLCD* serve per rappresentare caratteri alfanumerici su LCD basati sul diffusissimo driver HD44780 o controller compatibili. Fra parentesi sono riportati, nell'ordine, i pin dell'mbed connessi ai terminali "RS", "E", "D4-D7" del modulo LCD, mentre "TextLCD::LCD16x2" serve per definire la tipologia del modulo connesso. Attualmente sono supportati i display LCD, elencati nella **Tabella 1**, ognuno con la relativa istruzione da includere nel codice. Tornando al **Listato 1**, l'istruzione "lcd.locate(0,1);" analogamente a quanto avviene per Arduino, indica la colonna e la riga in cui visualizzare il testo tra le virgolette del comando "lcd.printf();".

In maniera analoga al codice di Arduino, la riga #include "TextLCD.h" richiama la libreria per la scrittura di caratteri sugli LCD, mentre la riga "TextLCD lcd(p27, p23, p22, p12, p21, p8, TextLCD::LCD16x2);" serve per definire i collegamenti tra l'mbed (connesso in modalità a 4 BIT) e il display, oltre che la tipologia (due righe e 16 caratteri per riga); il comando "lcd.printf("ELETTRONICA IN\n");" serve per scrivere il testo desiderato. Come per tutte le funzioni e le librerie di mbed, è possibile approfondire l'argomento mediante il cookbook (<https://mbed.org/cookbook/Homepage>) e l'handbook (<https://mbed.org/handbook/Homepage>).

Tabella 1

Tipo di display	Istruzione
8x1 LCD panel	TextLCD::LCD8x1
8x2 LCD panel	TextLCD::LCD8x2
16x1 LCD panel	TextLCD::LCD16x1
16x4 LCD panel	TextLCD::LCD16x4
24x2 LCD panel	TextLCD::LCD24x2
24x4 LCD panel	TextLCD::LCD24x4
40x2 LCD panel	TextLCD::LCD40x2
40x4 LCD panel	TextLCD::LCD40x4

ge). Qualora si volesse realizzare un proprio LCD shield, basta semplicemente modificare i parametri riportati fra le parentesi del comando TextLCD in modo da rispettare le connessioni utilizzate nel proprio PCB. Per esempio possiamo servirci del meteo shield (fascicolo n° 171 di Elettronica In) ed utilizzarlo per scrivere del testo a piacimento (**Fig. 1**). Il relativo codice è quello nel **Listato 2**.

Passiamo ora ad un esempio un po' più complesso: proviamo a visualizzare l'ora e la data del nostro RTC shield su un comune display LCD 16x2; il codice è quello mostrato nel **Listato 3**.

Ricordiamo che l'RTC shield e l'RTC presente sul meteo shield, utilizzano entrambi il DS1307, che ha una logica operante a 5V, mentre gli mbed lavorano a 3,3V; anche se sono "5V tolerant" (possono accettare al loro ingresso tensioni di 5V e leggerle correttamente come livello alto senza danneggiarsi) si



Fig. 1

Listato 2

```
#include "mbed.h"
#include "TextLCD.h"

TextLCD lcd(p27, p23, p22, p12, p21, p8, TextLCD::LCD16x2); //meteo shield

int main() {
    lcd.printf("ELETTRONICA IN\n");
}
```

Listato 3

```
#include "mbed.h"
#include "TextLCD.h"
#include "ds1307.h"

TextLCD lcd(p27, p23, p22, p12, p21, p8, TextLCD::LCD16x2); //meteo shield
Serial pc(USBTX, USBRX);
DS1307 my1307(p9,p10);

int sec = 0;
int min = 55;
int hours = 20;
int day = 7;
int date = 3;
int month = 3;
int year = 13;

void test_rw(int test)
{
    if (test == 0) pc.printf("Last R/W operation passed!\n\r");
    else pc.printf("Last R/W operation failed!\n\r");
}

int main()
{test_rw(my1307.settime( sec, min, hours, day, date, month, year)); // to set RTC
while(1){ test_rw(my1307.gettime( &sec, &min, &hours, &day, &date, &month, &year));
    lcd.locate(0,0);
    lcd.printf("%2D",hours);
    lcd.printf(":%2D",min);
    lcd.printf(":%2D",sec);

    lcd.locate(0,1);
    switch(day){
    case 1:
        lcd.printf("LUN");
        break;
    case 2:
        lcd.printf("MAR");
        break;
    case 3:
        lcd.printf("MER");
        break;
    case 4:
        lcd.printf("GIO");
        break;
    case 5:
        lcd.printf("VEN");
        break;
    case 6:
        lcd.printf("SAB");
        break;
    case 7:
        lcd.printf("DOM");
        break;
    }

    lcd.printf("\ %2D",date);
    lcd.printf("/%2D",month);
    lcd.printf("/%2D",year);
    }
}
```

consiglia di utilizzare un traslatore di livello (5V/3,3V) bidirezionale, ossia un circuito in grado di trasformare in 0/3,3V i livelli TTL in ingresso e di elevare a 5 volt i livelli logici alti in uscita dai pin dell'mbed. Il circuito può essere qualcosa di analogo allo shield suggerito nel fascicolo n° 173 della rivista, per adattare RaspberryPi agli shield per Arduino. La riga "switch-case" del Listato 3 può anche essere sostituita dalla seguente porzione di

codice (il risultato resta invariato):

```
if (day == 1)
    lcd.printf("LUN");
if (day == 2)
    lcd.printf("MAR");
if (day == 3)
    lcd.printf("MER");
if (day == 4)
    lcd.printf("GIO");
if (day == 5)
    lcd.printf("VEN");
if (day == 6)
    lcd.printf("SAB");
```


Listato 4

```

#include "mbed.h"
#include "TextLCD.h"
#include "ds1307.h" //http://mbed.org/users/harrypowers/code/DS1307/docs/c3e4da8feb10/ds1307_8h_source.html

Serial pc(USBTX, USBRX);
TextLCD lcd(p27, p23, p22, p12, p21, p8, TextLCD::LCD16x2);

DS1307 my1307(p9,p10);

int sec = 0;
int min = 00;
int hours = 16;
int day = 7;
int date = 23;
int month = 2;
int year = 13;

void test_rw(int test)
{
    if (test == 0) pc.printf("Last R/W operation passed!\n\r");
    else pc.printf("Last R/W operation failed!\n\r");
}

float VADC= 3.31;
float Vcc=4.21;

int DPR = 0;
int RHCORR = 0;
int PCORR = 0;

float STAMPA_T = 0;
float STAMPA_U = 0;
float STAMPA_P = 0 ;

AnalogIn val(p17);
AnalogIn UM(p16);
AnalogIn Pascal(p15);

float temp()
{
    float T=0.0;
    float nread = 100.0;          // NUMBER OF READINGS
    float somma = 0.0;
    for (int i=0; i<nread; i++) {
        float TADC=val.read();
        T = (((VADC*TADC)-0.5)* 100); //TEMPERATURE
        somma += T;
    }
    wait_ms(100);
    return (somma/nread);
}

float readUMID()
{
    float RHout=0.0;
    float nread = 100.0;          // NUMBER OF READINGS
    float somma = 0.0;
    for (int i=0; i<nread; i++) {
        float UMADC = UM.read();
        RHout=(((UMADC)-0.1515)/0.00636)+RHCORR; //HUMIDITY
        somma += RHout;
    }
    wait_ms(100);
    return (somma / nread);
}

float pressure()
{
    float P=0.0;
    float nread = 100.0;          // NUMBER OF READINGS
    float somma = 0.0;
    for (int i=0; i<nread; i++) {
        float PascalADC= Pascal.read();
        P=(((PascalADC*VADC)/Vcc+0.095)/0.009)*10+DPR+PCORR; //PRESSURE TRANSFERT FUNCTION
        somma += P;
    }
}

```



```

    wait_ms(100);
    return (somma / nread);
}

int main()
{
    //test_rw(my1307.settime( sec, min, hours, day, date, month, year)); //REGOLA ORARIO E DATA
    while (true) {

        STAMPA_T= (temp());
        STAMPA_U= (readUMID());
        STAMPA_P = (pressure());
        lcd.locate(0, 0);
        lcd.printf("%2.1f",STAMPA_T); //SHOW ONLY THE FIRST DECIMAL
        wait_ms(200);
        lcd.locate(4,0);
        lcd.printf("C");
        lcd.locate(6, 0);
        lcd.printf("%2.1f",STAMPA_U); //SHOW ONLY THE FIRST DECIMAL
        lcd.locate(10,0);
        lcd.putc(37);
        wait_ms(200);

        lcd.locate(12, 0);
        lcd.printf("%4.0f",STAMPA_P); //SHOW ONLY THE INTEGER PART
        wait_ms(200);

        test_rw(my1307.gettime( &sec, &min, &hours, &day, &date, &month, &year));
        lcd.locate(0,1); // Print and refresh data on line 2 of the LCD display
        lcd.printf("%2D",hours);
        lcd.printf(":%2D",min);
        //lcd.printf(":%2D",sec);
        //lcd.printf(" %1D",DAY);

        if (day == 1)
            lcd.printf(" D");
        if (day == 2)
            lcd.printf(" L");
        if (day == 3)
            lcd.printf(" Ma");
        if (day == 4)
            lcd.printf(" Me");
        if (day == 5)
            lcd.printf(" G");
        if (day == 6)
            lcd.printf(" V");
        if (day == 7)
            lcd.printf(" S");

        lcd.printf(" %2D",date);
        lcd.printf("/%2D",month);
        lcd.printf("/%2D",year);

    }
}

```

```

if (day == 7)
    lcd.printf("DOM");

```

A questo punto vediamo come modificare il codice scritto per Arduino allo scopo di utilizzare il meteo shield con l'mbed. Per prima cosa ricordiamo che sul PCB dello shield sono presenti due ponticelli che permettono di selezionare la corretta tensione di alimentazione del sensore di pressione, previsti a suo tempo per facilitare l'uso del circuito anche con i sensori a 3,3V (ad esempio il MP3H6115A), tensione usata dai microcontrollori di mbed e da Arduino Due. Prima di mostrare una possibile versione del codice per mbed, ricordiamo che il modello LCP1768 incorpora già un RTC, mentre la versione LPC11U24 no, per cui abbiamo inserito anche la libreria per il DS1307, scaricabile all'indirizzo web <http://mbed>.

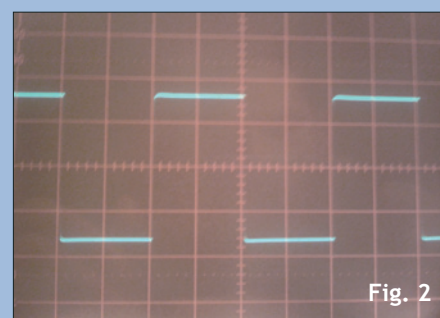


Fig. 2

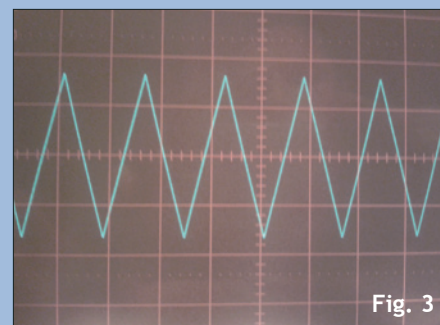


Fig. 3

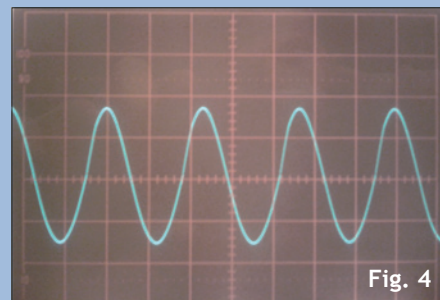


Fig. 4

Listato 5

```
#include "mbed.h"

PwmOut led(LED1);

int main() {
    while(1) {
        for(float p = 0.0; p < 1.0; p += 0.1) {
            led = p;
            wait(0.1);
        }
    }
}
```

Listato 6

```
#include "mbed.h"

PwmOut leds[] = {(LED1), (LED2), (LED3), (LED4)};
int i;

float p=0.0;

int main()
{
    while(1) {

        for (i=0; i<4; i++) {
            for(p = 0.0; p <= 1.0; p += 0.01) {
                leds[i] = p;
                wait(0.01);
            }
        }
        for (i = 3; i>=0; i--) {
            for(p = 1.0; p >= 0.0; p -= 0.01) {
                leds[i] = p;
                wait(0.01);
            }
        }

        wait(0.1);
    }
}
```

Listato 7

```
#include "mbed.h"

AnalogOut DA(p18);

int main()
{
    while(1) {
        DA=1;                                //ONDA QUADRA
        wait_us(100);
        DA=0;
        wait_us(100);
    }
}
```

org/users/harrypowers/code/DS1307/docs/c3e4da-8feb10/ds1307_8h_source.html. Si noti che tale libreria può comunque anche essere utilizzata con l'mbed basato sul processore Cortex M3. Il codice mbed è quello contenuto nel **Listato 4**. Un'altra funzionalità molto utile nei sistemi basati su microcontrollore è la creazione di segnali PWM, solitamente ottenuta da appositi moduli PWM integrati nei micro stessi. Nel

Listato 8

```
#include "mbed.h"

float i;

AnalogOut DA(p18);

int main() {
    for (i=0; i<1;i=i+0.01)    //ONDA TRIANGOLARE
    {DA=i;
      wait_us(1);}

    for (i=1; i>0; i=i-0.01)
    {DA=i;
      wait_us(1);}

}
}
```

Listato 9

```
#include "mbed.h"

float i;

AnalogOut DA(p18);

int main() {
    for (float i=0; i<360; i++) { //ONDA SINUSOIDALE
        DA = sin(i/180*3.14)*0.5+0.5;
        wait_us(1);
    }
}
```

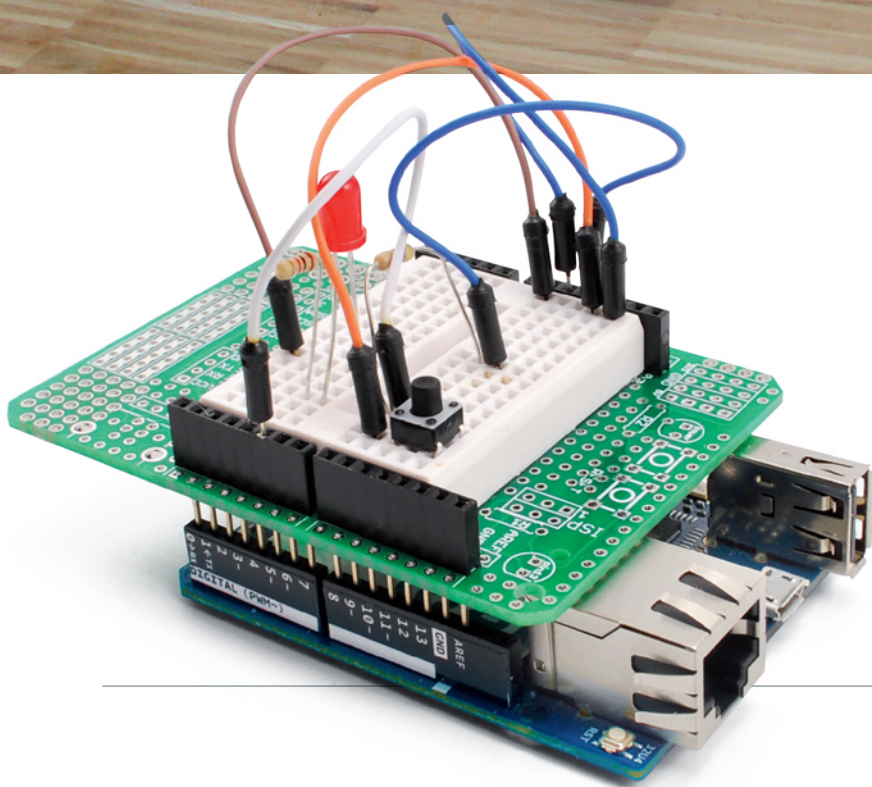
caso della nostra piattaforma mbed, la generazione del PWM è implementata attraverso la funzione PwmOut (<https://mbed.org/handbook/PwmOut>). Nel **Listato 5** e nel **Listato 6** trovate due semplici applicazioni pratiche che sfruttano i quattro LED a bordo dell'mbed LPC1768. Vediamo ora come generare delle semplici forme d'onda utilizzando il DAC (Digital to Analog converter) presente nell'mbed LPC1768. Cominciamo con la generazione di un'onda rettangolare come quella visibile in **Fig. 2**, la quale può essere generata senza alcuna difficoltà con il codice che trovate nel **Listato 7**. Con le piattaforme mbed non ci si limita alla generazione della classica onda quadra, perché è anche possibile generare forme d'onda più complesse come la triangolare (**Fig. 3**) e la sinusoidale (**Fig. 4**), ottenibili rispettivamente con i codici che trovate nei **Listati 8** e **9**. Come sempre, consigliamo di leggere attentamente la documentazione ufficiale per ulteriori dettagli e per imparare a sfruttare tutte le caratteristiche delle funzioni e delle librerie disponibili, che qui per ragioni di spazio, non abbiamo potuto descrivere. Bene, con questo abbiamo concluso. Riprenderemo il discorso nel prossimo fascicolo. ■

ARDUINO YÚN: CONTROLLER VIDEO

Collegiamo una telecamera e sfruttiamo le nuove funzionalità di Arduino Yún per realizzare un "cane" da guardia elettronico, che ci permetta di tenere sotto controllo le nostre cose, un angolo nascosto del giardino, o la nostra automobile posteggiata in cortile.

di MARCO MAGAGNIN

Nel precedente numero della rivista vi abbiamo presentato in anteprima le caratteristiche della nuova scheda della famiglia Arduino. La combinazione delle due "anime" presenti nella scheda Arduino Yún, costituite dalla tradizionale anima Arduino e da GNU/Linux, permette di ampliare notevolmente il ventaglio di applicazioni che possono essere affrontate, con la semplicità tipica della filosofia Arduino. In questa occasione abbiamo voluto affrontare un progetto che mai avrebbe potuto essere realizzato prima d'ora con una scheda Arduino. Collegare una webcam, potere visualizzare le immagini su un browser web trasmesse direttamente in "streaming" da un server web presente sulla scheda stessa e attivare delle azioni se "qualcosa" o "qualcuno" si muove nell'area del campo di ripresa che vogliamo tenere sotto controllo. Immaginiamo il caso di un cortile, magari con la nostra auto posteggiata all'inter-



Configurazione WiFi di Arduino Yún

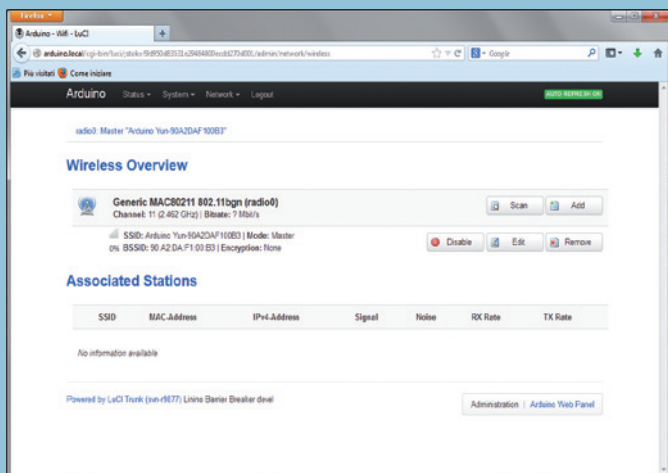


Fig. A

Per l'amministrazione del sistema operativo Linino è disponibile lo strumento LuCI, ereditato dal progenitore OpenWRT. Tra le varie funzionalità è disponibile il pannello di configurazione dell'interfaccia WiFi, inizialmente

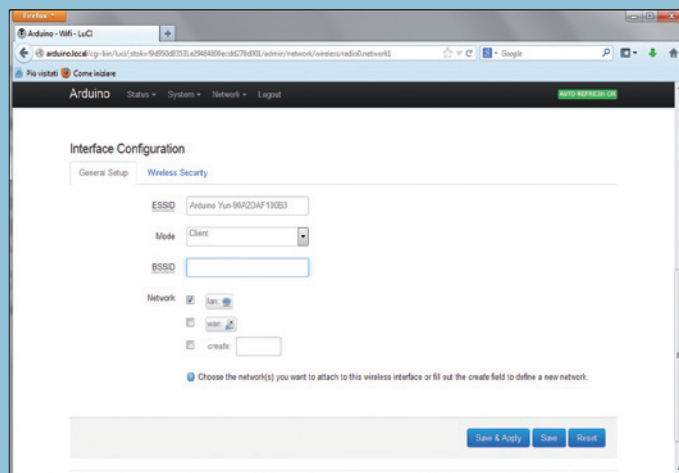


Fig. B

configurata come Access Point. Si entra nello strumento di amministrazione semplicemente mediante un browser web digitando l'indirizzo "arduino.local". Viene richiesta la password "arduino". Nella pagina di stato delle

connessioni clicchiamo sul pulsante "Configure" e nella pagina successiva clicchiamo sul link "advanced configuration panel (LuCI)". Ci troviamo nello strumento di amministrazione LuCI. In questa pagina selezioniamo

no. In genere in quest'area non dovrebbe muoversi nessuno, in particolare intorno al luogo dove è posteggiata l'auto. Con questo progetto potremmo, per esempio, inquadrare l'auto con la telecamera e selezionare l'area che inquadra la portiera come area "sensibile". Se qualcosa o qualcuno si muove in quest'area scatterà una procedura d'allarme per avvisarci dell'evento. A seguito del verificarsi dell'evento potremmo poi decidere di inviare un tweet di segnalazione, giusto per riutilizzare l'esempio proposto nel numero 180 della rivista, oppure una mail, come descritto in questo articolo, o anche azioni combinate, come quelle proposte, oltre all'accensione di un LED di notifica. Anche un piccolo progetto come questo ripropone la differenza di approccio da tenere nel progettare applicazioni per una scheda embedded rispetto a quelle per una scheda Arduino classica. Una scheda Arduino classica richiede il tipico approccio da microcontrollore, ovvero la progettazione di un'applica-

zione "monoblocco" costituita da un unico programma, "sketch", nel caso di Arduino, che una volta caricato nella memoria della scheda ne personalizza il comportamento. L'attenzione deve essere posta nel definire la giusta sequenza delle operazioni, il rispetto delle modalità di interfacciamento dei sensori e nel definire le temporizzazioni in modo da ottenere misurazioni corrette. La connettività con l'esterno è scarsa (anche se la disponibilità di shield Ethernet, WiFi e Bluetooth ha mitigato questo aspetto) mentre è pressoché nulla la possibilità di eseguire in contemporanea funzioni di controllo sensori e servizi web impegnativi come la gestione e la trasmissione di immagini. Diversa è la situazione per un sistema embedded con due processori dedicati come in Arduino Yún. In questo caso l'attenzione va posta sul disegno complessivo dell'architettura che si vuole realizzare, in particolare alla suddivisione in moduli dell'applicazione, nella collocazione di ciascun modulo

sul processore più appropriato e alla interazione e comunicazione tra questi. L'applicazione poggia su un'infrastruttura già esistente, costituita dal sistema operativo, nel nostro caso Linino, e da collezioni di programmi già pronti per svolgere molte delle funzioni che andremo a realizzare. Una volta disegnata l'applicazione, la prima operazione da compiere è la scelta dei componenti già disponibili, che ci possono essere utili. Realizzeremo codice solo per le parti strettamente specifiche dell'applicazione che non possiamo trovare già pronte. Questo approccio diventa possibile grazie alla presenza del sistema operativo GNU/Linux, e delle collezioni di programmi disponibili sotto varie licenze di distribuzione open source. Questo è vero anche per la sezione Arduino della scheda, per la quale sono disponibili tantissime librerie ed esempi di applicazioni già pronti. Dobbiamo anche porre attenzione alla comunicazione tra le diverse componenti dell'applicazione, che "girano"

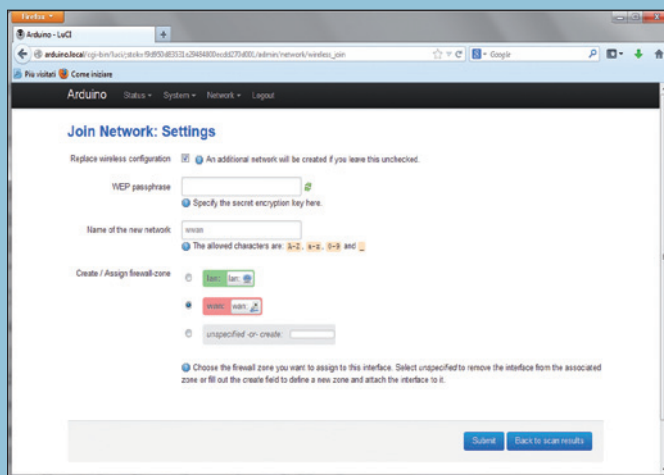
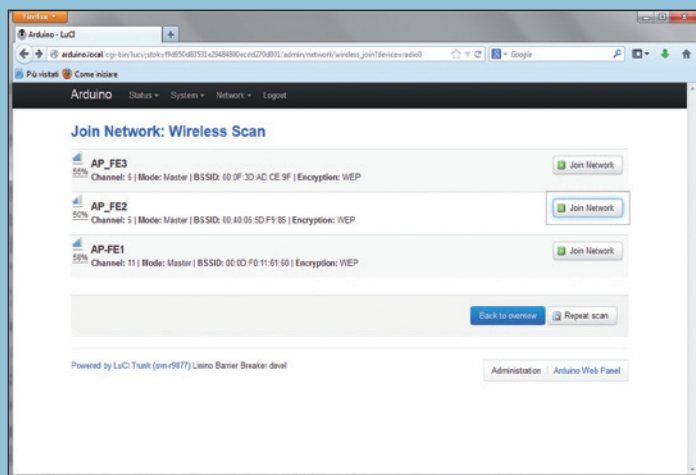


Fig. C

Fig. D

il menu “Network” ed il sottomenu “Wifi”. Nella pagina visibile in **Fig. A**, clicchiamo sul pulsante “Edit” e, nella pagina di **Fig. B**, scorriamo verso il basso fino alla sezione “Interface configuration” e modifichiamo il campo

mode, da “Access Point” a “Client”, utilizzando la selezione a tendina. Salviamo cliccando sul pulsante “Save & Apply”. Selezioniamo ancora il menu “Network” ed il sottomenu “Wifi” e questa volta clicchiamo sul

pulsante “Scan”. Ci appare l’elenco delle reti WiFi disponibili (**Fig. C**). Selezioniamo quella alla quale vogliamo collegarci ed inseriamo le credenziali di accesso nella pagina di configurazione di **Fig. D**.

in concorrenza sui processori: sempre un solo programma per Arduino mentre ve ne possono essere diversi in contemporanea nella sezione GNU/Linux. Dobbiamo quindi progettare le interfacce di comunicazione tra i processi, attività che deve essere coordinata sia in base alle esigenze applicative, sia tenendo conto della necessità di evitare situazioni di conflitto, per esempio nell’utilizzo di risorse condivise.

DESCRIZIONE DEL PROGETTO

Ma procediamo con ordine, quello che vogliamo realizzare è visibile nello schema di **Fig.1**: un sistema di monitoraggio di un’area da “proteggere”. In caso di rilevazione di un movimento nell’area sotto sorveglianza, vogliamo che il sistema ci avvisi in remoto con l’invio di una mail ed in locale con l’accensione di un LED. Le immagini che “immortalano” l’intruso devono essere memorizzate in una cartella sulla SDCard della scheda per una visione a “posteriori”. Un pulsante

permette di “azzerare” la notifica visuale via LED. La webcam è collegata alla porta USB del processore Atheros AR9339. Come software per la gestione in “streaming” delle immagini riprese dalla telecamera, per la rilevazione del movimento e archiviazione dei relativi fotogrammi utilizziamo il pacchetto “motion”,

realizzato appositamente per il riconoscimento del movimento. Per fare riconoscere la telecamera al sistema operativo Linino, dobbiamo anche caricare il driver per la periferica video. Ci siamo

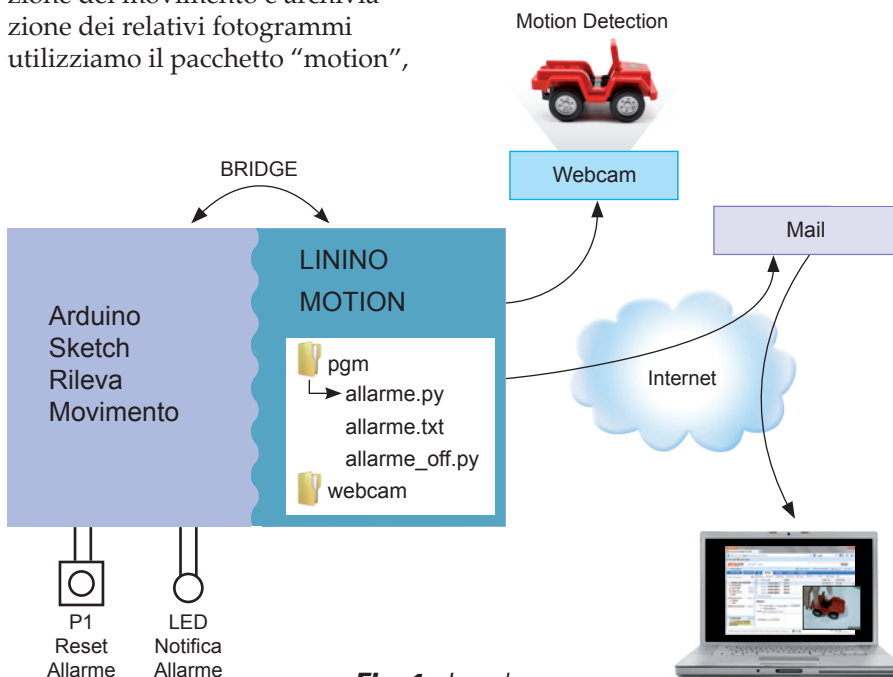
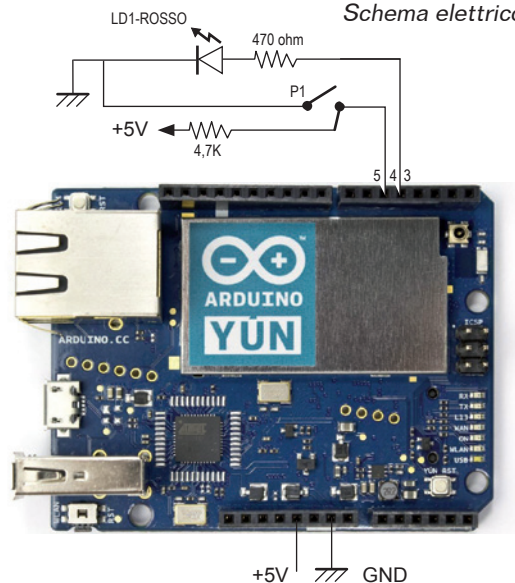


Fig. 1 - Lo schema dell'applicazione.

Fig. 2
Schema elettrico



già occupati della gestione delle periferiche e dei relativi driver in GNU/Linux, nei numeri scorsi della rivista e nel libro "Raspberry Pi, il mio primo Linux embedded". La particolarità in Linino riguarda la possibilità di aggiungere driver dall'esterno, senza necessità di dover riconfigurare e ricompilare i moduli del kernel di GNU/Linux. Il pacchetto motion ci consente di realizzare lo streaming delle immagini della telecamera, che rendiamo visibile via browser. Ogni volta che viene intercettato un movimento nell'area di ripresa da tenere sotto controllo, vogliamo che

vengano registrati una serie di fotogrammi dell'intruso e vogliamo inoltre che il sistema ci avvisi dell'evento sia con l'invio di una e-mail al nostro indirizzo, che con l'accensione di un LED, che rimane acceso sino a quando non lo spegniamo premendo un pulsante. È anche possibile, nel caso di allarme, farsi inviare un tweet sfruttando lo sketch (con qualche modifica), presentato nell'esempio del numero 180 della rivista. Quando viene intercettato un movimento, la funzione motion innesca l'esecuzione di un programma in ambiente Linino che procede ad eseguire le azioni di notifica. Il programma "allarme.py" come prima azione di notifica modifica il contenuto di un file "allarme.txt" portandolo dal valore "0" a "1". Una specie di "semaforo". Poi procede all'invio della mail all'indirizzo specificato allegando l'ultimo fotogramma generato dall'azione di rilevamento. Per la rilevazione della notifica da parte dello sketch Arduino utilizziamo le funzionalità rese disponibili dall'architettura Bridge. Lo sketch in esecuzione su Arduino, utilizzando la libreria "FileIO" legge periodicamente il contenuto del file "allarme.txt"

e, se trova il valore "1", accende il LED rosso collegato al pin 4 di Arduino. Sempre se il valore nel file è "1" lo sketch, utilizzando questa volta la funzionalità "Process", richiede l'esecuzione di un programma su Linino, "allarme_off.py" che riporti a zero il contenuto del file. Il LED rosso resta acceso fino a quando non viene spento volutamente premendo il pulsante collegato al pin 5 di Arduino.

REALIZZAZIONE PRATICA

La realizzazione di questo progetto ci permette di approfondire alcuni aspetti dell'amministrazione del sistema operativo Linino. Prima però realizziamo il semplice circuito visibile in Fig. 2. Noi abbiamo utilizzato il circuito su breadboard utilizzato nell'esempio del numero 180 della rivista dal quale abbiamo semplicemente eliminato il LED verde. Sempre per la realizzazione di questo progetto è necessario che la scheda Arduino Yún sia nella configurazione prevista nell'esempio del numero 180, con inserita una microSDCard, nella quale siano presenti le cartelle

/arduino e
/data

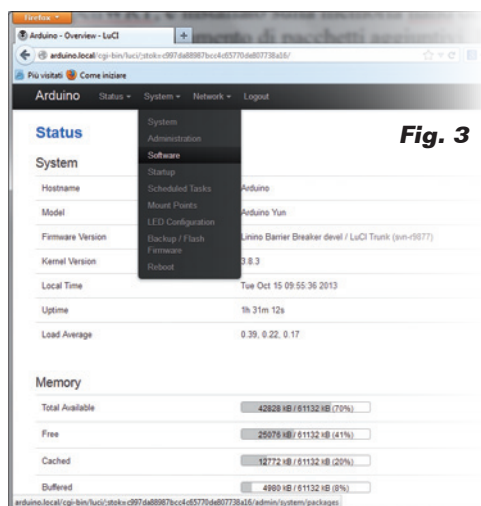


Fig. 3

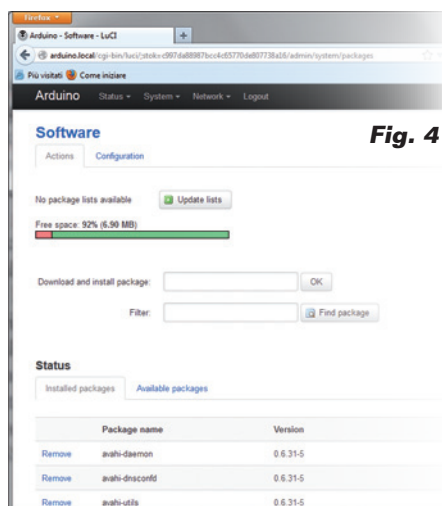


Fig. 4

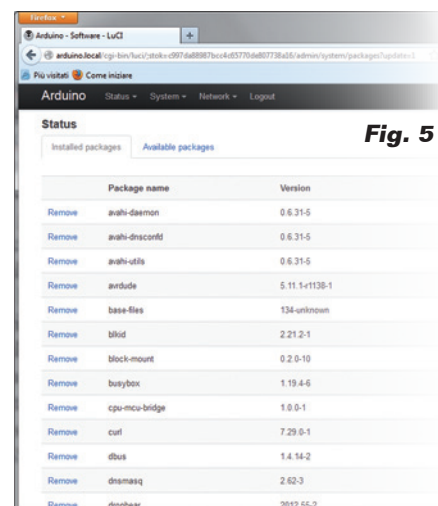
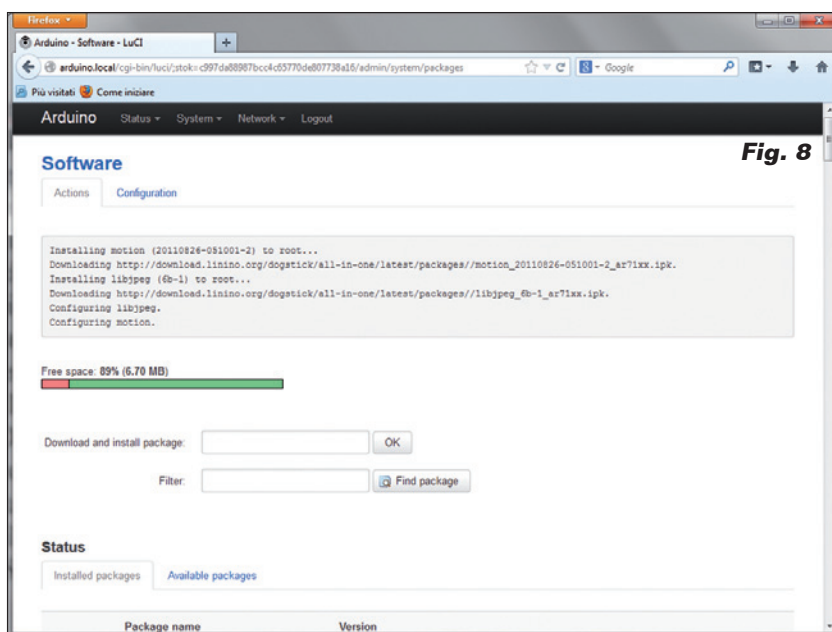
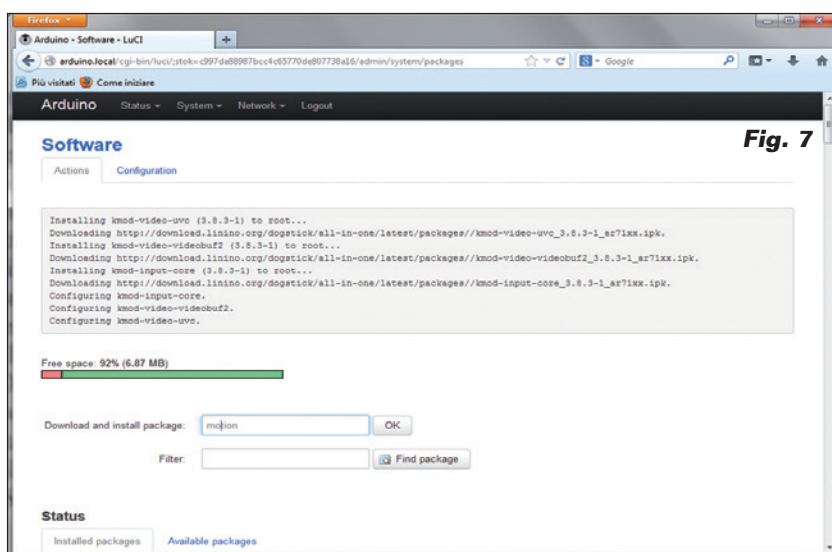
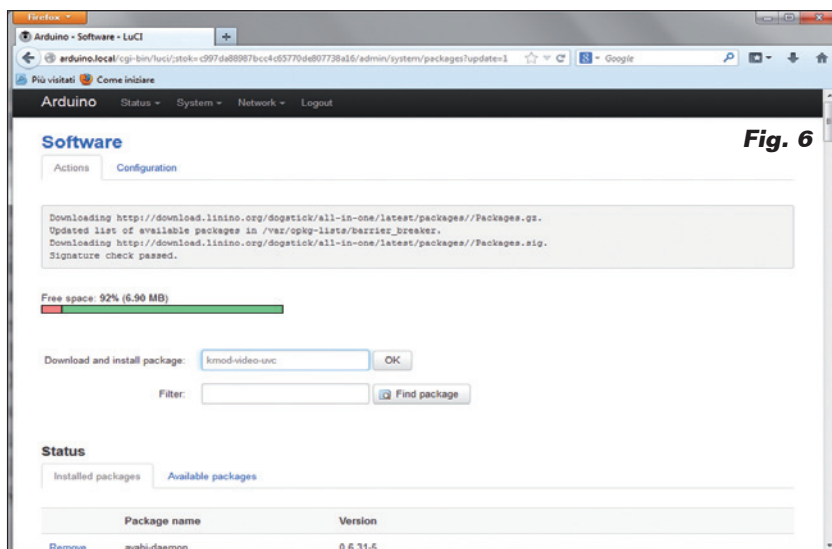


Fig. 5

Torniamo poi alla configurazione "software". Lo strumento a disposizione è LuCi, l'applicativo con interfaccia web, che ci fornisce un menu di funzionalità di amministrazione, suddiviso per argomenti, dei vari aspetti del sistema operativo Linino, tra i quali lo strumento di gestione dei pacchetti. Negli articoli dedicati alla scheda Raspberry Pi abbiamo sempre gestito l'installazione dei pacchetti utilizzando l'interfaccia a linea di comando. Ovviamente anche in Linino possiamo utilizzare l'interfaccia a linea di comando, utilizzando il gestore di pacchetti opkg, (open package manager), l'equivalente di APT di Raspberry Pi, ma è bene introdurre anche gli strumenti dotati di interfaccia grafica che, quando disponibili e ben fatti, semplificano la vita agli utilizzatori. Una nota di approfondimento, prima di cominciare. Su Arduino Yún il sistema operativo Linino, come il suo progenitore OpenWRT, è installato sulla memoria NAND del dispositivo. Una memoria di 64Mb di capacità che lascia poco spazio al caricamento di pacchetti aggiuntivi. Per poter estendere la memoria dove caricare i pacchetti è necessario un intervento di hacking della scheda che vi descriveremo in un prossimo articolo. Per ora accontentiamoci di caricare i nuovi pacchetti nella memoria NAND. Collegiamo la scheda alla nostra rete via cavo Ethernet oppure via connessione WiFi, ma in questo caso dobbiamo configurare la scheda come client WiFi, come descritto nel riquadro "Configurazione WiFi" di Arduino Yún. In ogni caso assicuriamoci che Arduino Yún sia collegato alla stessa rete del nostro PC. Attendiamo qualche tempo, il boot completo di Arduino Yún dura circa mezzo minuto, assai più lungo di quello



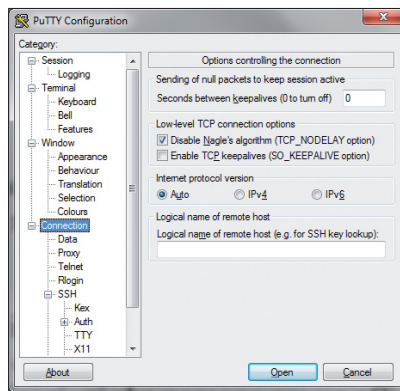
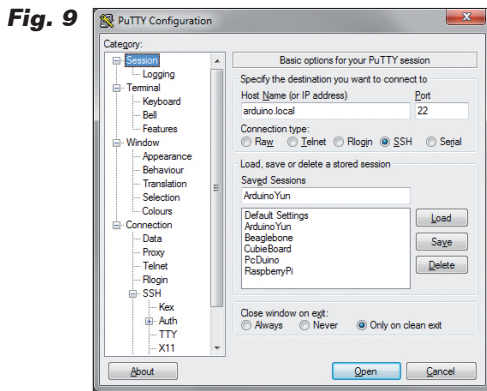
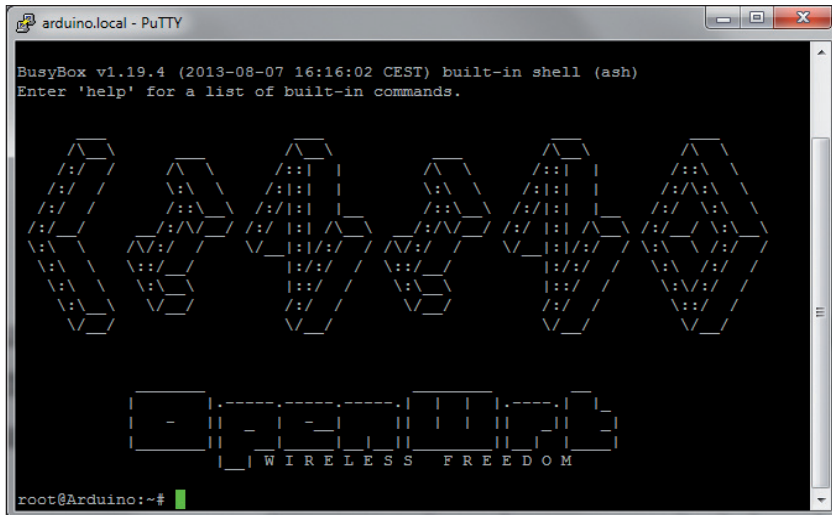


Fig. 10

Fig. 11



di una normale scheda Arduino. Apriamo un browser e digitiamo l'indirizzo: <http://arduino.local>. Diamo la password "arduino" ed entriamo nell'applicazione LuCI. Clicchiamo sul pulsante azzurro "Configure" e poi, nella pagina successiva, sul link "advanced configuration panel (luci)". Ci appare la schermata di Fig. 3 dove dal menu "System" selezioniamo la voce "Software" (Fig. 4). In questo modo entriamo nella pagina di gestione dei pacchetti software. Possiamo vedere lo

spazio libero per l'installazione di nuovi pacchetti. Per curiosità scorriamo la pagina verso il basso, dove possiamo vedere la lista dei pacchetti già installati (Fig. 5). Se avete seguito i nostri articoli dedicati a Raspberry Pi riconoscerete molti dei pacchetti installati. Per prima cosa aggiorniamo l'elenco dei pacchetti disponibili cliccando sul pulsante "Update lists", l'equivalente del comando "opkg update" dato da linea di comando. Ottenuto l'aggiornamento del catalogo dei

pacchetti disponibili, installiamo il driver per la webcam. Inseriamo il nome del pacchetto "kmod-video-uvic" nel campo di testo a destra dell'etichetta "Download and install package" e clicchiamo sul pulsante "OK" (Fig. 6). Al termine otteniamo la pagina di Fig. 7. Con la stessa procedura installiamo il pacchetto "motion". Terminata anche questa installazione (Fig. 8) dobbiamo configurare il pacchetto.

A tal fine, dobbiamo entrare nel cuore del file system, ed il modo migliore è quello di utilizzare il servizio SSH (Secure Shell) che è il servizio di accesso remoto normalmente utilizzato dagli utenti GNU/Linux per amministrare i sistemi collegandosi da un PC esterno. Il servizio è attivo in modo predefinito su Linino, mentre per utilizzarlo da PC dobbiamo installare i relativi programmi client. Abbiamo descritto questo argomento nel numero 169 della rivista, riguardo alle modalità di accesso a Raspberry Pi. Ricordiamo solo che i pacchetti da reperire sono Putty e WinSCP. Si trovano con una semplice ricerca sul web, sono open source e di libero utilizzo. Una volta installati vanno configurati per collegarsi ad Arduino Yún. Per prima cosa lanciamo il programma Putty, lo configuriamo come visibile nelle Fig. 9 e Fig. 10 e ci colleghiamo ad Arduino Yún. Ci vengono richieste le credenziali di login. Digitiamo "root" alla richiesta di login e rispondiamo

Fig. 12

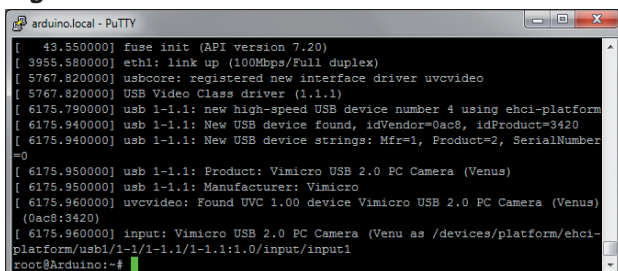
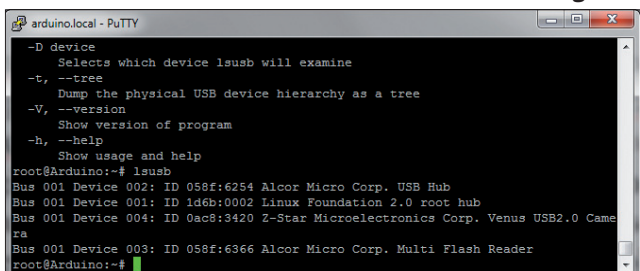


Fig. 13



con “arduino” alla richiesta della password. Otterremo la pagina di benvenuto della shell di Lini-
no visibile in **Fig. 11**.

Ora colleghiamo la webcam alla presa USB-A (quella grande), che fa capo al processore Atheros che ospita Linino. Diamo il comando:

dmesg

In fondo alla lista vediamo se la webcam è stata riconosciuta, come visibile nella **Fig. 12**. Ora diamo il comando:

lsusb

e verifichiamo che la webcam sia riconosciuta come device USB (**Fig. 13**).

Possiamo controllare con quale nome di file (file di device) viene riconosciuta la webcam, in genere “video0”, lanciando lo strumento WinSCP, il file manager che utilizza il protocollo SSH. Configuriamo WinSCP come visibile nelle **Fig. 14, 15, 16 e 17**. Andiamo alla cartella /dev e cerchiamo un file di nome *video0*: si tratta del file di device che identifica la webcam (**Fig. 18**).

Ora, utilizzando sempre WinSCP, configuriamo il pacchetto motion. Andiamo alla cartella /etc/ e clicchiamo sul nome di file “motion.conf” (**Fig. 19**). Si apre un editor di testo con il contenuto del file di configurazione.

Di tutta la lunga lista di parametri di configurazione analizziamo e configuriamo quelli indispensabili alla partenza iniziale del pacchetto. Tutti gli altri parametri potranno essere configurati in un secondo momento utilizzando l’interfaccia web.

Iniziamo dal parametro:

```
; area_detect value
```

Se non specifichiamo nulla, sarà

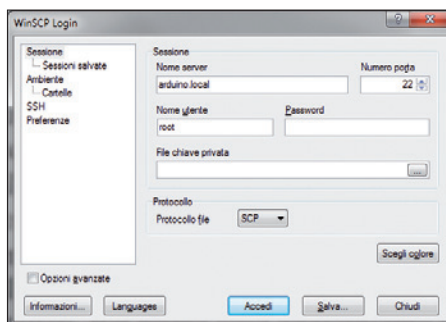


Fig. 14

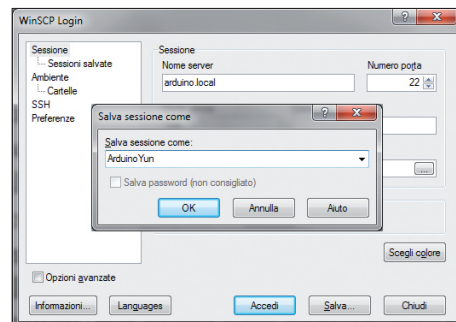


Fig. 15

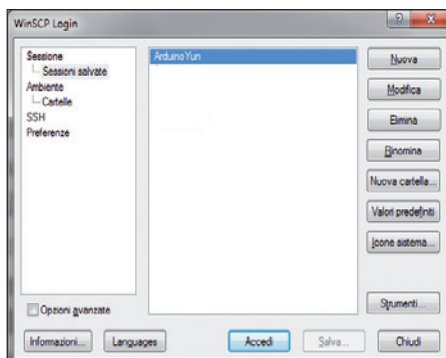


Fig. 16

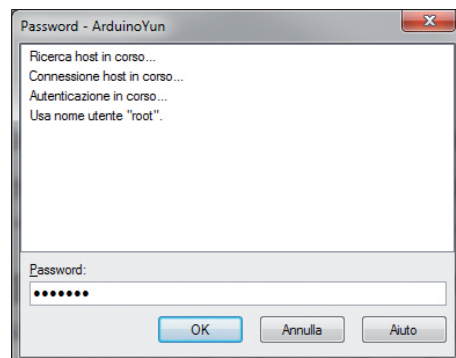


Fig. 17

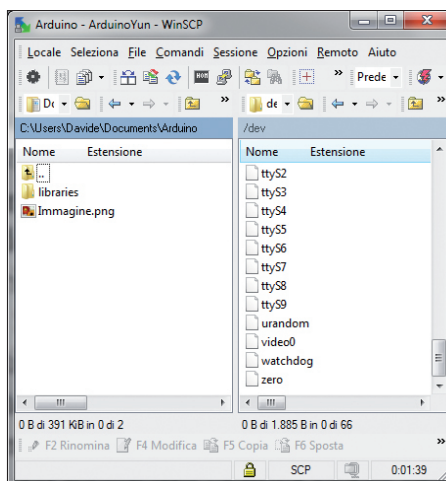


Fig. 18

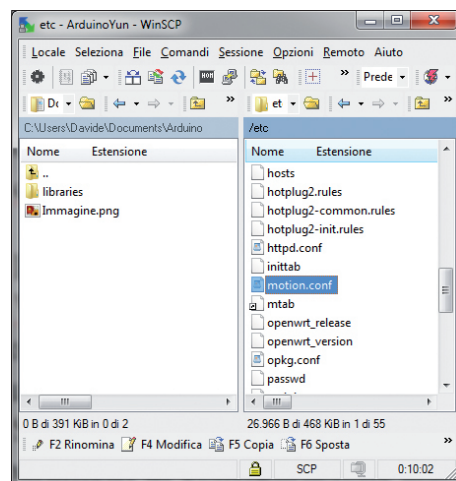
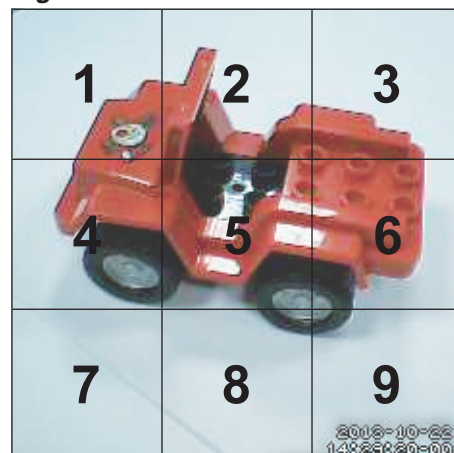


Fig. 19

Fig. 20



analizzata tutta l’area coperta dalla webcam per individuare un eventuale movimento. Se vogliamo sottoporre a controllo solamente una porzione dell’immagine, lo possiamo fare impostando questo parametro. L’area è suddivisa in nove parti denominate come nella **Fig. 20**.

Per attivare il riconoscimento del movimento in aree determinate basta includere nel parametro i numeri corrispondenti alle zone da tenere sotto controllo, ovvia-

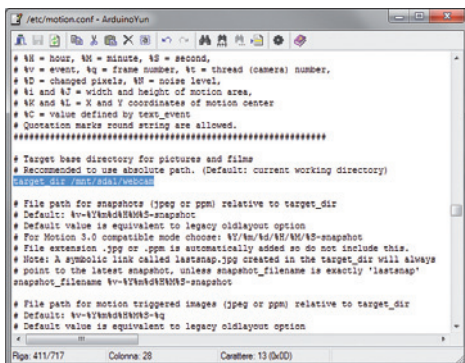


Fig. 21

configurazione:

target_dir /usr/local/apache2/htdocs/cam1

Questo parametro contiene la cartella predefinita nella quale vengono memorizzate tutte le immagini acquisite dalla telecamera. La configurazione predefinita prevede di salvare solo le immagini raccolte durante le intercettazioni di movimento, in modo da poter riconoscere l'intruso, a posteriori. Se lasciamo la cartella predefinita, le immagini saranno scritte nella memoria NAND di Arduino Yún.

Data la ridotta dimensione della memoria in pochi minuti le immagini memorizzate satureranno tutto lo spazio disponibile, mettendo in crisi anche il funzionamento generale del sistema operativo Linino.

Poco male, se capita, in genere basta cancellare i file di troppo e rilanciare il tutto. Nella peggiore delle ipotesi si ripristina il sistema originale con il pulsante di reset "WiFi reset button" tenuto premuto per più di 30 secondi. Per evitare qualsiasi inconveniente modifichiamo la configurazione predefinita con la seguente (Fig. 21):

target_dir /mnt/sd1/webcam

Il parametro:

stream_localhost on

permette di limitare la visione dello streaming della webcam ai soli utenti locali, non è il nostro caso in quanto noi ci colleghiamo con il browser del nostro PC. Quindi abilitiamo la connettività dall'esterno modificando il parametro di configurazione in (Fig. 22):

stream_localhost off

Allo stesso modo modifichiamo il parametro che restringe agli utenti locali l'utilizzo dello strumento di configurazione via web del server motion, modificandolo da:

webcontrol_localhost on

a (Fig. 23):

webcontrol_localhost off

L'ultima configurazione serve a fare eseguire il programma "allarme.py" quando viene rilevato un movimento nell'area controllata dalla webcam (Fig. 24).

on_motion_detected "python /mnt/sd1/pgm/allarme.py"

Ora occupiamoci di predisporre i programmi per la gestione dell'evento che si verifica quando viene intercettato un movimento nella zona del campo di visione della telecamera configurato

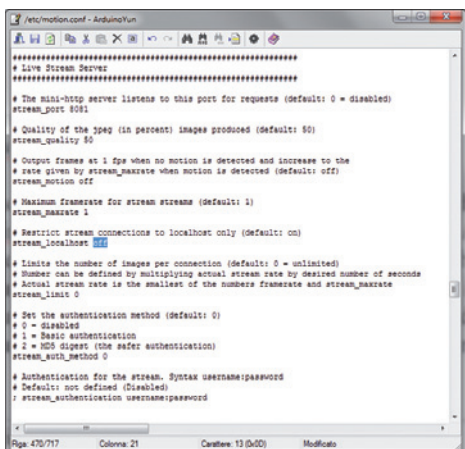


Fig. 22

mente dopo avere tolto il ";" di commento. Ad esempio, per individuare un movimento nella fascia centrale dell'immagine, dall'alto in basso, si utilizza la configurazione:

area_detect 258

Per il momento lasciamo le cose come stanno in modo che tutta l'area inquadrata dalla telecamera sia analizzata al fine di riconoscere un movimento.

Andiamo invece al parametro di

Fig. 23

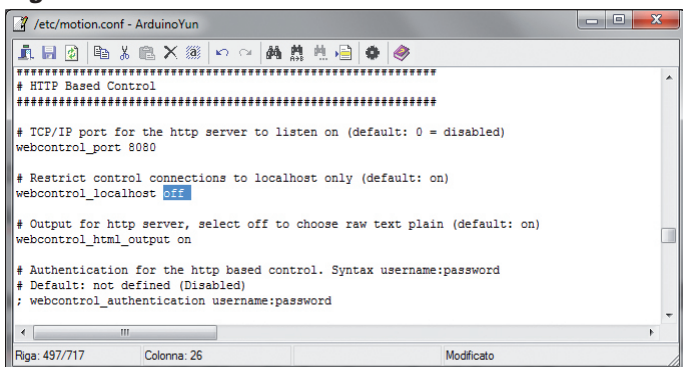
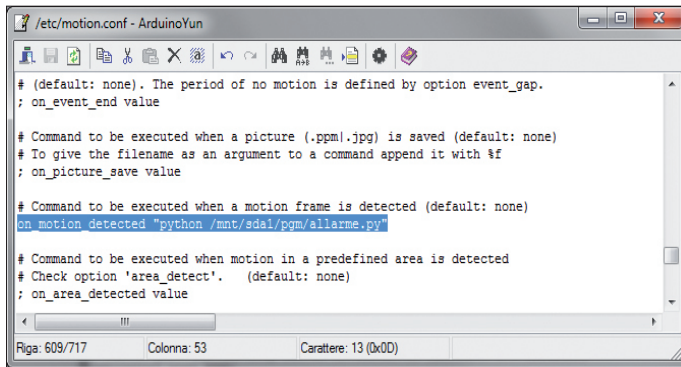


Fig. 24



Listato 1

```
#!/usr/bin/python

import os
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage

# Importa la libreria per eseguire comandi GNU/Linux da pgm
# Importa la libreria per inviare mail
# Importa moduli gestione mail con attachment
# Importa moduli per attachment di tipo testo
# Importa moduli per attachment immagini

# Apre il file allarme.txt in sovrascrittura (w+) scrive il valore 1 e richiude
fileout = open("/mnt/sda1/pgm/allarme.txt", "w+")
fileout.write("1");
fileout.close()

# Istruzioni per cercare l'ultimo fotogramma memorizzato nella cartella /mnt/sda1/webcam
# da inviare come allegato alla mail
cartella = "/mnt/sda1/webcam" # Importa la libreria per eseguire comandi GNU/Linux da pgm

# Si posiziona sulla cartella delle immagini
os.chdir(cartella)

# Cerca il fotogramma piu recente
filelist = os.listdir(os.getcwd())
filelist = filter(lambda x: not os.path.isdir(x), filelist)
# Memorizza il nome del fotogramma piu recente per successivo attachment
attachment = max(filelist, key=lambda x: os.stat(x).st_mtime)

# Imposta le intestazioni del messaggio di posta elettronica
to_addr = "<Indirizzo_mail_mail_di_destinazione>"
from_addr = "<Indirizzo_mail_mail_del_mittente>"
# Titolo della mail
subject_header = "Allarme"
# Testo della mail
body = "Messaggio di allarme"

# Instanzia oggetto messaggio come MIMEMultipart
msg = MIMEMultipart()
# Inizia a comporre il messaggio con la intestazione
msg["To"] = to_addr
msg["From"] = from_addr
msg["Subject"] = subject_header
# Aggiunge il testo della mail
msgText = MIMEText('<plaintext>%s' % body, 'html')
msg.attach(msgText)
# Aggiunge l'immagine come allegato
fp = open(attachment, 'rb')
img = MIMEImage(fp.read(), name = "immagine.jpg")
fp.close()
msg.attach(img)
# Inizio invia della mail
# Imposta server e porta SMTP del provider di posta
emailRezi = smtplib.SMTP("<server_email>", <porta_email>)
emailRezi.set_debuglevel(0)
# Imposta utente e password dello utente mail
emailRezi.login("<utente_mail>", "<password_mail>")
# Invia la mail
emailRezi.sendmail(from_addr, to_addr, msg.as_string())
# Chiude il collegamento con il server di mail
emailRezi.quit()
```

come zona "sensibile". Abbiamo appena configurato il parametro `on_motion_detected` in modo da eseguire un programma python quando viene innescato un evento.

Per scrivere il programma utilizziamo sempre WinSCP, andiamo nella cartella `/mnt/sda1` e creiamo una nuova cartella "pgm". Per farlo clicchiamo col

tasto destro del mouse e selezioniamo "Nuovo">>"Cartella". Digitiamo il nome "pgm" e salviamo, aspettiamo quindi che ci venga chiesta la password e rispondiamo sempre con "arduino". Entriamo in questa cartella e creiamo un file di nome "allarme.py", sempre cliccando col tasto destro del mouse e scegliendo "Nuovo">>"File". Lo salviamo.

Poi copiamo nell'editor del programma le righe contenute nel **Listato 1**. I listati dei programmi sono comunque sempre scaricabili dal sito www.elettronica.in.it. Salviamo e chiudiamo. Ora apriamo un altro file di nome "allarme.txt", inseriamo il valore "0" (senza apici), salviamo e chiudiamo anche questo file. Infine creiamo un ultimo pro-

Listato 2

```
#!/usr/bin/python

fileout = open("/mnt/sdal/pgm/allarme.txt", "w+")
fileout.write( "0");
fileout.close()
```

gramma di nome "allarme_off.py", che verrà richiamato dallo sketch di Arduino per azzerare il contenuto del file di allarme. Copiamo le istruzioni del **Listato 2**. Tanto che ci siamo creiamo anche la cartella "webcam" come sotto-cartella di /mnt/sdal/ per accogliere i fotogrammi risultato della registrazione del movimento. Il primo programma è il più complesso, soprattutto in considerazione della logica "classica" di Arduino. Infatti ci permette, oltre ad "accendere il semaforo" di intercettazione di un movimento -azione che si concretizza nel portare al valore "1" il contenuto del file "allarme.txt"- di inviare una email al nostro indirizzo di posta elettronica, con la notifica dell'evento ed allegato l'ultimo fotogramma scattato durante la rilevazione del movimento. Una brevissima introduzione al linguaggio Python prima di descrivere i programmi. Python è un linguaggio interpretato e quindi i programmi non hanno necessità di essere compilati. Anche la struttura dei programmi è molto semplice. La specifica di moduli e gruppi di istruzioni avviene in base all'indentazione delle istruzioni. Non vi sono parentesi graffe di apertura e chiusura di blocchi di istruzioni e nemmeno ";" di terminazione delle istruzioni. Ciononostante è un linguaggio molto potente, che sta godendo di una diffusione sempre maggiore e permette di realizzare applicazioni dalle prestazioni di tutto rispetto. Per quanto riguarda la logica dei programmi, abbiamo inserito commenti pressoché ad ogni riga di codice. Come potete notare una grossa semplificazione viene

Listato 3

```
/*
Sketch gestione notifica movimenti da motion
*/
#include <FileIO.h>
#include <Process.h>

int led_rosso = 4; // LED di avviso di motion detection
int buttonPin = 5; // Pulsante reset LED rosso
char buttonState = 0;
String stato = "";

void setup() {
  // Setup Bridge richiama bridge.py su Linino
  Bridge.begin();

  pinMode(led_rosso, OUTPUT); // Imposta pin 5 come OUTPUT

  // Setup File IO richiama files.py su Linino
  FileSystem.begin();
}

void loop() {

  // Istanza oggetto file e Open del file in lettura
  File nome = FileSystem.open("/mnt/sdal/pgm/allarme.txt", FILE_READ);
  stato = ""; // Lettura di un carattere dal file
  stato += (char)nome.read();
  nome.close(); // chiude il file allarme.txt

  if (stato == "1") { // Se stato 1 e stato rilevato un movimento
    digitalWrite(led_rosso, HIGH); // accende il LED rosso di avviso
    Process zero_alarm; // istanzia oggetto Process
    zero_alarm.begin("python"); // Comando che si richiede di eseguire a Linino
    zero_alarm.addParameter("/mnt/sdal/pgm/allarme_off.py"); // Parametri comando
    zero_alarm.run(); // Richiesta di esecuzione del comando
  }

  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW) { // Se viene premuto il pulsante
    digitalWrite(led_rosso, LOW); // viene spento il LED rosso
  }
  delay(2000); // Ritardo di 2 secondi ad ogni ciclo
}

Ora possiamo provare il tutto.
```

dall'utilizzo di librerie specializzate che offrono classi, oggetti e metodi per lo svolgimento delle funzioni principali del programma. Nel nostro caso abbiamo utilizzato librerie già incluse nella distribuzione Linino e quindi immediatamente disponibili. Passiamo ora alla realizzazione dello sketch per Arduino. Lo sketch utilizza la libreria "Bridge" ed i metodi disponibili per la gestione dei file sulla SDCard. Lo schema logico dello sketch, visibile nel **Listato 3**, prevede i seguenti passi. All'inizio del programma si includono le librerie che espongono gli oggetti che utilizzeremo nel programma, ovvero la libreria Bridge per la gestione del "ponte"

di comunicazione e la libreria FileIO per la gestione dei file residenti nel file system di Linino. Definiamo il pin 4 del LED e lo configuriamo come output.



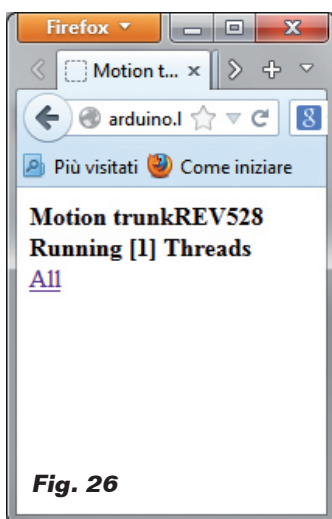


Fig. 26

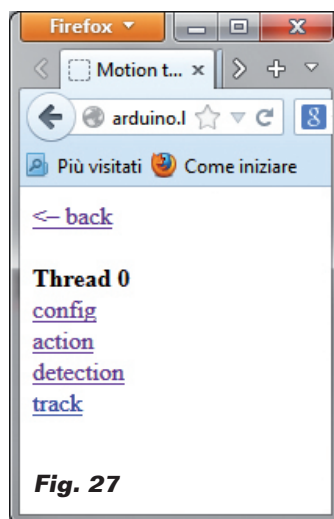


Fig. 27

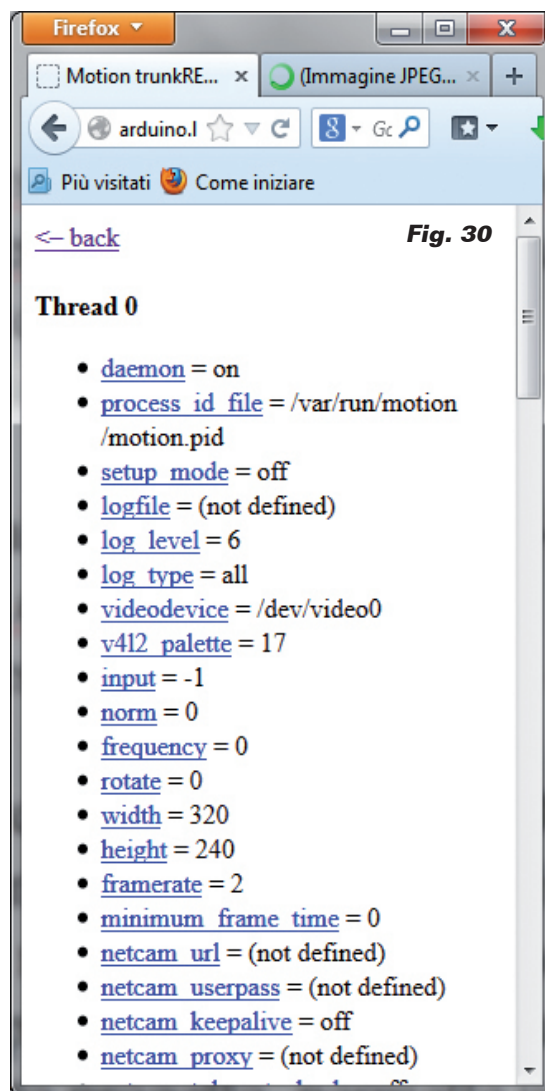


Fig. 30

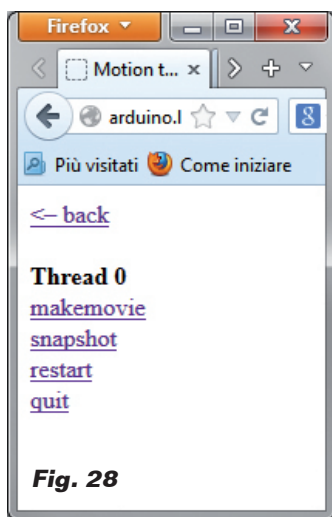


Fig. 28

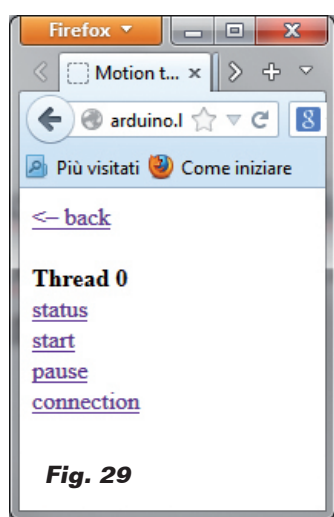


Fig. 29

Definiamo il pin 5 collegato al pulsante. Nella funzione *setup()* troviamo l'istruzione *Bridge.begin()*, che inizializza il "ponte" di comunicazione tra le due sezioni della scheda, di fatto richiama il caricamento in memoria, in ambiente Linino, dei programmi di gestione del ponte di comunicazione. L'istruzione *FileSystem.begin()* richiama in memoria in ambiente Linino i programmi che permettono agli sketch di Arduino di condividere il file system di Linino. Nel blocco *loop()* troviamo la logica principale del programma. Viene aperto il file "allarme.txt" in lettura e ne viene letto il contenuto. Da sottolineare che il metodo *read()* legge il con-

tenuto del file un byte alla volta. Nel nostro caso non utilizziamo un loop per lo scorrimento di tutto il file, in quanto abbiamo previsto una lunghezza del file di un solo byte. Chiudiamo subito il file per non interferire con un'eventuale richiesta di scrittura da parte del server "motion". Segue il test del contenuto del file. Un valore "1" significa che è stato intercettato un movimento dal server "motion". In questo caso si accende il LED rosso e si procede a riportare al valore "0" il contenuto del file "allarme.txt". In questo caso abbiamo scelto la via per passare a Linino un comando che richieda l'esecuzione, direttamente in ambiente Linino, del programma python "allar-

me_off.py", che scrive "0" nel file "allarme.txt". I motivi sono, da una parte, presentare nello stesso articolo una funzionalità delle più importanti di Bridge e dall'altra quella di utilizzare una modalità di azzeramento del file che interferisca il meno possibile con il funzionamento generale della nostra applicazione. Ricordiamo che il server "motion" sta sempre in agguato per intercettare un movimento e notificarlo scrivendo il file "allarme.txt". Queste condizioni di potenziale conflitto devono essere tenute in considerazione durante la progettazione delle applicazioni embedded in quanto fonte di possibili errori saltuari e maligni, difficili da individuare e rimuov-

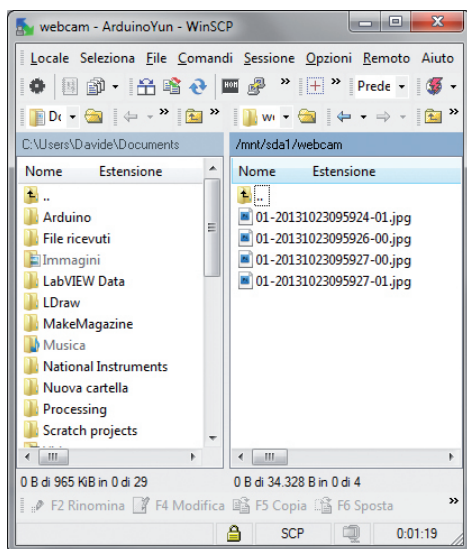


Fig. 31

vere. Il metodo utilizzato è di instanziare un oggetto "Process" che permette di predisporre un comando da eseguire in ambiente Linino. L'istruzione

```
zero_alarm.begin("python");
```

predisporre il comando da eseguire, cioè l'esecuzione dell'interprete Python. La successiva aggiunge i parametri al comando, nel nostro caso il programma da eseguire. L'istruzione

```
zero_alarm.run()
```

mette in esecuzione il comando in ambiente Linino. Un ritardo di due secondi permette di non soffocare troppo il "ponte" di comunicazione tra Arduino e Linino.

Ora possiamo provare il tutto. Dalla finestra di terminale di Putty, lanciamo il programma motion con il comando:

motion

A questo punto apriamo un browser e digitiamo l'indirizzo <http://arduino.localhost:8081> dovremmo vedere all'interno del browser l'immagine ripresa dalla telecamera (Fig. 25).

In una finestra separata digitiamo l'url:

<http://arduino.localhost:8080>

in modo da poter accedere all'applicazione essenziale di gestione di motion.

Otteniamo una pagina con il contenuto di Fig. 26.

Clicchiamo su "All" e otteniamo il menu di Fig. 27.

Cominciamo da action che ci presenta il menu di Fig. 28.

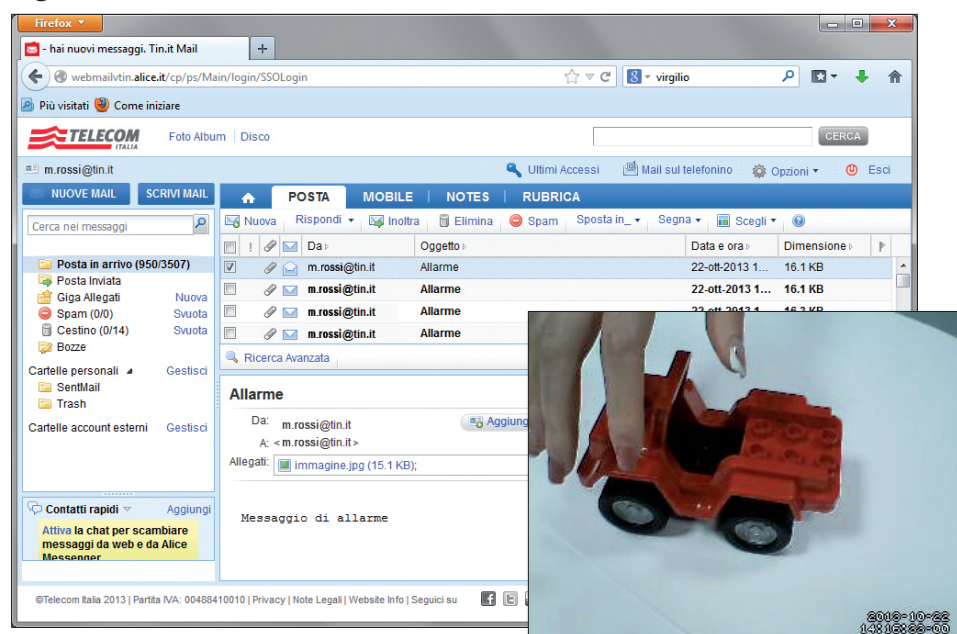
Con "quit" possiamo interrompere il funzionamento del server, con "restart" lo inizializziamo nuovamente, per esempio dopo avere modificato dei parametri di configurazione. "Makemovie" registra le immagini riprese dalla telecamera mentre "Snapshot" salva il fotogramma corrente.

Il menu "detection" (Fig. 29) ci consente di gestire l'attività di riconoscimento del movimento, mentre con "pause" e "start" possiamo disattivarla e riattivarla. Infine "config", con la funzione "list" (Fig. 30), ci permette di personalizzare i parametri on-line, se le salviamo ("write"), le modifiche verranno registrate nel file motion.conf, che abbiamo precedentemente modificato a

mano. "get" e "set" ci permettono di gestire singoli parametri del file di configurazione.

Ora il test. Facciamo intercettare un movimento dalla telecamera, per esempio passandoci davanti con una mano, vedremo accendersi il LED rosso e nella cartella /mnt/sda1/webcam (Fig. 31) troveremo una sequenza di immagini che immortalano ciò che è successo. Apriamo la casella di posta elettronica e troveremo il nostro messaggio di notifica con allegata l'immagine intercettata (Fig. 32). Ora che abbiamo preso visione dell'accaduto possiamo resettare il tutto premendo il pulsante sulla breadboard, spegnendo il LED rosso. Questo progetto, permette una varietà enorme di sperimentazione, con la variazione dei parametri di motion e la gestione dell'evento mediante lo sketch di Arduino. Potete sperimentare senza problemi: nel caso di situazioni "irreparabili", anche se dalla nostra esperienza abbastanza difficili da raggiungere, è sempre possibile "ripassare dal via" con il pulsante di reset. ■

Fig. 32





Pubblichiamo la proposta che ha vinto il concorso iniziato nel fascicolo n° 178: si tratta di una revisione del Dotklok che implementa nuove funzionalità come la lettura della temperatura ambiente e il controllo del livello di luminosità del display sulla base della luce presente nell'ambiente. Il tutto può essere comandato a distanza tramite un comune telecomando TV.



AGGIUNGI UNA FUNZIONE IN PIÙ AL TUO DOTKLOK: LA SOLUZIONE

..... a cura della REDAZIONE

Come sapete, il concorso "Aggiungi una funzione in più al tuo DOTKLOK" si è concluso a fine ottobre e nelle scorse settimane abbiamo vagliato con attenzione le soluzioni giunte in Redazione. Diciamo subito che ci aspettavamo più firmware e meno teoria, ma nonostante ciò qualche interessante proposta - degna di pubblicazione - è arrivata. Tra queste, quella che ci è sembrata

la migliore e che si è aggiudicata il premio in palio, ossia una stampante 3D, è quella del Sig. Stefano Del Bufalo.

Come promesso, dedichiamo qualche pagina all'esposizione della soluzione vincente, per "snocciolare" gli aspetti che la caratterizzano e che possono certamente rappresentare, per tutti i nostri lettori che lavorano con Arduino, interes-

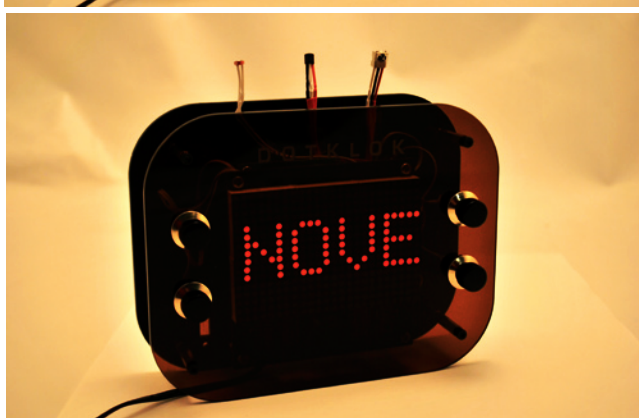


Fig. 1

Modalità di visualizzazione dell'orario sequenziale in parole.

santi spunti da cui partire per sviluppare ulteriori applicazioni. Il soggetto lo conoscete dalla pubblicazione dell'articolo nel fascicolo n° 175: è il celebre Dotklok, l'orologio hold fashioned e un po' vintage, con il display a matrice di LED rossi o verdi che richiama le sveglie di trent'anni fa; oltre che essere un orologio capace di segnare il tempo in vari modi, questo progetto consente di realizzare animazioni grazie, proprio, alla matrice di LED.

A rendere ancora più interessante il dispositivo, c'è il fatto che nasce secondo la filosofia open source, cosa che non sorprende, visto che il suo creatore, Andrew O'Malley, ha utilizzato come sistema di programmazione per il micro che controlla tutte le funzioni l'IDE di Arduino; più esattamente, Dotklok è costruito con un display a matrice di LED da 16x24 diodi (disponibile in due colori: rosso e verde), gestito da un controller Holtek HT1632 ad interfaccia seriale. Il tutto è interfacciato ad una scheda di controllo basata su architettura Arduino con microcontrollore Atmel Atmega328P e Real Time Clock integrato incentrato sul classico integrato DS1307 della Dallas Semiconductor. Proprio il firmware e l'hardware open hanno spronato numerosi sperimentatori a costruire e personalizzare il proprio Dotklok, condividendo gli sketch da essi realizzati per aggiungere o cambiare le animazioni di cui Dotklok è capace. Questo è stato poi il senso del concorso che abbiamo lanciato e che proponeva di ideare e inviare varianti al funzionamento del Dotklok come sviluppato da O'Malley, così da incentivare uno stimolante scambio di idee. Modificare il firmware per aggiungere nuove funzionalità o modificarle non richiede solo pratica con il linguaggio di programmazione e con l'IDE di Arduino, ma anche la capacità di tenere da conto una limitazione di Arduino Uno, che nello specifico è la massima dimensione della Flash a disposizione per il firmware; infatti, ogni eventuale aggiunta richiede spazio, e il firmware originale non ne lascia moltissimo, quindi la modifica impone un lavoro di "limatura" che consiste nell'ottimizzare il firmware togliendo ciò che non serve e aggiungendo il possibile.

LA SOLUZIONE

Ma entriamo nel vivo del progetto vincente segnalando come l'autore abbia proposto due interessanti modifiche: una apporta nuovo hardware e l'altra propone il software che per-

mette di sfruttarle.

Per quanto riguarda l'hardware, Stefano ha implementato l'aggiunta di due sensori:

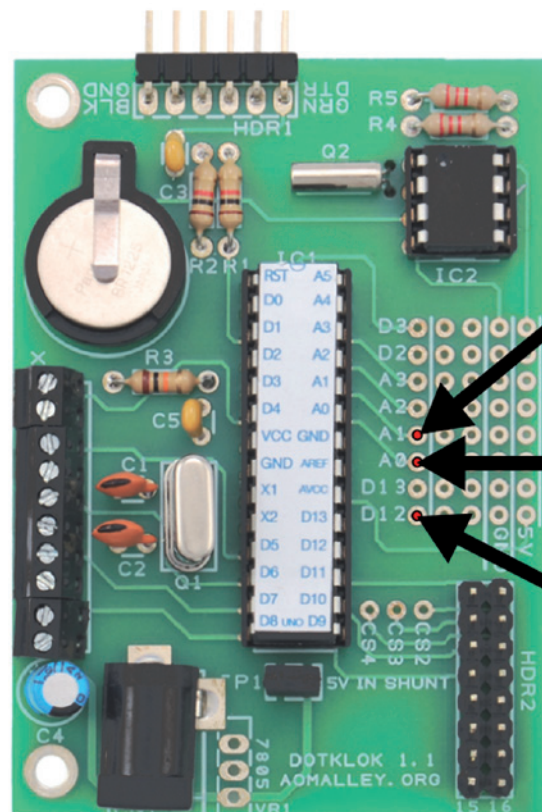
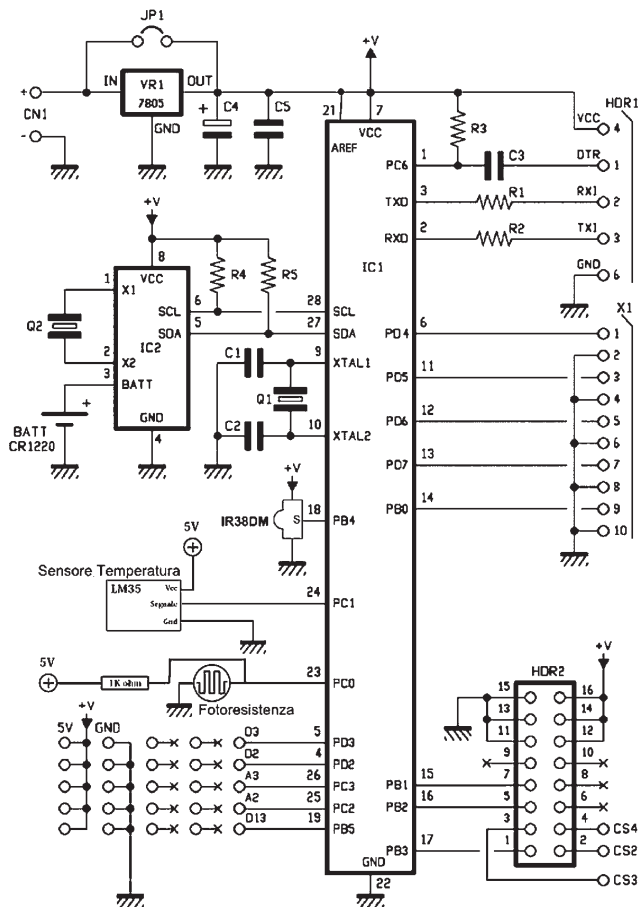
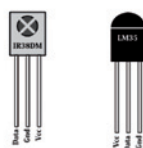
- uno di temperatura;
- uno di luce.

Inoltre, è stato previsto anche un ricevitore infrarosso standard a 38 kHz, adatto a ricevere comandi dai classici telecomandi IR dei televisori, DVD, videoregistratori, decoder TV ecc. Questo permette di inviare comandi a distanza a Dotklok, che ognuno potrà scrivere a piacere nel firmware; infatti, la soluzione proposta contiene l'implementazione del codice per leggere il telecomando usato dal vincitore, ma ognuno potrà sostituire il codice con quello adatto al proprio. Oltre a queste funzionalità, il Dotklok presenta modifiche firmware che cambiano la modalità di rappresentazione dell'orario, implementando, ad esempio, quella visibile in Fig. 1. Con le modifiche apportate, il firmware occupa "appena" 27 kb, quindi entra senza problemi nella memoria di programma del micro, ampia 32 kb.

Per quanto riguarda i sensori, quello di temperatura permette, mediante un'opportuna modifica dello sketch (che trovate nel nostro sito www.elettronica.in.it), di rappresentare sul display la temperatura ambiente come se fosse una delle numerose animazioni già presenti in Dotklok. La sonda di temperatura scelta è la classica e collaudatissima LM35 di Texas Instruments, che fornisce un segnale analogico consistente in una tensione continua di 10 mV per ogni grado centigrado rilevato; l'LM35 fornisce il valore di temperatura con un'accuratezza di 0,5 °C in un campo di misura compreso tra -55 e +150 °C. L'integrato ha tre piedini, due dei quali sono per l'alimentazione (4÷30 Vcc) mentre quello centrale fornisce la tensione analogica che esprime la temperatura rilevata. Siccome l'LM35 fornisce 10mV/°C a partire dallo 0 K, 0 gradi centigradi, corrispondenti a 273,15 K, determinano in uscita 2,7315 volt.

E passiamo al secondo sensore, che permetterà di attivare o escludere la regolazione automatica della luminosità del display in base alla luce ambientale, per evitare, ad esempio, in situazione di poca luce nell'ambiente, che i LED possano risultare troppo fastidiosi, o al contrario

Connessioni del ricevitore IR e del sensore di temperatura



Uscita sensore di temperatura; connettere anche i terminali Vcc e GND del chip alle relative piazzole della board.

Fotoreistenza e resistore di pull-up da 1 kohm.

Segnale del ricevitore IR avendo cura di collegare anche Vcc e GND ai relativi piedini.

Fig. 2
I sensori aggiunti
al Dotklok.



che in caso di forte luce, gli stessi siano poco visibili. La lettura della sonda di temperatura e quella della fotoresistenza vengono realizzate applicando le tensioni corrispondenti agli ingressi analogici di Arduino; in particolare, la fotoresistenza che presenta un valore in piena oscurità di circa 20 kohm, viene collegato in serie ad una resistenza fissa da 1 kohm, in modo da formare un partitore di tensione alimentato dal +5V. Affinché la lettura avvenga correttamente, a questo potenziale va portato anche il pin Vref di Arduino.

IL TELECOMANDO

L'ultima variazione sul tema apportata dal vincitore del concorso è l'aggiunta del telecomando: implementata con poche righe di codice. Questa funzione permette di leggere i segnali inviati dai comuni telecomandi ad infrarossi impiegati in TV, decoder, ecc. Con questa modifica si possono rendere gran parte delle funzioni del Dotklok controllabili da remoto, tanto da poter sostituire i tasti del telecomando ai pulsanti già presenti.

Negli esempi di codice forniti dal Sig. Del Bufalo, il telecomando permette di passare da un'animazione ad un'altra stando comodamente seduti sul divano, oppure, sempre da remoto, disattivare il sensore di luminosità per impostare in modo manuale il grado di luminosità del display; infatti una volta introdotto un sensore e uno sketch che reagisce alla pressione di un qualsiasi pulsante di un qualsiasi telecomando, l'unico freno all'introduzione di nuove funzioni, è la fantasia. Infatti il bello di questa soluzione è che non serve un telecomando specifico, ma il sistema può apprendere i codici della gran parte dei telecomandi, purché funzionanti con portante dati di 38 kHz. Allo scopo, basta impostare il Dotklok nella modalità DEBUG, per vedere sul monitor seriale (Serial Monitor dell'IDE di Arduino) quali valori restituiscono i vari tasti di un qualsiasi telecomando, ed introdurre questi

ultimi valori nello sketch; vedremo tra breve come apprendere i codici direttamente dal display, invece che da Serial Monitor.

IL NUOVO FIRMWARE

Purtroppo per realizzare tutto questo, occorre rinunciare a qualche animazione per motivi di spazio, già precario, del microcontrollore AT-Mega328P; in questo caso l'autore ha rinunciato a "percent_time", "random_dot_time", "morse_time" e "game_time", che sono state rimosse dal firmware. Inoltre, chi volesse recuperare più spazio di quello liberato dall'autore, potrà seguire le istruzioni contenute nelle guide del prof. Michele Menniti sul suo sito (<http://www.michelemenniti.it/>), modificando opportunamente il file board.txt per la programmazione dell'ATMega Stand-Alone con il metodo ISP. Le funzioni aggiunte e quelle tolte hanno comportato una sostanziale modifica dello sketch, per quanto riguarda la parte di setup.

Lo sketch inviato dal Sig. Del Bufalo (scaricabile dal nostro sito web) è aggiornato per essere inserito con l'IDE 1.0.1 di Arduino e le relative librerie (la RTCLib attuale, ad esempio) supportano tutte le versioni dell'IDE.

La libreria aggiornata RTCLib contiene già il controllo della versione dell'IDE. Alla libreria **Button**, contenuta nel file *Button.h*, l'autore ha aggiunto le seguenti righe per verificare la versione dell'IDE:

```
#if (ARDUINO >= 100)
#include <Arduino.h>
```

Listato 1

```
//-----
//#### introduzione nuove funzionalita' (Stefano Del Bufalo) inizio
//-----
// Sensore LM35
int pin_lm35 = A1;

// Sensore IR Remote Control
int pin_IR = 12;
#include <IRremote.h>
IRrecv irrecv(pin_IR);
decode_results results;
int ir_mode = 0; //int ir_mode = 2 (test IR) - int ir_mode = 0
//((funzionamento normale)

int c;
boolean remote_control = false;

// Luminosita' da fotoresistenza
int pin_fotoresistenza = A0;
int fotoresistenza_val;
int lux_val = 10;
boolean fotoresistenza_stato = true;

//-----
//#### introduzione nuove funzionalita' (Stefano Del Bufalo) fine
//-----
```



```
#else
#include <WProgram.h>
#endif
```

La libreria **ht1632** invece, è rimasta quella originalmente pubblicata da noi per il Dotklok originale. Nel *Listato 1*, vedete la dichiarazione delle variabili per lo sketch modificato. Notate le righe:

```
// Sensore LM35
int pin_lm35 = A1;
```

che riguardano l'aggiunta del sensore di temperatura e:

```
// Sensore IR Remote Control
int pin_IR = 12;
```

che si riferiscono al telecomando IR. Per questa funzionalità, si carica la libreria *IRremote.h*:

```
#include <IRremote.h>
```

La libreria da importare per questo progetto è la IR Remote versione 0.11 e solo quella; altre potrebbero non funzionare. Per quanto riguarda il telecomando, l'abbinamento si effettua in questo modo: per prima cosa bisogna portare Dotklok in modalità 2 (test IR) modificando la riga `int ir_mode = 0;` nel Listato 1, sostituendo 2 a 0 fino a far diventare la riga `int ir_mode = 2;`. A questo punto, si punta il telecomando verso il sensore IR a 38 kHz, restando ad almeno mezzo metro di distanza e si trasmette usando i tasti che si desidera comandino le funzioni di Dotklok; grazie alla modifica alla funzione implementata dal Sig. Del Bufalo, il display di Dotklok mostra direttamente il numero corrispondente al codice del tasto premuto. Basta prendere nota dei numeri e poi andarli a sostituire nella funzione "press_remote", così il sistema riconoscerà i tasti che desiderate. Per provare IRremote, che è la funzione di lettura del telecomando IR, bisogna impostare la variabile `int ir_mode = 2;`, che trasforma il DOTKLOK in un tester per telecomandi; potete usare qualsiasi telecomando, ma se vedete passare troppi "0" (zero) alla pressione dei vari tasti, è possibile che esso non lavori proprio a 38 kHz, quindi il sistema non riesce a rilevarlo. Lo sketch fornito dall'autore contiene, nella

Listato 2

```
//-----
//##### introduzione nuove funzionalita' (Stefano Del Bufalo) inizio
//-----

// press_remote
// Verifica all'interno dei loop delle animazioni, se si e' inviato
un comando mediante telecomando boolean press_remote(){

//=====
// Ricezione segnale IR
//=====
if (irrecv.decode(&results)) {

String ir_val = String(results.value, HEX);
// legge il codice esadecimale letto dal sensore IR

if (ir_mode==0 && ir_val=="15") {
animation++;
if(animation > ani_max) animation = 1;
delay(50);
} else if (ir_mode==0 && ir_val=="16") {
animation--;
if(animation < 1) animation = ani_max;
delay(50);
} else if (ir_mode==1 && ir_val=="15") {
lux_val++;
fotoresistenza_stato = false;
ht1632_sendcmd(11, HT1632_CMD_PWM + lux_val);
delay(50);
} else if (ir_mode==1 && ir_val=="16") {
lux_val--;
fotoresistenza_stato = false;
ht1632_sendcmd(11, HT1632_CMD_PWM + lux_val);
delay(50);
}

if(ir_val=="30")
{ // Tasto Mode (il valore esadecimale dipende dal telecomando usato)
ir_mode++;
if(ir_mode > 2) { ir_mode = 0; }
plot(ir_mode,0,1);
delay(1000);
} else if(ir_val=="12")
{ // Tasto on/off luminosita' automatica
// il valore esadecimale dipende dal telecomando usato
if (fotoresistenza_stato) {
fotoresistenza_stato = false;
plot(0,13,1);plot(1,13,1);plot(2,13,1);
} else {
fotoresistenza_stato = true;
plot(1,12,1);
plot(0,13,1);plot(1,13,1);plot(2,13,1);
plot(1,14,1);
}
delay(1000);
} else if(ir_val=="11")
{ // Funzione 1 (il valore esadecimale dipende dal telecomando usato)
// test Funzione 3
plot(1,15,1);
delay(50);
} else if(ir_val=="10")
{ // Funzione 2 (il valore esadecimale dipende dal telecomando usato)
// test Funzione 4
plot(2,15,1);
delay(50);
} else if(ir_mode==2)
{ //(Modo 2 - debug per rappresentare a display i codici esadecimali
dei vari pulsanti di telecomandi sconosciuti)
// visualizza i codici esadecimali non usati
ht1632_clear();
int step = 0;

char char_print[10];
ir_val.toCharArray(char_print, sizeof(char_print));

scroll_text(char_print);

delay (500);
}

irrecv.resume(); // Pronto per il prossimo segnale

return true;
}

return false;
}

void luce_ambiente(boolean fotoresistenza_stato) {

if (fotoresistenza_stato) {
fotoresistenza_val = analogRead(pin_fotoresistenza);

lux_val = map(fotoresistenza_val, 1023, 700, 0, 15);
if (fotoresistenza_val<700) { lux_val = 0; }
ht1632_sendcmd(11, HT1632_CMD_PWM + lux_val);
}

}
//-----
//##### introduzione nuove funzionalita' (Stefano Del Bufalo) fine
//-----
```



Il super premio

Finirà nella capitale la stampante 3D in palio per questo concorso. Se l'è infatti aggiudicata il Sig. Stefano Del Bufalo di Roma con le modifiche hardware e software descritte in queste pagine. Ricordiamo che la 3Drag, questo il nome della nostra stampante 3D, utilizza la tecnologia FFF (Fused Filament Fabrication) e può operare sia con PLA che con ABS. Con una velocità tipica di 120 mm/sec, un piano di stampa riscaldato di 20 x 20 centimetri ed un'altezza massima di 20 cm, è sicuramente la stampante 3D della sua categoria col migliore rapporto prezzo/prestazioni. La stampante è disponibile presso Futura Elettronica sia in scatola di montaggio (cod. 3DRAG/K Euro 680.00 IVA compresa) che già montata (cod. 3DRAG/M Euro 860,00 IVA compresa) www.futurashop.it

funzione "press_remote", i codici esadecimali contenuti e restituiti da un telecomando in suo possesso.

Nell'esempio dell'autore, mediante il telecomando è possibile:

A) cambiare modalità 0,1 e 2 (ovverosia valore di ir_mode)

0. in modalità 0 (si accende per un secondo il primo led in alto a sinistra), con i tasti + e - si può cambiare animazione.

1. in modalità 1 (si accende per un secondo il secondo led in alto a sinistra), con i tasti + e - si può cambiare la luminosità del display.

2. in modalità 2 (si accende per un secondo il terzo led in alto a sinistra), si attiva il debug per il controllo remoto, infatti qualsiasi tasto si preme su di un qualsiasi telecomando viene visualizzato il codice esadecimale che restituisce.

B. Attivare o disattivare la luminosità automatica del display in base alla luce ambientale: premendo una volta la si disattiva, compare un meno (-) in basso a sinistra, premendo di

nuovo la si attiva e compare un più (+) in basso a sinistra. Da tener presente che usando la modalità 1, questa funzione viene disattivata automaticamente.

La parte di firmware che gestisce il telecomando IR è riportata nel *Listato 2*. I numeri corrispondenti ai tasti del trasmettitore che avete ricavato durante la prova in modalità test, dovete scriverli nelle porzioni di codice if (ir_mode==0 && ir_val=="15"). Il numero che segue _val== va sostituito per i vari tasti con i codici corrispondenti all'animazione che volete comandare. Nel *Listato 2*, la porzione di codice seguente:

```
if (ir_mode==0 && ir_val=="15") {
    animation++;
    if(animation > ani_max) animation = 1;
    delay(50);
} else if (ir_mode==0 && ir_val=="16") {
    animation--;
    if(animation < 1) animation = ani_max;
    delay(50);
```



Fig. 3 - Visualizzazione dell'ora, della temperatura con la versione modificata di Dotklok.

definisce i codici dei tasti assegnati allo scorrimento in avanti e indietro nelle animazioni (modo 0): la pressione del tasto identificato dal codice 15 avanza e quella del tasto contraddistinto dal 16 fa tornare all'animazione precedente. Se i tasti che avete scelto corrispondono ad altri numeri, inserite questi al posto di 15 e 16.

Sempre nel *Listato 2*, la porzione di codice:

```

} else if (ir_mode==1 && ir_val=="15") {
    lux_val++;
    fotoresistenza_stato = false;
    ht1632_sendcmd(11, HT1632_CMD_PWM
+ lux_val);
    delay(50);
} else if (ir_mode==1 && ir_val=="16") {
    lux_val--;
    fotoresistenza_stato = false;
    ht1632_sendcmd(11, HT1632_CMD_PWM
+ lux_val);
    delay(50);

```

definisce la funzione dei soliti tasti 15 e 16 nel caso il Dotklok sia in modo 1 (regolazione luminosità). Valgono le considerazioni appena fatte per il modo 0.

La chiamata alle varie funzioni interessate al nuovo firmware è visibile nel *Listato 3*, che riguarda il setup del sistema; qui trovate una chiamata a vuoto nel `setup()` "`analogRead(pin_lm35);`" che serve alla gestione della fotoresistenza.

Notate che attualmente, visto che il partitore di cui fa parte il fotoresistore è collegato al +5V, il riferimento per l'ADC dell'ATmega328 di Arduino è impostato anch'esso a 5 volt; per aumentare la precisione, si potrebbe usare "`analogReference(INTERNAL)`", che imposta AREF a 1,1 volt: così facendo, i 10 bit del convertitore porterebbero ad una risoluzione di circa 1 mV, ma in questo caso occorrerebbe ricalibrare i valori della fotoresistenza.

L'HARDWARE

Per ottenere le funzioni descritte è necessaria qualche modifica hardware al circuito di controllo di Dotklok presentato dallo schema elettrico che si trova a pagina 60 del fascicolo n° 175 di *Elettronica In*; questo schema va modificato come mostra quello che trovate in queste

Listato 3

```

/*****
* traditional Arduino sketch functions:
and loop.
*****/

void setup()
{
    // init the button inputs
    pinMode(4, INPUT);
    pinMode(5, INPUT);
    pinMode(6, INPUT);
    pinMode(7, INPUT);
    pinMode(8, INPUT);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
    digitalWrite(8, HIGH);

    // debug mode forced by B4 pressed during start up
    if(!digitalRead(7)) DEBUG=true;

    if(DEBUG){
        Serial.begin(57600);
        Serial.print("DOTKLOK "); Serial.println(REV);
        Serial.println("DEBUG true, serial port open at 57600");
        //Serial.print("Avail mem = ");
        //Serial.println(availableMemory());
    }
    else{
        Serial.begin(57600);
        Serial.print("DOTKLOK "); Serial.println(REV);
        Serial.println("DEBUG false, serial port now closed");
        Serial.end();
    }

    // display screen setup, software version display, (and test)
    ht1632_setup();
    ht1632_clear();
    putchar_3x5(6, 5, REV[0]);
    putchar_3x5(12, 5, REV[2]);
    putchar_3x5(18, 5, REV[4]);
    plot(10, 9, 1);
    plot(16, 9, 1);
    delay(1000);
    if(DEBUG) screenTest();

    // general set up
    //pinMode(13, OUTPUT);
    randomSeed(analogRead(0));

    // RTC setup
    Wire.begin();
    RTC.begin();

    if (!RTC.isrunning()) {
        if(DEBUG) Serial.println("RTC is NOT running!");
        // following line sets the RTC to the date & time this sketch was compiled
        RTC.adjust(DateTime(__DATE__, __TIME__));
    }
    else {
        if(DEBUG) Serial.println("RTC is running.");
    }

    // test buttons
    /*
    while( b5.isPressed() ){
        for(int i=0; i<4; i++){
            if( buttons[i].isPressed() ) plot(i,0,1);
            else plot(i,0,0);
        }
    } */

    //-----
    //#### introduzione nuove funzionalita' (Stefano Del Bufalo) inizio
    //-----
    // Avvio ricevitore IR
    irrecv.enableIRIn();
    // settaggio pin fotoresistenza
    pinMode( pin_fotoresistenza, INPUT);
    analogRead(pin_lm35);
    //-----
    //#### introduzione nuove funzionalita' (Stefano Del Bufalo) fine
    //-----

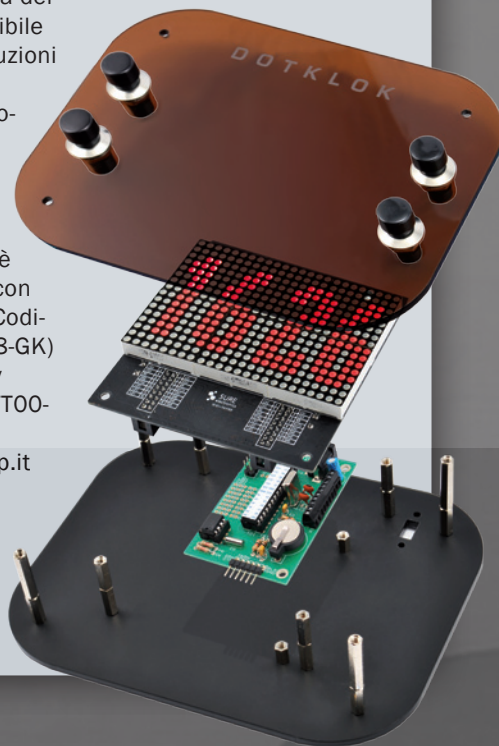
} // end setup()

void loop()
{
    // all time and button checking handled in animation functions
    // main loop just handles animation number (init. to 1)

```


Il progetto Dotklok

Il progetto Dotklok è stato presentato sui fascicoli 175 e 176: un originale orologio Open Source basato su Arduino che permette di visualizzare su uno schermo a matrice da 32x16 pixel non solo l'ora, ma anche una serie di animazioni e soluzioni grafiche che rappresentano il tempo e il suo trascorrere. Grazie alla disponibilità dei sorgenti e alle spiegazioni fornite circa il funzionamento del codice, questa piattaforma è a disposizione di tutti per essere arricchita, modificata o radicalmente reinterpretata. Di questo progetto abbiamo realizzato un kit che dispone di numerose possibilità di visualizzazione dell'ora, anche in modalità grafica. Tuttavia, grazie alla completa programmabilità del circuito, è possibile aggiungere soluzioni personalizzate come quella proposta dal Sig. Del Bufalo. Ricordiamo che il kit costa 78,00 Euro ed è disponibile sia con display verde (Codice 7100-FT1048-GK) che con display rosso (Codice FT00-FT1048-RK). www.futurashop.it



pagine, ossia aggiungendo la fotoresistenza con la relativa resistenza da 1 kohm al positivo dei 5 volt, e la sonda LM35.

A proposito di quest'ultima, va detto che in realtà, fornendo in uscita una tensione continua ed essendo alimentata dalla stessa linea a 5 V che alimenta la matrice di LED, in qualche modo il segnale di temperatura potrebbe essere affetto da lievi fluttuazioni dovute all'assorbimento dalla linea a 5 volt. Si potrebbe ovviare a questo inconveniente, pur poco rilevante, impiegando un sensore (a uscita digitale) tipo DHT11 o DHT22 collegato ad un pin digitale piuttosto che analogico, ma ciò implicherebbe la scrittura di codice per acquisire i dati dalla nuova sonda e la rimozione di quello per la lettura analogica. Vedete un po' voi.

Continuazione Listato 3

```
// bound the animation number
//animation = constrain(animation,1,ani_max);
//if(DEBUG) animation=6; // used to force any animation during
testing/debugging
if(DEBUG) { Serial.print("loop / animation: "); Serial.
println(animation); }

// run the clock animation
switch(animation) {
case 1:
big_time();
break;
case 2:
basic_date_time();
break;
case 3:
scroll_time();
break;
//-----
//#### introduzione nuove funzionalita' (Stefano Del Bufalo)
inizio
//-----
case 4:
temperatura_time();
break;
//-----
--
//#### introduzione nuove funzionalita' (Stefano Del Bufalo)
fine
} // end switch(animation)
} // end main loop

//-----
//#### introduzione nuove funzionalita' (Stefano Del Bufalo) > time_animations.h
//-----
// temperatura_time
void temperatura_time(){

long millis_now = 0;
long millis_temp = 0;
long gap_temp = 1000;

boolean power_up = true; // used to force time display on fun-
ction startup

/* INIT DISPLAY */
ht1632_clear();

/* TIME LOOP */
do{ // exit based on button presses, checked later in loop

//=====
// Lettura Sensore Temperatura
//=====
millis_now = millis();

if (millis_now - millis_temp > gap_temp )
{ // Refresh Sensore Temperatura ogni "gap" sec
millis_temp = millis_now;

float tempLM35 = (5.0 * analogRead(pin_lm35) * 100.0) / 1024.0;
// (5.0 tensione di alimentazione)
int tempcalc = tempLM35 * 10.0;

String str_temp = String(tempcalc, DEC);
char char_print[3];
str_temp.toCharArray(char_print, sizeof(char_print));

int step = 0;
for (c = 0; c < 3; c++){
// tre cifre in tutto
putchar_5x12(1 + step, 2, char_print[c]);
step = step + 6;
}

plot(12, 13, 1);

plot(19, 1, 1);
plot(20, 1, 1);
plot(19, 2, 1);
plot(20, 2, 1);
}

/* CHECK BUTTONS, return if necessary */
if( change_animation() ) return;
if( press_remote() ) return;
luce_ambiente(fotoresistenza_stato);

}while(1);

} // end function temperatura_time
//-----
//#### introduzione nuove funzionalita' (Stefano Del Bufalo)
//-----
```

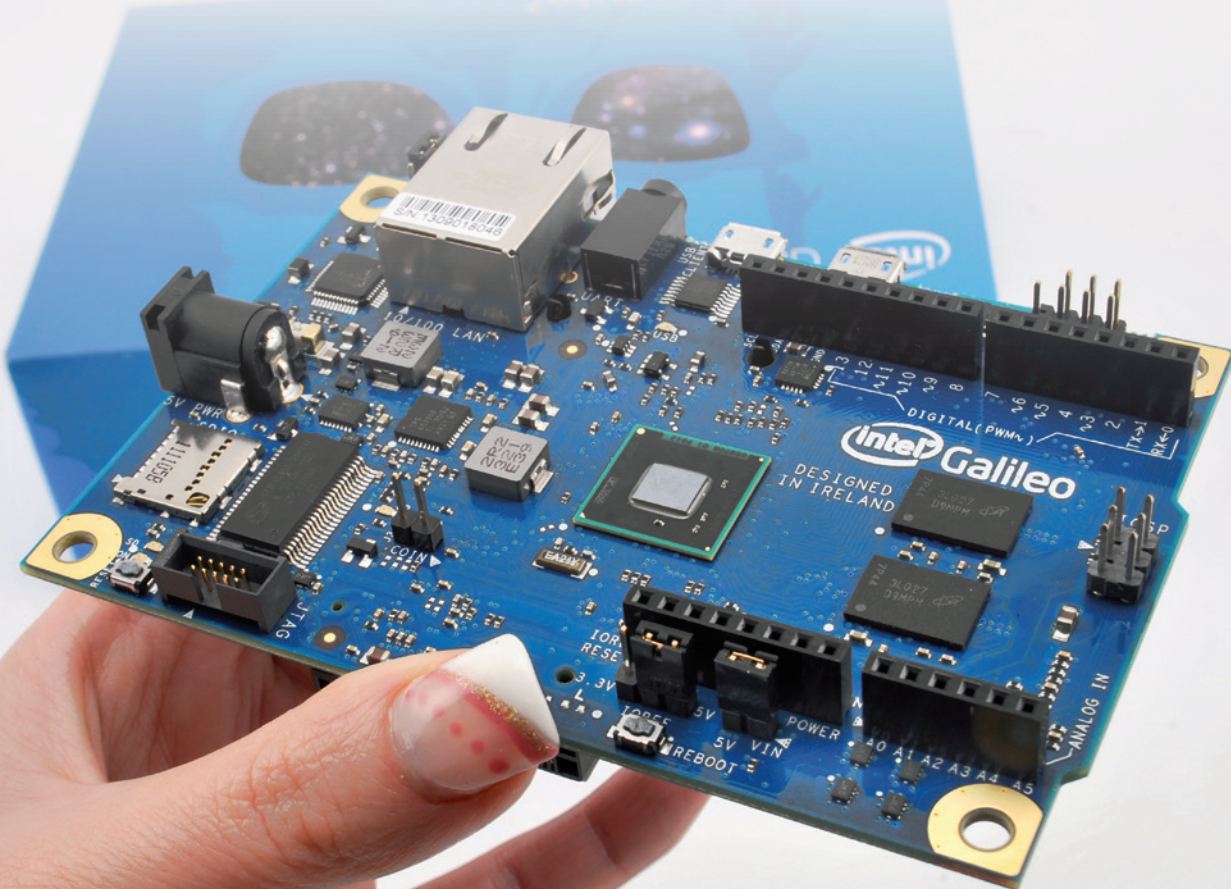
INTEL GALILEO

Abbiamo appena finito di presentare Arduino Yún, che già si affaccia una nuova board, sempre Arduino compatibile, sempre con GNU/Linux a bordo ma questa volta basata su tecnologia x86. Infatti a realizzarla è Intel il leader incontrastato di questa tecnologia.

..... di MARCO MAGAGNIN

L'evoluzione delle tecnologie embedded è un fatto conclamato se anche un gigante come Intel ha deciso di parteciparvi con la sua "interpretazione" del fenomeno. La tecnologia embedded unisce la potenza e flessibilità di un sistema operativo come GNU/Linux alla possibilità di gestire delle periferiche di I/O secondo le specifiche tipiche dei microcontrollori, il tutto realizzato con architetture provviste di uno o più processori. "Galileo", così Intel ha battezzato la sua nuova creatura, è una scheda basata sul SoC Intel Quark SoC X1000, un sistema costruito attorno ad un processore di classe Pentium a 32 bit, single-core, single-thread, compa-

tibile con l'architettura ISA (Instruction Set Architecture) che opera con velocità fino a 400 MHz. Galileo è la prima board embedded in architettura Intel progettata per essere compatibile hardware e software con le specifiche di Arduino Uno R3. Di fatto un PC di dimensioni ridotte, anche se non ridottissime dove ci si può montare uno shield progettato per Arduino. La piedinatura, come descriveremo nelle specifiche tecniche, è compatibile con quella di Arduino Uno, ed i terminali di I/O possono funzionare sia a 5 che a 3,3V mediante il posizionamento di un ponticello, che attiva i convertitori di livello per ciascun pin. La tensione nativa è 3,3V, quella a



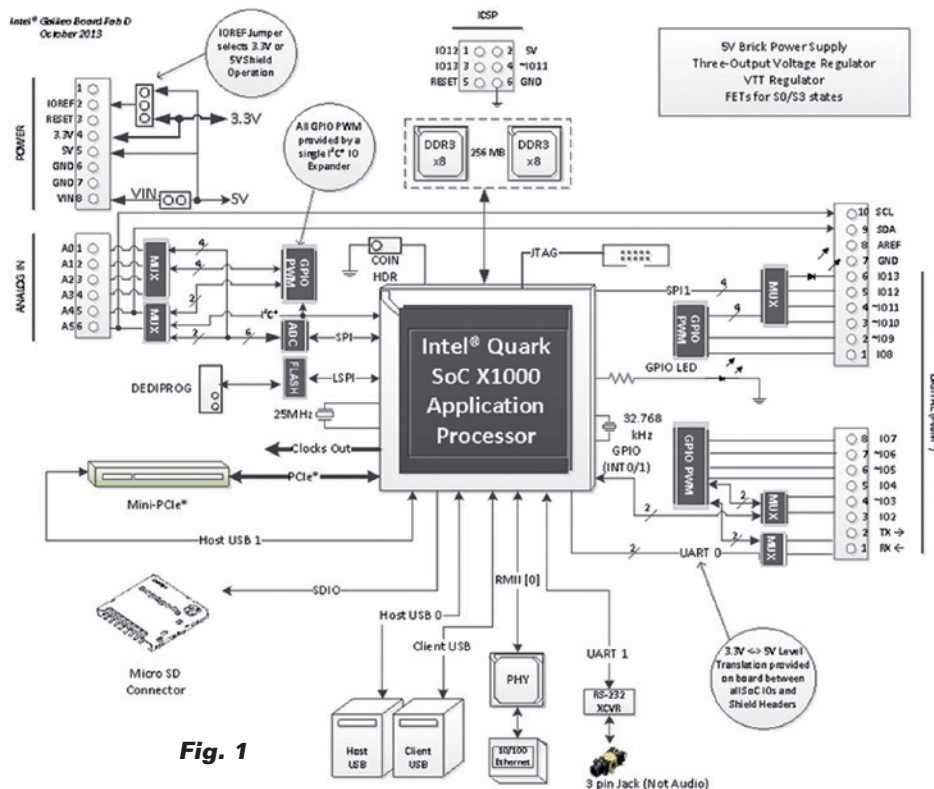


Fig. 1

5V è ottenuta con la conversione. La “Galileo” è compatibile a livello software con l’IDE di Arduino, facilitandone l’utilizzo come già per la famiglia Arduino. Il sistema operativo di “Galileo” è una distribuzione GNU/Linux, che “gira” sull’unico processore della scheda. Gli sketch di Arduino vengono eseguiti come processi nello spazio utente del sistema operativo GNU/Linux. L’IDE disponibile, provvede a compilare gli sketch in formato .elf, un formato binario eseguibile, originariamente sviluppato dagli UNIX System Laboratories e comunemente utilizzato nel mondo GNU/Linux.

CARATTERISTICHE TECNICHE

Data la sua discendenza da una famiglia di processori per PC, la scheda “Galileo” porta in dote tutta una serie di connettori, bus e porte, tipiche dello standard dei PC, ai quali si aggiungono le periferiche classiche di Arduino. Dando uno sguardo allo schema a blocchi della scheda, visibile

nella **Fig. 1**, possiamo riconoscere un connettore mini-PCI Express, una porta Ethernet a 100 Mb, un alloggiamento per una Micro-SD, una porta seriale RS-232, una porta USB Host, una porta USB Client.

Vi sono inoltre 8Mbyte di memoria flash NOR.

Galileo richiede un’alimentazione a 5V attraverso il connettore di alimentazione coassiale. A questo proposito, una raccomandazione ben evidenziata nelle istruzioni di Intel. È importante dare alimentazione alla scheda attraverso il connettore di alimentazione prima di collegarla al PC per la programmazione mediante il cavo USB, pena il danneggiamento della scheda stessa.

Vediamo le caratteristiche principali dell’architettura di Galileo

- Processore Intel Pentium 400MHz 32-bit in architettura ISA (instruction set architecture) con 16 KBytes di cache L1, 512 KBytes SRAM embedded;
- Real Time Clock (RTC) integrato: per tenerlo permanen-

temente attivo è sufficiente collegare agli appositi pin una batteria a bottone da 3V;

- Connettore Ethernet 10/100 Mb
- Uno slot PCI Express mini-card, in standard PCIe 2.0 in grado di accogliere le schede mini-PCIe a mezza altezza, eventualmente con un adattatore, lo slot è stato progettato espressamente per collegare una scheda WiFi da poter utilizzare con la libreria WiFi;
- Connettore Host USB 2.0 in grado di supportare sino a 128 dispositivi USB;
- Un connettore USB client, utilizzabile sia per il download degli sketch sulla scheda, che per collegare dispositivi compatibili USB 2.0;
- Connettore JTAG Standard a 10-pin per il debug;
- Pulsante per il reboot del processore;
- Pulsante per il reset del solo sketch Arduino;
- Memorie di massa – Una memoria da 8 MByte Legacy SPI Flash dove viene memorizzato il bootloader di GNU/Linux e l’ultimo sketch caricato. Allo sketch possono essere riservati da 256KByte a 512KByte di memoria. Una memoria SRAM da 512 Kbyte ed una memoria DRAM da 256 MByte gestite direttamente da sistema operativo;
- Una memoria EEPROM da 11 Kbyte che può essere programmata da sketch con la libreria EEPROM;
- Un alloggiamento per una micro SDCard fino a 32GByte di capacità;
- Connettore USB che può accogliere tutte le tipologie di memorie di massa compatibili con lo standard USB 2.0;
- Connettore jack da 3,5mm che permette di utilizzare una seconda porta seriale in standard

UART, da sottolineare che non si tratta di un ingresso/uscita audio.

Connettore compatibile Arduino

Vediamo più in dettaglio le periferiche compatibili con l'architettura Arduino:

- 14 pin che riproducono gli ingressi I/O di Arduino dei quali 6 possono essere utilizzati come uscite PWM (Pulse Width Modulation). Ciascuno dei 14 pin può essere configurato come ingresso o uscita utilizzando le classiche funzioni *pinMode()*, *digitalWrite()*, and *digitalRead()*. I pin sopportano i livelli sia a 5 che a 3,3V ed una corrente di 10mA in uscita e 25mA in ingresso. Possono essere configurate resistenze di pull-up da 5,6k a 10 kOhm., disattivate con configurazione predefinita;
- I pin di ingresso analogici, da A0 ad A5, sono collegati ad un integrato AD7298. Ciascun pin fornisce 12 bit di risoluzione (4096 valori tra 0 e 5V);
- I pin A4 ed A5, rispettivamente SDA ed SCL, supportano il protocollo di comunicazione

I2C (TWI) e possono essere interfacciati con la libreria Wire;

- Il bus SPI è configurato a 4MHz per compatibilità con le schede Arduino ma può essere portato a 25MHz. La scheda Galileo può operare unicamente come master nella comunicazione SPI;
- UART, la porta seriale è collegata ai pin 0 (RX) e 1 (TX);
- ICSP, connettore a 6 pin per la programmazione in-circuit, posizionato sulla scheda in modo da essere compatibile con gli shield esistenti;
- VIN - Il pin VIN permette di alimentare la scheda direttamente invece di utilizzare il connettore di alimentazione coassiale. L'avvertenza è di fornire una tensione stabilizzata a 5V, pena il danneggiamento della scheda;
- Pin di uscita a 5V che fornisce al massimo 800mA di corrente;
- Pin di uscita a 3,3V che fornisce al massimo 800mA di corrente;

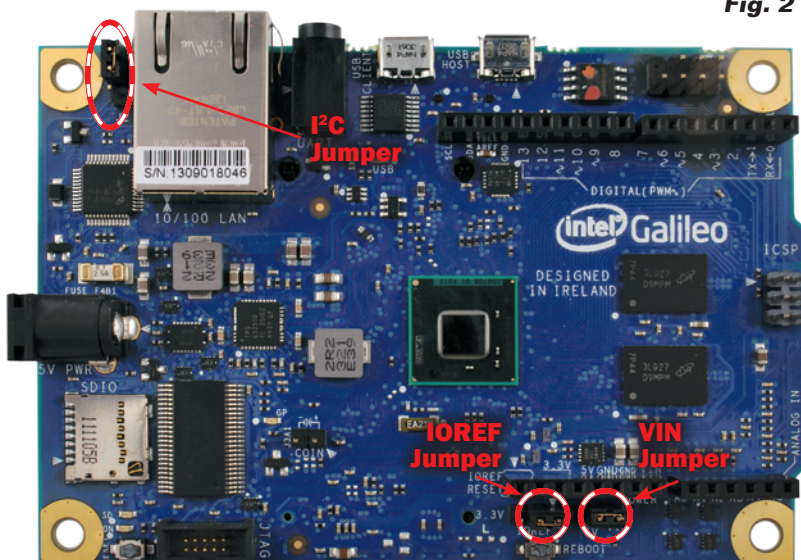


- GND pin di massa;
- IOREF - Il pin IOREF può essere configurato a 3,3V o 5V mediante un selettore a ponticello;
- RESET - Portato a livello LOW permette il reset dello sketch. È utilizzato per portare il pulsante di Reset sugli shield che non permettono l'accesso al pulsante sulla scheda;
- AREF - Il pin di riferimento di tensione per i convertitori ADC. Non è possibile utilizzare questo pin sulla scheda Galileo.

Configurazione dei ponticelli (jumper)

Sulla scheda Galileo sono presenti tre ponticelli (Fig. 2), che permettono di modificarne la configurazione. Il ponticello IOREF J1 permette ai pin di ingresso/uscita di poter gestire livelli sia a 5V che a 3,3V. Quando il ponticello è posizionato sui 5V, i pin sono configurati con il livello alto a 5V ed il pin IOREF è posto a 5V. Allo stesso modo, quando il ponticello è posizionato sui 3,3V sia i pin di I/O che il pin IOREF sono configurati con i livelli a 3,3V in modo da renderli compatibili con gli shield che richiedono livelli a 3,3V. Occorre fare attenzione alla compatibilità dei livelli per evitare di danneggiare in modo irreparabile la scheda Galileo e/o gli shield. Un secondo ponticello J2 permette di modificare l'indirizzo I2C dei pin di espansione

Fig. 2



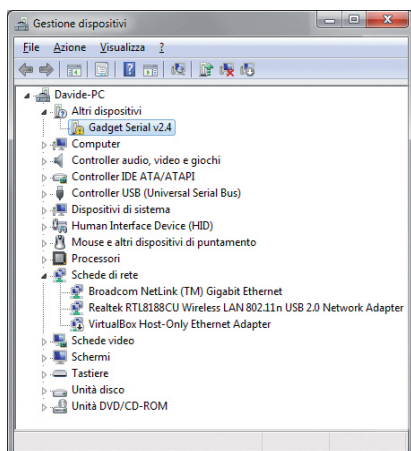


Fig. 3

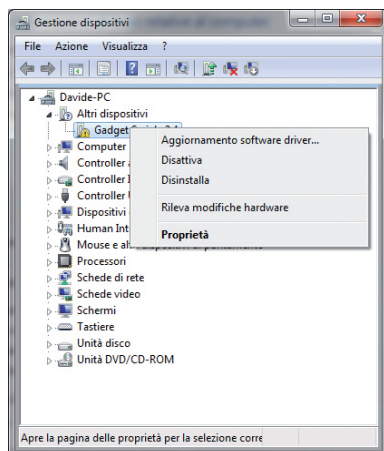


Fig. 4

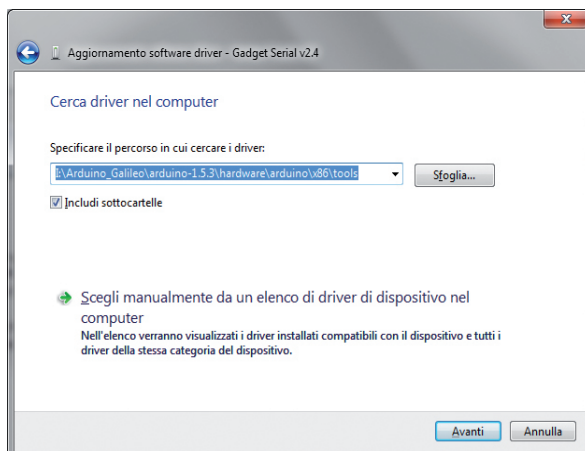


Fig. 5

di I/O e della memoria EEPROM, funzionalità entrambe gestite via bus I2C in modalità slave. Con il ponticello collegato al pin 1, quello con il piccolo triangolo bianco, i pin di espansione di I/O hanno indirizzo 0100001 e la memoria EEPROM 1010001. Modificando la posizione del ponticello si assegna ai pin di espansione I/O l'indirizzo 0100000 e alla memoria EEPROM l'indirizzo 1010000. Questa possibilità permette di risolvere eventuali conflitti di indirizzi nel caso si vogliano impiegare dispositivi esterni slave che utilizzano il bus I2C. Il terzo ponticello J3 permette di gestire l'alimentazione degli shield compatibili Arduino montati sulla scheda Galileo. Con il ponticello montato, lo shield viene alimentato con la tensione di 5V proveniente dal regolatore di tensione interno alla scheda. Se lo shield deve essere alimentato con una tensione superiore ai 5V è indispensabile rimuovere il ponticello e dotare lo shield di alimentazione autonoma. Anche in questo caso bisogna prestare attenzione. Dimenticare di rimuovere il ponticello alimentando lo shield a tensioni superiori a 5V può portare danni irreparabili alla scheda Galileo.

METTIAMO IN FUNZIONE GALILEO

Per poter essere programmata, la

scheda Galileo richiede un apposito IDE reperibile all'indirizzo <http://www.intel.com/support/galileo/index.htm> sezione "Download". L'aspetto esterno e l'interfaccia utente è identica all'IDE classico di Arduino, la differenza sta nel fatto che le schede selezionabili nella voce di menu "Board" sono quelle della famiglia Intel (per ora solo Galileo). Anche diversa è la toolchain di compilazione degli sketch, che comunque resta completamente trasparente all'utilizzatore.

Prima di tutto colleghiamo l'alimentazione alla scheda Galileo utilizzando l'alimentatore fornito. Adattiamo sullo stesso la spina compatibile con il nostro impianto e colleghiamo la spina coassiale all'opposita presa sulla scheda. Ricordiamo che la prima operazione da eseguire sulla scheda è il collegamento dell'alimentazione, pena la possibilità di danni irreparabili. A questo punto la programmazione può avvenire unicamente mediante collegamento via cavo USB al PC sul quale è installato l'IDE di programmazione. Aspettiamo un attimo prima di collegare il cavo USB alla porta microUSB client, quella vicina al connettore Ethernet.

Prima di poter programmare la scheda con il primo sketch dobbiamo eseguire dei passi preliminari. Nell'ordine, scarica-

re e installare l'IDE di sviluppo, installare il driver USB per far riconoscere la scheda ed infine aggiornare il firmware della scheda. L'ultima operazione è probabilmente richiesta dal fatto che la scheda in nostro possesso è in versione preliminare e porta a bordo un firmware non aggiornato. In ogni caso la funzionalità è prevista nell'IDE e va eseguita prima del caricamento di qualsiasi sketch.

Dopo aver scaricato dall'indirizzo indicato la versione di IDE adatta al nostro PC, lo scompattiamo semplicemente, in una cartella di nostro gradimento. Ora possiamo procedere ad installare il driver USB. Verifichiamo di aver collegato l'alimentazione a Galileo, poi colleghiamo il cavo USB al PC ed alla presa micro USB client di Galileo, quella vicina allo shield Ethernet. Aspettiamo qualche tempo e probabilmente vedremo un tentativo non riuscito di installazione automatica di driver USB. Per conferma, dal menu Start andiamo nel pannello di controllo, selezioniamo "Sistema" e poi "Gestione dispositivi". Troveremo una situazione simile a quella di Fig. 3. Una breve nota, noi abbiamo eseguito l'installazione utilizzando un PC con sistema operativo Windows 7, con localizzazione italiana, di conseguenza le figure si riferiscono a questo tipo di am-

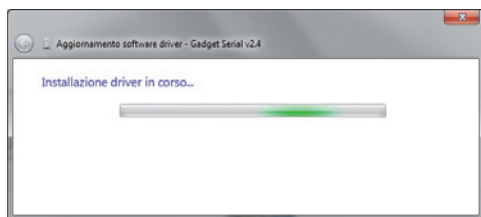


Fig. 6

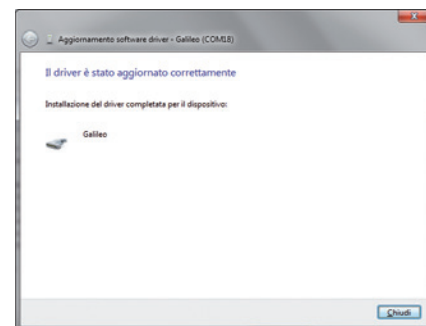
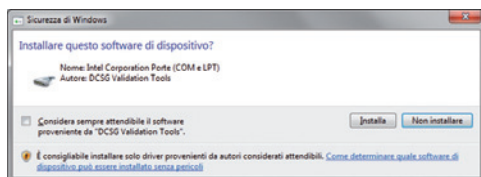


Fig. 8

Fig. 7

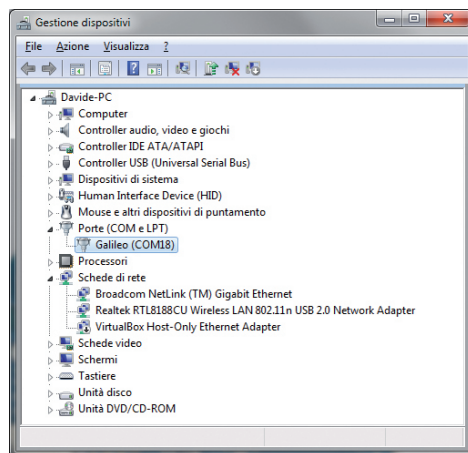


Fig. 9

biente. Le differenze sono poche per localizzazioni in inglese e per PC con sistema operativo Windows XP. Nella Fig. 3 è possibile vedere il dispositivo "Gadget Serial v2.4" evidenziato con un segnale giallo con all'interno un piccolo punto esclamativo. Segno che l'installazione del relativo driver non è andata a buon fine. Possiamo trovare "Gadget Serial v2.4" sotto il menu a espansione "Altri dispositivi" oppure "Porte (COM e LPT)". Per aggiornare il driver clicchiamo col tasto destro del mouse sulla voce "Gadget Serial v2.4" e dal menu a tendina scegliamo "Aggiornamento software driver" (Fig. 4). Nella finestra di dialogo scegliamo "Cerca driver nel computer" e nella successiva finestra, con "Sfoglia" andate alla cartella (Fig. 5):

hardware/arduino/x86/tools

che si trova all'interno della vostra installazione dell'IDE. Lasciate il segno di spunta nel check-box "Includi sottocartelle" e cliccate sul pulsante "Avanti" per iniziare l'installazione del driver (Fig. 6). Se appare la segnalazione visibile in Fig. 7 proseguite cliccando sul pulsante "Installa" ed attendete il termine dell'installazione. Confermate il tutto cliccando sul pulsante "Chiudi" (Fig. 8). Alla fine, tornando alla finestra "Gestione dispositivi",

sotto la voce Porte (COM e LPT) troviamo la porta COM assegnata alla nostra scheda Galileo, nel nostro caso COM18 (Fig. 9). Negli utilizzi successivi della connessione USB la porta COM può cambiare, verificate sempre nell'IDE e se necessario nella finestra "Gestione dispositivi". Ora possiamo aprire l'IDE e fare

in modo di far riconoscere la scheda Galileo. Dal menu "Strumenti" >> "Board" scegliamo Intel® Galileo (Fig. 10), che in effetti è l'unica scelta possibile. Successivamente, sempre dal menu "Strumenti" >> "Porta seriale" scegliamo la porta assegnata alla scheda, nel nostro caso COM18 (Fig. 11). A questo

Fig. 10

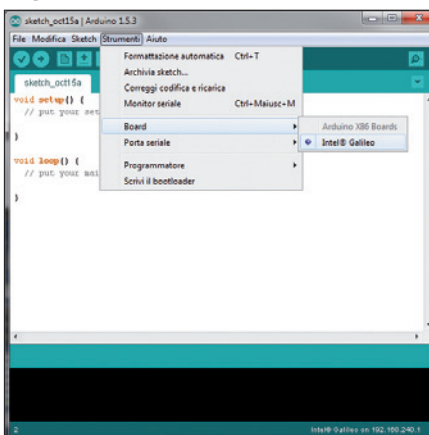


Fig. 11

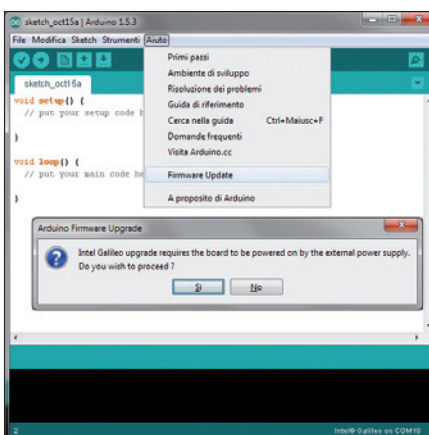
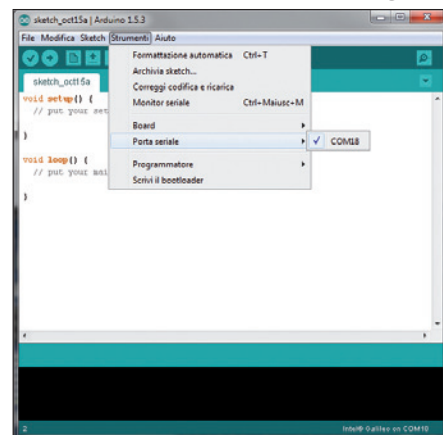


Fig. 12

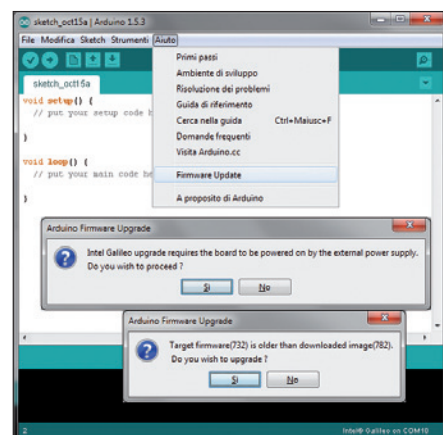


Fig. 13

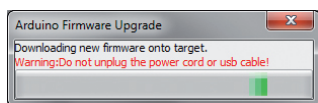
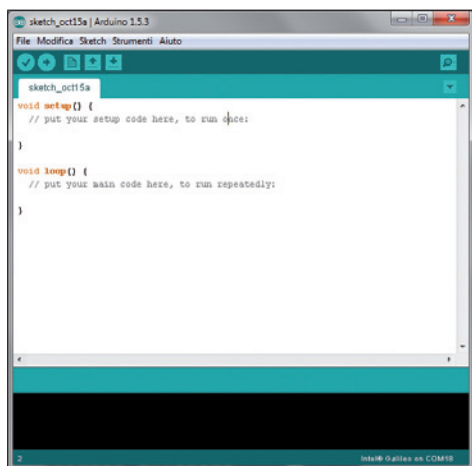
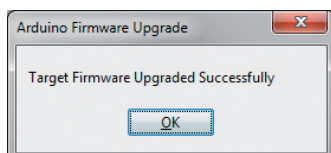


Fig. 14

punto possiamo programmare la scheda con l'IDE. Come prima operazione aggiorniamo il firmware della scheda, come richiesto dalle istruzioni "Getting started". Prima di questa operazione evitiamo di caricare qualsiasi sketch. Per aggiornare il firmware andiamo al menu "Aiuto" >> "Firmware update". Appare un messaggio che ci chiede di verificare che la scheda sia alimentata attraverso il connettore coassiale (Fig. 12). Verifichiamo e rispondiamo "Sì". Non dovrebbe essere possibile una situazione diversa, data la forte raccomandazione di alimentare sempre la scheda prima di collegarla al PC. Un nuovo messaggio ci segnala la versione di firmware installata sulla scheda e la versione dell'aggiornamento disponibile per la sostituzione (Fig. 13). Rispondiamo "Sì" anche a questa finestra e vedremo iniziare il download del nuovo firmware.

Un ulteriore messaggio ci invita a non togliere alimentazione alla scheda fino al termine del processo. Ci mancherebbe. Dopo

Fig. 15



alcuni lunghi minuti, una nuova finestra ci avvisa che il download è terminato. Con sollievo rispondiamo "OK". Ora proviamo a caricare il solito sketch di prova che ci permette di far lampeggiare un LED, il LED collegato al pin 13 presenta sulla scheda. Seguiamo la catena di menu "File" >> "Esempi" >> "01.Basics" >> "Blink".

Verifichiamo, dal menu Strumenti, che la scheda e la porta seriale siano impostati correttamente. Clicchiamo sul pulsante "Carica" e aspettiamo la compilazione ed il successivo download del modulo compilato sulla scheda. Oops, dei messaggi rossi, quelli che normalmente nell'IDE di Arduino segnalando la presenza di errori (Fig. 16). Leggiamo, ci dicono che lo sketch è stato caricato correttamente sulla scheda. Perché non farli bianchi come nell'IDE "standard"? Misteri. Guardiamo il LED sulla scheda e vediamo un lampeggio verde una volta al secondo. Tutto ha funzionato a dovere.

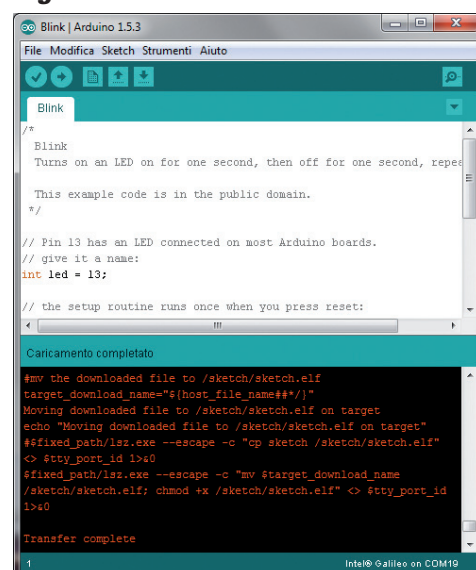
CONCLUSIONI

Non abbiamo avuto tempo di approfondire ulteriormente le funzionalità della scheda Galileo, in particolare la possibilità di caricare una versione di GNU/Linux su micro SDCard che permette il boot dalla SDCard stessa e possiede funzionalità aggiuntive rispetto alla versione "base" memorizzata sulla memoria NAND. Le funzionalità aggiuntive riguardano la presenza dei driver per i dongle WiFi USB. La presenza di un server SSH e quindi la possibilità di collegarci al sistema operativo dall'esterno e la presenza del pacchetto OpenCV, per il trattamento immagini, già installato. Per ora la nostra impressione è che la scheda Galileo rappresenta il punto d'ingres-

so per chi vuole sperimentare in modo economico (facilitato dalla presenza dell'IDE) una versione di GNU/Linux embedded in architettura Intel, per poi passare ad utilizzare sistemi di fascia superiore caratterizzati dalla stessa architettura. Infatti la scheda dispone di diverse espansioni adatte a interfacciare dispositivi tipici del mondo Intel.

Da quello che abbiamo capito lo sketch di Arduino "gira" all'interno del sistema GNU/Linux come un processo utente. Lo svantaggio di questa soluzione è di non poter disporre di un microcontrollore completamente dedicato all'elaborazione dello sketch, indipendente dal processore GNU/Linux. Il vantaggio è che dovrebbe essere semplice fare interagire lo sketch con gli altri processi in elaborazione su GNU/Linux, in quanto si trovano sulla stessa "macchina". Trattandosi di una versione preliminare è presto per dare un giudizio definitivo. Approfondiremo la componente GNU/Linux della scheda e vi terremo informati sulle possibilità di applicazione di Galileo. ■

Fig. 16



3D PRINTING CONTESTI

a cura della REDAZIONE

STAMPA & VINCI

Disponi di una stampante 3D che utilizzi quotidianamente per realizzare gli oggetti più vari? Inviaci, entro il 31 marzo 2014, quello che ritieni il più interessante o suggestivo.

Pubblicheremo le stampe più belle e premieremo gli autori più bravi.

Veder nascere, strato dopo strato, un oggetto fisico che abbiamo appena finito di disegnare, e che magari ci servirà per costruire un prodotto più complesso, o semplicemente un oggetto personalizzato che nessun altro possiede - un semplice vaso, una lampada o un portapenne - provoca un'emozione difficilmente descrivibile.

Cercare di capire cosa possiamo fare con una stampante 3D leggendo un articolo di giornale che ne parla o guardando una pagina pubblicitaria non è così semplice: per capire fino in fondo quello che una stampante 3D può fare è necessario mettersi a stampare.

Non a caso coloro che riescono a "mettere le mani" su uno di questi oggetti, continuano a stampare, stampare, stampare. Dopo aver realizzato gli oggetti per i quali la macchina è stata acquistata, magari realizzano un regalo per la moglie o il pezzo rotto del giocattolo del figlio o una originale lampada.

E col tempo e con l'esperienza accumulata gli oggetti riescono sempre meglio: ed ecco, allora, la voglia di condividere con altri utenti i propri lavori, ecco gli upload sui siti dedicati, primo tra tutti Thingiverse.

Perché, dunque, ci siamo domandati, non incentivare questa voglia di fare con qualcosa che premi i più bravi e che sia da stimolo per tutti gli altri?

Ecco dunque nascere questo concorso riservato a quanti posseggono una stampante 3D purché in tecnologia a filamento

fuso, non importa di che marca, può essere la nostra 3Drag ma può essere qualsiasi altra macchina.

E di macchine ormai ce ne sono in giro sempre di più: lentamente ma inesorabilmente le stampanti 3D, specie quelle di fascia bassa, si stanno diffondendo tra studenti, hobbysti, nuovi artigiani, creando nuove competenze e nuove opportunità di lavoro.

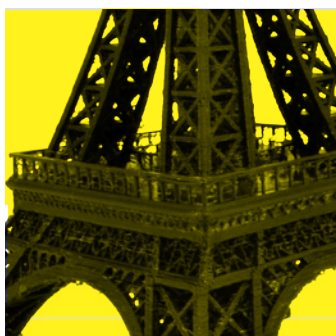
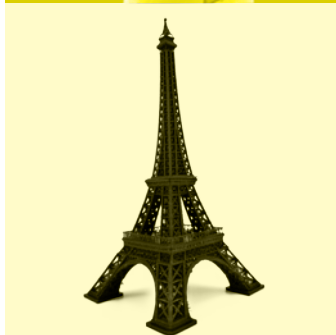
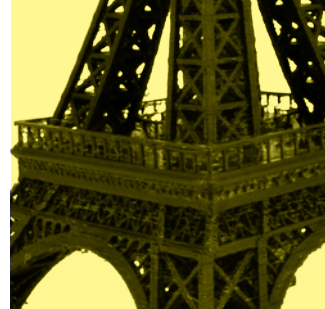
Il concorso al quale abbiamo pensato vuole premiare sia le persone con spiccato senso artistico che quelle più portate per gli aspetti tecnico/progettuali, senza dimenticare quanti, con la loro esperienza, riescono a ricavare dai file disponibili in rete oggetti perfetti dal punto di vista costruttivo.

IL CONTEST

Abbiamo perciò identificato tre categorie per permettere a ciascun partecipante di esprimersi al meglio. Le tre categorie sono:

- 1) tecnica di stampa;
- 2) utilità e originalità;
- 3) design/senso artistico.

Nella prima, la nostra giuria valuterà le opere sotto l'aspetto della qualità di stampa, delle caratteristiche meccaniche e qualitative dell'oggetto stampato, ma anche della difficoltà di realizzazione e dell'impegno richiesto dalla produzione o modifica del file di stampa 3D e della complessità meccanica: per esempio, una stampa che realizza un meccanismo



Le regole

Il concorso è rivolto alle vostre realizzazioni con stampanti FFF/FDM e prevede tre categorie:

- 1) tecnica di stampa;
- 2) utilità/originalità;
- 3) design/senso artistico.

- Per partecipare bisogna iscriversi sul nostro sito web compilando l'apposito form ed eventualmente allegando il file di stampa dell'oggetto che si intende proporre.
- Ogni concorrente può proporre il proprio elaborato in tutte e tre le categorie.
- I file relativi agli oggetti proposti nelle categorie 2 e 3 devono essere originali o comunque modificati in maniera significativa dal concorrente.
- La spedizione delle stampe è a carico del concorrente.
- Tutte le stampe verranno restituite ai concorrenti al termine del concorso a nostre spese.



con parti in movimento ben funzionanti, sarà sicuramente considerata più impegnativa e meritevole di un oggetto a pezzo singolo.

Nella seconda, rientrano le stampe che riguardano oggetti di utilità per la nostra vita quotidiana o il nostro lavoro (quindi non soprammobili) che il concorrente ha ideato, disegnato e stampato. In linea di massima il disegno deve essere originale anche se sono ammessi progetti che sono delle variazioni sul tema (purché significative) di file o proposte disponibili in rete.

Nella terza categoria si collocano stampe artistiche e di oggetti di design, di cui verranno premiate le qualità estetiche. Anche in questo caso gli oggetti ideati e inviati debbono essere assolutamente originali. Al termine del Contest tutti i lavori pervenuti verranno resi disponibili on-line ed i file potranno essere scaricati liberamente da tutti i lettori.

PER PARTECIPARE

Tutti i dettagli del Contest sono disponibili sulla home page del nostro sito www.elettronica.in.it compreso il modulo da compilare per l'iscrizione e il successivo invio del file di stampa (da caricare sul sito) e dell'oggetto stampato (da inviare via posta con pacchetto postale raccomandato).

Diciamo sin da subito che, al termine del Contest, tutti gli oggetti verranno restituiti ai rispettivi concorrenti con spese di spedizione a nostro carico. Ogni concorrente potrà iscrivere la propria realizzazione anche in tutte e tre le categorie, in quanto, in qualche modo, un oggetto può essere tecnicamente avanzato, esteticamente

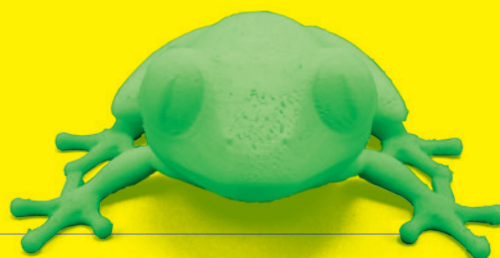


valido e anche utile; è tuttavia consigliabile iscriversi in una sola categoria proprio per dare il meglio di sé nel campo prescelto.

Ricordiamo che i file delle categorie 2 e 3 dovranno essere originali, ovvero ideati e disegnati dal concorrente; sono ammessi anche oggetti ispirati a lavori disponibili in rete purché il file originale venga rielaborato in maniera significativa.

IN PALIO...

Per incentivare la partecipazione del maggior numero possibile di persone, abbiamo deciso di mettere in palio tre premi prestigiosi; per ogni categoria verrà scelto dalla giuria un vincitore e delle tre soluzioni scelte, sarà decisa la migliore in senso assoluto. In altre parole, la giuria valuterà la proposta migliore, indipendentemente dalla categoria cui appartiene. A chi avrà proposto la soluzione vincente regaleremo un biglietto aereo A/R per San Francisco (California) in occasione della prossima Maker Faire di



San Mateo (una località a poche miglia a sud di San Francisco) che si terrà nel maggio 2014. Un'occasione imperdibile per vedere i progetti di centinaia di maker e attrazioni che solo lì potrete trovare nella località della California che ha visto la nascita di questo genere di manifestazioni.

Le altre due soluzioni scelte, classificatesi al primo posto della loro categoria (resta ovviamente esclusa la categoria cui appartiene il vincitore assoluto), saranno premiate ciascuna con una stampante 3Drag in kit di montaggio. Vi terremo costantemente aggiornati sull'evoluzione del Contest, sia sulle pagine

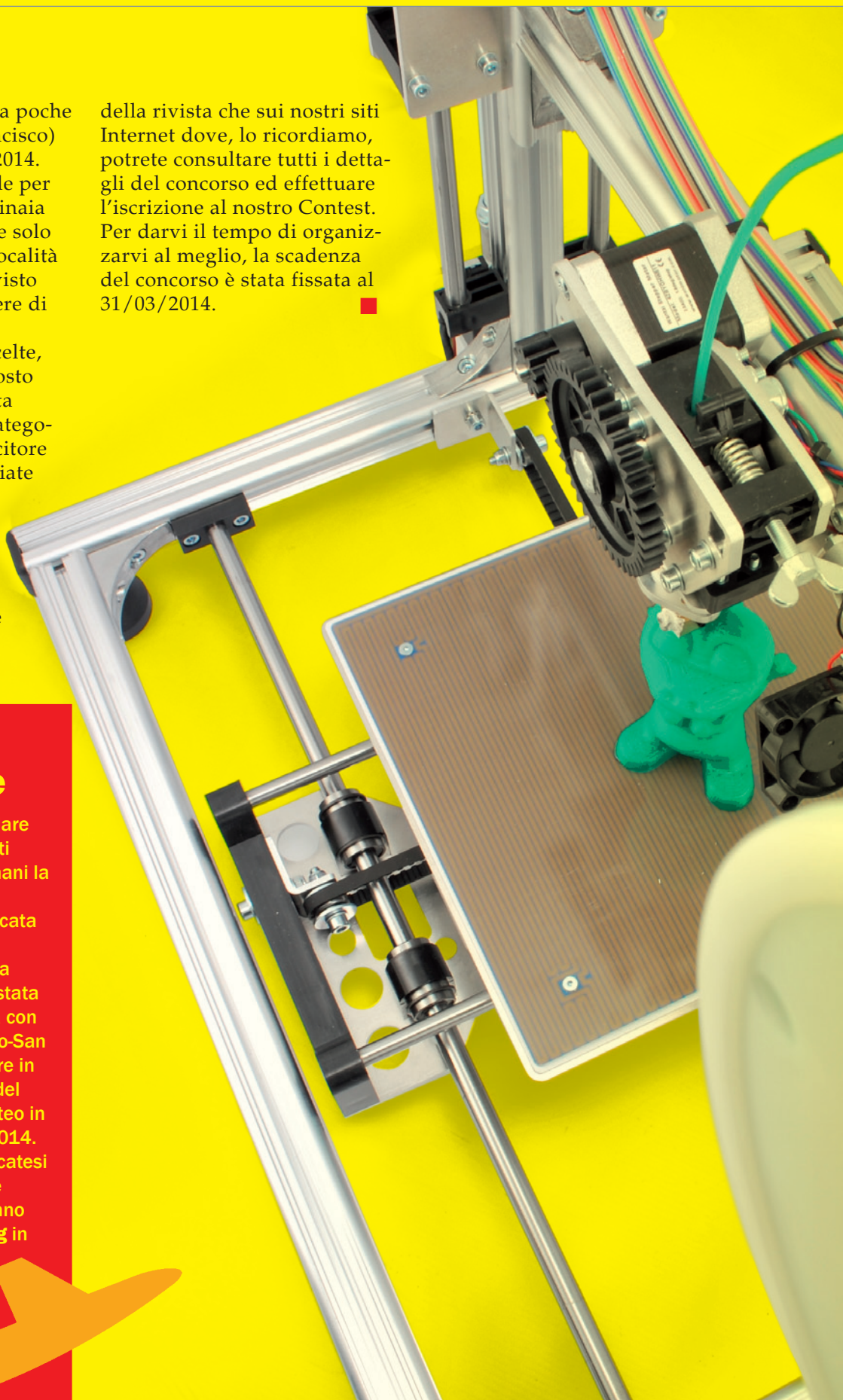
della rivista che sui nostri siti Internet dove, lo ricordiamo, potrete consultare tutti i dettagli del concorso ed effettuare l'iscrizione al nostro Contest. Per darvi il tempo di organizzarvi al meglio, la scadenza del concorso è stata fissata al 31/03/2014. ■

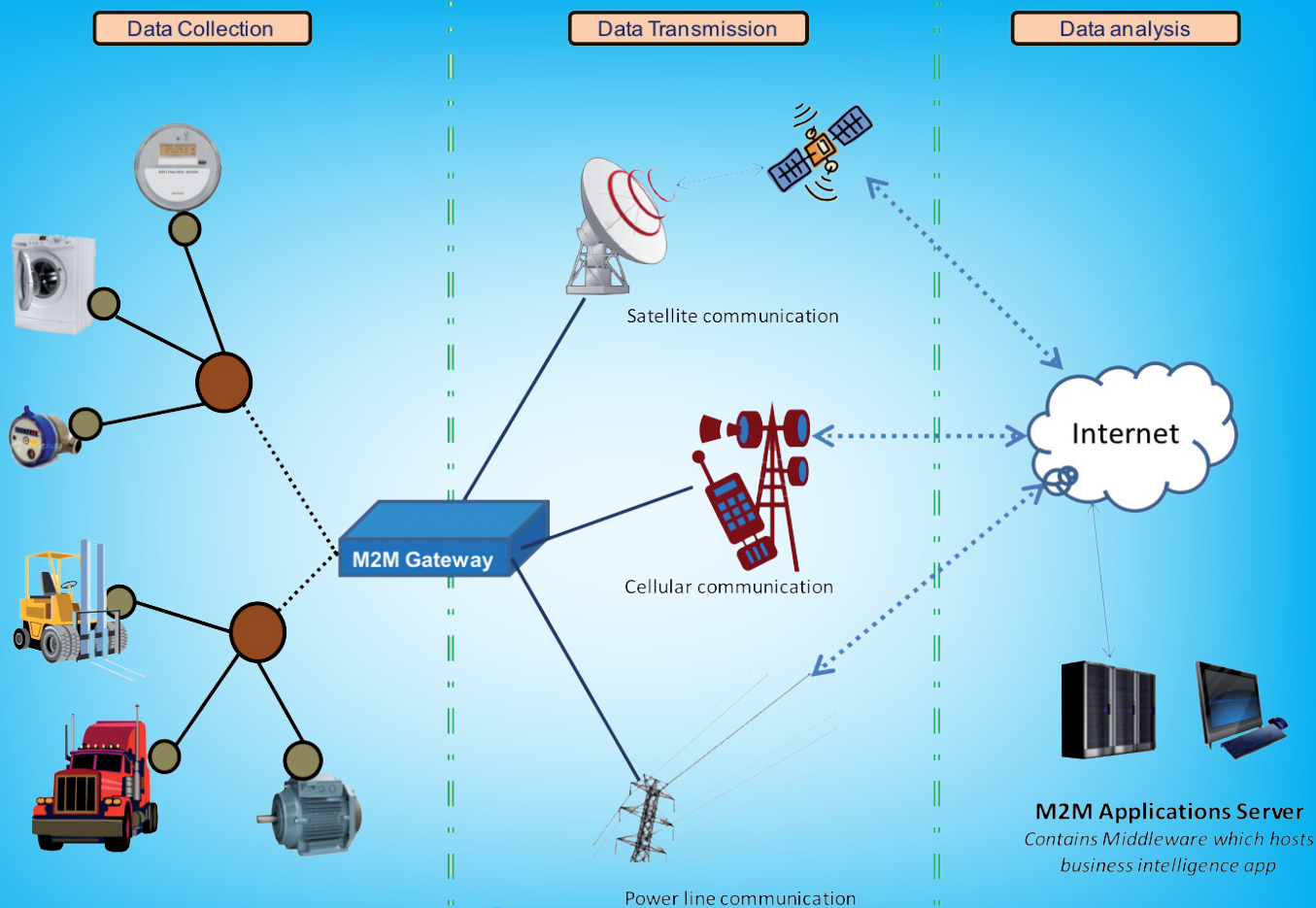
Vola alla Maker Faire

E se non sarai tu ad andare in California, potrai rifarti costruendo con le tue mani la nostra 3Drag.

Infatti la soluzione giudicata la migliore in assoluto, indipendentemente dalla categoria nella quale è stata proposta, sarà premiata con un biglietto aereo Milano-San Francisco A/R per andare in California in occasione del **Maker Faire** di San Mateo in programma a Maggio 2014.

Le due soluzioni classificate al primo posto nelle due categorie rimaste, saranno premiate con una **3Drag** in kit di montaggio.





4G & IPv6 PER MACCHINE DI COMUNICAZIONE PIÙ INTELLIGENTI

di RAMANUJA KONREDDY

Dopo aver rivoluzionato il modo con cui ogni persona comunica con tutte le altre, la rete Internet è ora pronta per dare il via ad una seconda rivoluzione: la proliferazione di comunicazione intelligente tra macchine. Le tecnologie che introdurranno

no comunicazioni intelligenti *machine-to-machine* (M2M) sono gli standard wireless 4G ad alta velocità e *Internet Protocol version 6* (IPv6). Insieme, queste tecnologie consentiranno ai progettisti di creare un mondo nel quale dispositivi intelligenti potrebbero offrire un nuovo livello di

La combinazione di 4G e IPv6 consentirà ai sistemi M2M di comunicare in maniera più intelligente, spiega Ramanuja Konreddy, Senior Product Marketing Engineer MCU32 Division di Microchip.

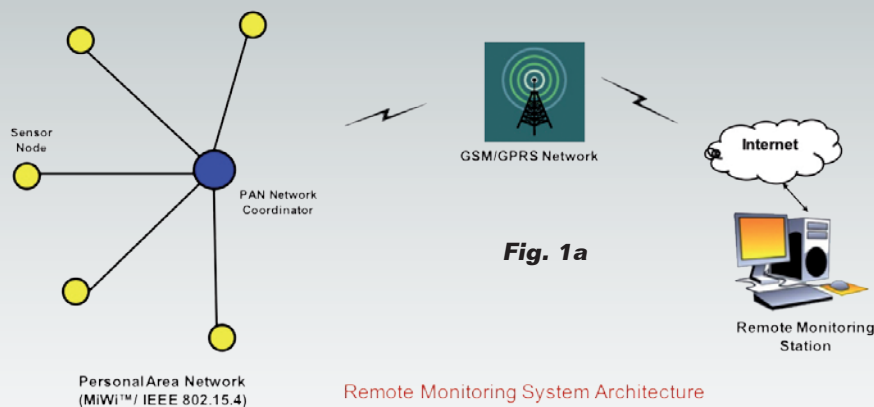
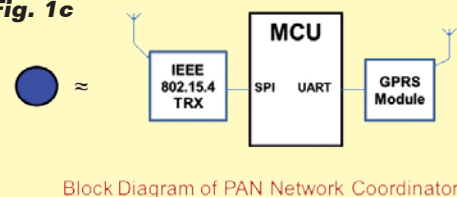


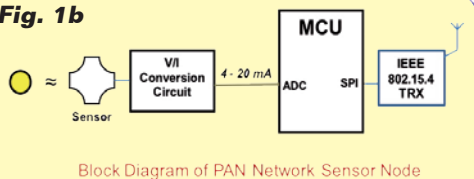
Fig. 1a

Fig. 1c



Block Diagram of PAN Network Coordinator

Fig. 1b



Block Diagram of PAN Network Sensor Node

servizi M2M. Le possibilità comprendono: uno smartphone che potrebbe ricevere un messaggio di allerta e posizione quando l'auto del suo proprietario viene rubata; una fotocamera/video-camera che potrebbe comunicare con dispositivi GPS per suggerire belle località in quella zona; un frigo che potrebbe inviare un promemoria per ricordarci di comprare il burro; uploading automatici di dati da un dispositivo di monitoraggio medico personale verso un GP e una vettura che potrebbe mandarci un avviso quando ha bisogno di essere riparata. Queste e molte altre applicazioni saranno possibili grazie alle comunicazioni intelligenti M2M.

La figura riportata in apertura mostra l'interazione tra gli elementi chiave della tecnologia M2M. Questi elementi consentono al sistema di ricevere dati da un dispositivo finale e trasmettere quei dati, su una rete di comunicazione, ad una applicazione software che li elabora e li converte in informazioni riconoscibili per l'applicazione. Tipicamente, le reti M2M trasmettono

no dati usando comunicazioni *Global System for Mobile (GSM)*, *General Packet Radio Service (GPRS)*, onde convogliate o via satellite.

I progettisti di applicazioni industriali hanno capito rapidamente il potenziale della comunicazione M2M. La combinazione di potenti sensori, avanzati microcontroller embedded e reti wireless ha permesso nuovi livelli di gestione attraverso applicazioni come il controllo remoto di apparecchiature industriali e la gestione di flotte aziendali. Nel prendere in esame queste due applicazioni con maggiore dettaglio, si evidenzia come la comunicazione M2M può trasformare il modo in cui i sistemi operano, portando risparmi sui costi, migliorando l'efficienza e riducendo i lavori manuali.

CONTROLLO REMOTO

Un sistema di controllo remoto per un'apparecchiatura industriale potrebbe utilizzare ingressi combinati di sensori che rilevino vibrazioni, perdite di liquidi, temperatura, usura meccanica e rumore, per valutare le condizioni dell'apparecchiatura. La Fig. 1(a) mostra la posizione di ognuno dei sensori o nodi, rappresentati dai punti gialli e connessi in modalità a stella al coordinatore della rete rappresentato dal punto blu.

Insieme, i nodi ed il coordinatore di rete formano una *Personal Area Network (PAN)*. Il coordinatore di PAN si comporta come un gateway M2M tra la PAN e la rete GPRS e trasmette i dati alla stazione di controllo remoto. Se il sensore rileva un parametro fuori dai normali valori di soglia operativa, la stazione di controllo remoto invia un avviso che avvia la manutenzione preventiva necessaria. Essendo in grado di fornire avvisi avanzati di potenziale malfunzionamento delle apparecchiature, la comunicazione M2M può essere usata per ridurre i costi di fermo macchina non pianificati e per aumentare l'efficienza nella programmazione della manutenzione preventiva.

NODI SENSORE E COORDINATORE PAN

I diagrammi a blocchi di un nodo sensore e coordinatore PAN sono mostrati in Fig. 1(b) e Fig. 1(c).

I parametri del sensore dell'apparecchiatura sono trasmessi attraverso un loop di corrente tra 4mA e 20mA, costituito da un sensore fisico che è uno standard di segnalazione per processi industriali e di controllo. Questi parametri vengono ricevuti dalla MCU attraverso il suo *Analogue-to-Digital Converter (ADC) on-chip*. La MCU usa un *protocol stack PAN*, come *MiWi™*, per interfacciarsi con un *transceiver IEEE 802.15.4* che invia i dati elaborati del sensore al coordinatore PAN.

Il motore che conduce la comunicazione per l'intero PAN è l'MCU embedded all'interno del coordinatore PAN. Questo MCU si interfaccia sia con il *transceiver* IEEE 802.15.4 che con il modulo GPRS attraverso porte seriali. La memoria flash dell'MCU viene utilizzata per immagazzinare la mappa completa della PAN. MiWi, assegnare indirizzi di rete e stabilire collegamenti a nuovi nodi sensori. Traduce, anche, i dati del protocollo PAN nel formato GPRS da inviare sulla rete GPRS. La MCU controlla il modulo GPRS usando comandi *Attention* (AT) per la trasmissione dei dati. I fattori chiave per la scelta della MCU embedded dovrebbe essere un elevato *throughput*, memoria sufficiente, ed un ampio range di periferiche che supportino *Direct Memory Access* (DMA). Un collaudato e solido stack di protocolli wireless, come MiWi di Microchip, e l'accesso alla libreria GPRS sono utili ad assicurare una connessione stabile tra la rete PAN e GPRS. L'uso di una scheda di sviluppo, come *PIC32 Wi-Fi® Comm Board* di Microchip mostrata nella Fig. 2(a), può ugualmente essere d'aiuto per accelerare lo sviluppo di una applicazione wireless.

GESTIONE DELLE FLOTTE AZIENDALI CON M2M

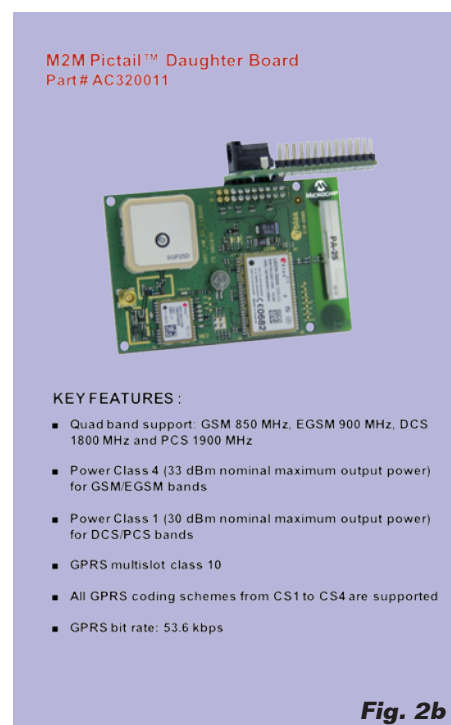
La gestione delle flotte aziendali è divenuta una diffusa applicazione per la tecnologia M2M. Veicoli dotati di dispositivi GPS vengono tracciati e i dati della posizione vengono inviati periodicamente al sistema di controllo utilizzando moduli GPRS o GSM. Il modulo GPS viene utilizzato per fornire le coordinate di posizione dei veicoli, mentre il bus CAN è utilizzato per raccogliere dati dai diversi moduli

all'interno dei veicoli, come l'*Antilock Brake System* (ABS), l'*Engine Control Unit* (ECU) e gli *airbag*. Un microcontroller 32-bit fornisce le risorse per un gateway CAN/GPRS interfacciando il bus CAN, come pure ai moduli GPRS e GPS, e traducendo i dati CAN nel formato GPRS, prima che siano inviati sulla rete GPRS alla *Fleet Management Station*. La *daughter board* M2M PICTail™ di Microchip contiene moduli GPS e GSM/GPRS equipaggiati con numerosi comandi AT, oltre che del vasto range di caratteristiche mostrate in Fig. 2(b). La scheda M2M PICTail fornisce tutte le informazioni hardware, i software binari comprensivi degli stack, schema di banco di prova e software di prova, certificazione, formazione e documentazione che permettono un *time-to-market* più rapido. Consente di realizzare una soluzione di comunicazione wireless a basso consumo, ultra compatta per una facile integrazione nella progetta-

zione di prodotti consumer di grandi volumi, industriali e *automotive*.

UN MONDO CON MACCHINE PIÙ INTELLIGENTI

Il mondo emergente della comunicazione ad alta velocità 4G e *Internet Protocol* versione 6 (IPv6) stanno consentendo alla comunicazione tra macchine di divenire qualcosa di simile alla comunicazione tra le persone. La proliferazione di reti di comunicazione e la flessibilità delle soluzioni di silicio significano che il mondo sta per raggiungere un punto di svolta nel quale il numero di persone connesse attraverso il villaggio Internet sarà superato da quello di dispositivi e macchine connessi via Internet. Per i progettisti questo significa che c'è un' aumentata pressione per vincere la corsa verso il mercato di prodotti che siano attrezzati per disporre di tutte le possibilità della comunicazione M2M.



Continuiamo il nostro viaggio alla scoperta del sistema di sviluppo grafico per microcontrollori proposto da Matrix Multimedia. In questa puntata iniziamo a lavorare con le periferiche di comunicazione avanzate, partendo da quella che ha riscosso un notevole e crescente successo negli ultimi 15 anni: il CAN-Bus. Quinta puntata.

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

Il mese scorso abbiamo illustrato l'uso delle periferiche di comunicazione di base, imparando ad utilizzare le porte di comunicazione UART, I²C, SPI e OneWire. In questa puntata introduciamo una periferica di comunicazione decisamente più avanzata, che permette di creare reti anche piuttosto complesse e largamente utilizzata ai giorni nostri, specialmente nel settore automotive: stiamo parlando della porta CAN (Controller Area Network), che oggi costituisce lo standard *de facto* per le reti di comunicazione delle automobili e si sta espandendo anche ad applicazioni industriali e custom, grazie alla sua robustezza, alla relativa semplicità

rispetto a soluzioni simili e ai costi ridotti dei componenti. Come fatto nelle puntate precedenti, arricchiremo il tutto con un progetto pratico: realizzeremo un prototipo di sensore di parcheggio per autoveicoli.

Controller Area Network (CAN)

Il protocollo CAN nasce negli anni '80 del secolo scorso dalla Robert Bosch GmbH, che aveva l'esigenza di sviluppare uno standard per le comunicazioni seriali a corto raggio tra diverse unità elettroniche di controllo, di tipo multicast, dotato di elevata immunità ai disturbi, basso costo ed elevata affidabilità. Lo standard im-

piega come mezzo trasmissivo una linea differenziale bilanciata (i due terminali sono indicati come CANH e CANL), esattamente come nel caso dell'RS485. Il massimo bit rate raggiungibile è di 1Mbit/s, con estensione massima della rete di circa 100 m. Accontentandosi di velocità inferiori è possibile coprire distanze maggiori (ad esempio con 125 kbit/s si arriva a 500 m). In **Tabella 1** sono riportati alcuni esempi di estensioni massime della rete rispetto alla velocità trasmissiva.

La tecnica trasmissiva del CAN usa un modello basato su bit "dominanti" e "recessivi", in cui i bit dominanti sono gli 0 logici ed i bit recessivi sono gli 1 logici. Se, durante una trasmissione, il nodo A invia un bit dominante e, contemporaneamente, il nodo B invia un bit recessivo, allora il bit dominante "vince" fra i due. Durante la trasmissione, ogni nodo che sta trasmettendo verifica lo stato del bus e confronta il bit ricevuto con il bit trasmesso: se viene rilevato un bit dominante quando ne viene trasmesso uno recessivo, allora il nodo che ha rilevato l'errore interrompe la comunicazione.

Con questa tecnica, se due nodi iniziano una trasmissione contemporaneamente, essi si contendono l'uso esclusivo del bus (arbitraggio) trasmettendo i propri dati, bit dopo bit, finché uno dei due non rileva un errore di trasmissione, autoescludendosi (si dice che il nodo in questione perde l'arbitraggio) dal bus. Nella teoria delle reti di comunicazione, un approccio di questo tipo è definito CSMA/BA (Carrier Sense Multiple Access/Bitwise Arbitration).

Il protocollo CAN è centralizzato sui messaggi, che sono chiamati in genere anche frames, identificati per mezzo di un ID. Ogni nodo gestisce uno o più frame in ricezione e/o trasmissione, a seconda dell'architettura della network.

I frame possono essere di quattro tipi:

- *Data frame*, frame contenente i dati che il nodo trasmette;
- *Remote frame*; richiede la trasmissione di un determinato identificatore;
- *Error frame*; trasmesso da un qualsiasi nodo che ha rilevato un errore;
- *Overload frame*; frame che introduce un ritardo fra data frame e/o remote frame.

I Data frame sono quelli cui è affidata l'effettiva trasmissione dei dati e possono essere di due tipi, a seconda della versione dello standard CAN presa in considerazione:

- *Base frame format*; ID ad 11 bit (versione 2.0A);
- *Extended frame format*; ID a 29 bit (versione 2.0B).

I dispositivi standard CAN devono riconoscere il formato *base frame* e possono riconoscere (ma devono comunque tollerarlo) il formato *extended frame format*. Il CAN base permette $2^{11}=2048$ tipi di messaggi diversi, ma da specifiche Bosch se ne possono usare solo 2031. Nella versione *extended* si possono avere fino a $2^{29}=536.870.912$ tipi di messaggi. L'*extended frame format*, detto anche CAN esteso, è oggi il formato più diffuso nelle varie applicazioni che utilizzano questo standard.

Un'applicazione che faccia uso del protocollo CAN necessita di almeno un paio di componenti elettronici aggiuntivi: un controller ed un transceiver; quest'ultimo è l'elemento che si occupa dell'interfacciamento a livello fisico con il bus. È quindi responsabile della corretta traslazione e dell'adattamento dei livelli logici delle linee. Si interfaccia direttamente con il controller. Ci sono diverse aziende al mondo che producono transceiver CAN; noi utilizzeremo il transceiver della Microchip, MCP2551.

Il controller è l'elemento che si occupa dell'implementazione del livello data link (livello 2 dello stack OSI); quindi ad esso è demandata la gestione delle principali logiche del bus, svincolando l'utilizzatore dalla gestione di problematiche di arbi-

Tabella 1 - Data-rate riferito all'estensione della rete CAN.

Data-rate(kbps)	Estensione massima (m)
125	500
250	250
500	100

Fig. 1 - Connessioni del componente Extended CAN.

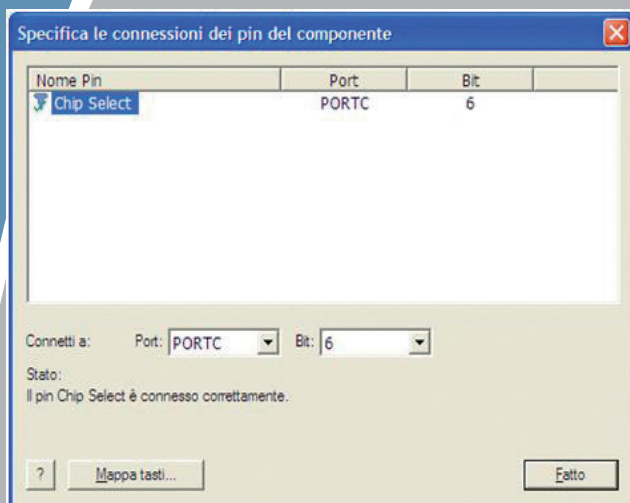
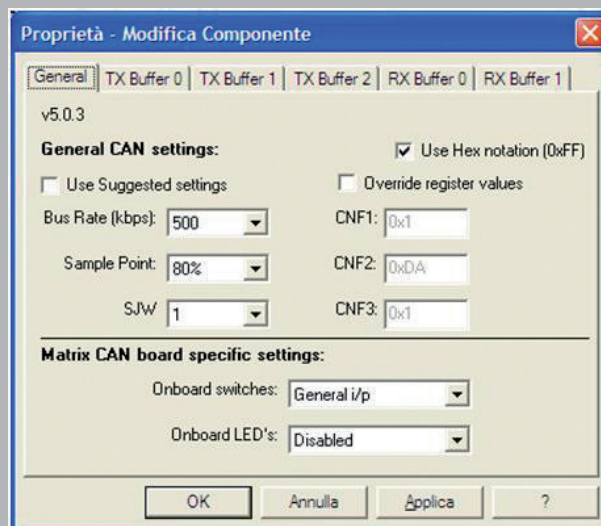


Fig. 2 - Proprietà extra del componente Extended CAN.



traggio, di gestione degli errori, di politiche di ritrasmissione, ecc. Il controller dispone anche di buffer di trasmissione, di ricezione e di maschere, che permettono di configurare il set di messaggi in ricezione e trasmissione, riducendo l'overhead della CPU del microcontrollore a cui è interfacciato. Anche per i controller, esistono diverse aziende produttrici al mondo; nel nostro caso noi faremo uso del controller della Microchip MCP2515.

Il componente CAN in Flowcode

Analizziamo adesso il componente Flowcode che ci permette di semplificare la gestione di un'interfaccia CAN nelle nostre applicazioni. Flowcode dispone di due componenti CAN, uno per la gestione del *base frame format* (che si chiama semplicemente CAN) e uno per la gestione dell'*extended format* (chiamato Extended CAN) quindi, se decidiamo di sviluppare un'applicazione CAN, la prima scelta da fare sarà quella relativa al tipo di formato. Nella versione attuale dell'ambiente, i controller interni sono supportati solo dal componente che supporta il base format, quindi se usiamo il componente extended saremo vincolati all'uso di un controller SPI esterno. In questa puntata ci occuperemo del componente CAN Extended, e quindi utilizzeremo il controller MCP2515 nelle nostre applicazioni.

Per inserire il componente CAN all'interno del nostro progetto Flowcode, sele-

zioniamo il componente dalla barra dei componenti, cliccando sulla voce *Extended CAN*. Una volta inserito, l'icona relativa al componente comparirà sul pannello, ed a questo punto possiamo configurarlo per adattarlo alle esigenze del progetto specifico.

Per prima cosa analizziamo le connessioni, che nel caso di questo componente si riducono al pin di CS del controller CAN (che è un SPI slave). Possiamo accedere alle connessioni cliccando con il tasto destro sull'icona del componente e selezionando la voce "Connessioni". Apparirà la finestra rappresentata in **Fig. 1**.

Per impostazione predefinita, il pin di CS è il pin 6 della porta C; se non c'è l'esigenza di cambiarlo (ad esempio nel caso in cui venga utilizzato l'E-block CAN) può essere mantenuto così. Per configurare il componente clicchiamo con il tasto destro sulla relativa icona e selezioniamo la voce "Proprietà Extra". Comparirà la finestra di **Fig. 2**. Dalla prima scheda di questa finestra è possibile selezionare le principali opzioni di configurazione del bus, come il baud-rate, il sample point ed altri ancora. Questi parametri dipendono a loro volta dalle caratteristiche proprie della rete, come la sua estensione, il numero di nodi, il bus load, ecc. I tab successivi permettono di configurare i vari buffer di ricezione e trasmissione. Nel progetto pratico, vedremo un esempio di configurazione dei buffer.

Tabella 2 - Macro disponibili per il componente CAN.

Nome Macro	Descrizione
Init	Inizializza il componente CAN.
SendBuffer	Invia il messaggio contenuto all'interno del buffer di trasmissione (passato come parametro).
CheckRx	Controlla se ci sono messaggi ricevuti su un determinato buffer di ricezione (passato come parametro).
ShowLEDs	Controlla lo stato dei LED della EB018.
ReadSwitches	Legge lo stato degli switch della EB018.
GetRxDataCount	Riceve il conteggio dei byte del messaggio contenuto in un determinato buffer (passato come parametro).
GetRxData	Estrae un determinato byte dal messaggio presente in un determinato buffer di ricezione. Riceve come parametro sia il buffer che l'indice del byte.
SetTxStdID	Modifica lo standard ID di un determinato buffer di trasmissione (passato come parametro).
SetTxExtID	Modifica l'extended ID di un determinato buffer di trasmissione (passato come parametro) come combinazione dell'ID standard e dell'ID extended.
SetTxExtAsFull	Modifica l'extended ID di un determinato buffer di trasmissione (passato come parametro).
GetRxStdID	Estrae un determinato byte di un determinato messaggio con ID standard. Vengono controllati tutti i buffer e viene passato come parametro l'indice del byte.
GetRxExtID	Estrae un determinato byte di un determinato messaggio con ID extended, visto come combinazione di un ID standard e di un ID extended.. Vengono controllati tutti i buffer e viene passato come parametro l'indice del byte.
GetRxExtIDAsFull	Estrae un determinato byte di un determinato messaggio con ID extended.. Vengono controllati tutti i buffer e viene passato come parametro l'indice del byte.

Ora, come di consueto, procediamo con l'analisi delle macro messe a disposizione da questo componente, che trovate riepilogate nella **Tabella 2**.

Progetto pratico: sensore di parcheggio

Passiamo adesso al progetto pratico relativo a questa puntata. Come abbiamo già accennato, abbiamo pensato di realizzare un progetto pratico un po' più impegnativo, proprio per evidenziare le potenzialità di *rapid development* offerte da flowcode, che permette la realizzazione di progetti anche piuttosto complessi in tempi ridottissimi e con un ottimo risultato finale. Nel caso specifico ci cimentiamo nella realizzazione di un prototipo di sensore di parcheggio per auto, costituito da due nodi: un nodo sensore ed un nodo display, collegati tra di loro tramite bus CAN. Quindi realizzeremo due progetti Flowcode distinti, che utilizzeremo per generare gli eseguibili binari con cui programmare le due schede. Come di consueto, analizziamo le specifiche di progetto, in questo caso raggruppate per nodo.

Nodo Sensore:

- Interfacciamento con sensore ad ultrasuoni SFR02, su bus I²C;
- Interfaccia CAN 2.0B a 500 kbaud;

- Invio dati sulla distanza espressa in cm;
- Autodiagnosi su stato batteria e stato link I²C con il sensore;
- Modalità sleep attivabile tramite messaggio da nodo display.

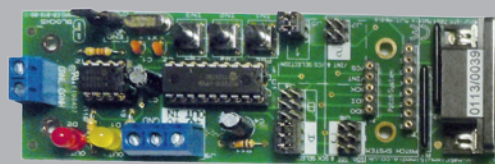
Nodo Display:

- Interfacciamento con display alfanumerico 16x2;
- Interfaccia CAN 2.0B a 500 kbaud;
- Lettura e visualizzazione dei dati sulla distanza;
- Lettura e visualizzazione dei dati diagnostici;
- Gestione della modalità sleep tramite pulsante.

Sebbene il CAN sia nativamente una rete peer-to-peer, è possibile forzare delle relazioni di tipo master/slave, agendo a livello dell'applicazione; è ciò che faremo in questo esempio, dove il nodo display agisce da nodo master (può mettere "a dormire" lo slave nodo sensore), inviando un messaggio apposito per forzare lo stato di sleep, in modo da disabilitare il sistema durante le fasi di guida ed abilitarlo in fase di parcheggio.

Per quanto riguarda la realizzazione hardware, in questo progetto avremo bisogno delle seguenti schede:

Fig. 3a
Prototipo del nodo sensore.



- 2x EB006 "USB PICmicro multiprogrammer board" (PIC18F4580 e quarzo da 12MHz);
- 2x EB018 "CAN bus Board";
- 1x EB005 "LCD board";
- 1x EB016 "Prototypeboard".

Non disponendo di un e-block specifico, per interfacciare l'SRF02 ci serviremo, come già fatto in passato, della EB016, che colleghiamo sulle porte A e B. Posizioniamo l'SRF02 sulla breadboard e colleghiamo i terminali di alimentazione e quelli di comunicazione SDA ed SCK del bus. Non potendo utilizzare la porta hardware I²C (perché, per interfacciare l'MCP2515 usiamo la MSSP in modalità SPI), emuliamo il collegamento I²C a livello software, e quindi usiamo RB0 ed RB1, rispettivamente per SCK ed SDA. Oltre all'SRF02, sulla EB016



dobbiamo posizionare anche un partitore di tensione con rapporto di partizione 1:4, allo scopo di leggere lo stato della batteria, che colleghiamo all'ingresso analogico An0. Completiamo il nodo sensore collegando la EB018 sulla porta C. Per il nodo display, colleghiamo la EB018 sulla porta C e la EB005 sulla porta B. Per entrambi i nodi, l'impostazione dei jumper della EB018 è la seguente:

- J4 su A;
- J5 inserito;
- J6 e J7 disinseriti.

Poi, assicuriamoci di chiudere su entrambe le schede J8, in modo da terminare la linea da entrambi i lati e collegare i cavetti per l'alimentazione.

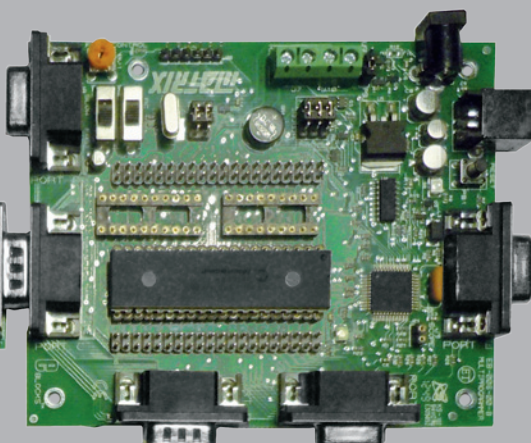
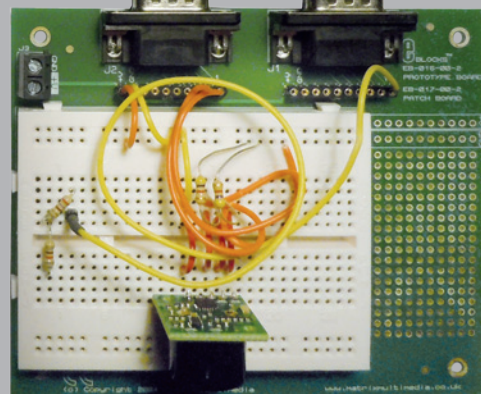
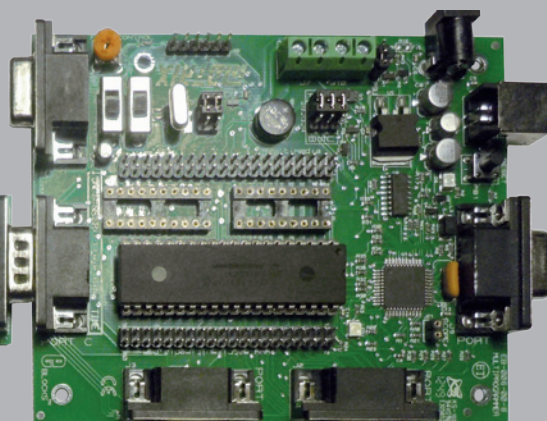


Fig. 3b
Prototipo del nodo display.

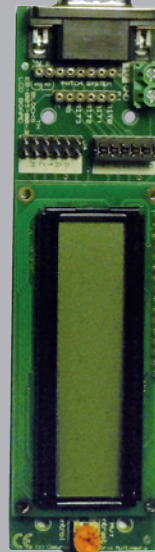


Tabella 3a - Caratteristiche messaggio 0xC40A5.

ID	0xC40A5			Type	Periodico		
Tx	Nodo_Sensore			Rx	Nodo_Display		
Periodo	250ms						
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
DistHighByte	DistLowByte	DiagSts	Not Used	Not Used	Not Used	Not Used	Not Used
Distanza in cm		0 = No fail 1 = Battery fail 2 = I²C fail 3 = Mult. fail					

Tabella 3b - Caratteristiche messaggio 0xC40A4.

ID	0xC40A4			Type	Ad evento		
Tx	Nodo_Display			Rx	Nodo_Sensore		
Periodo	NA						
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
SystemSts	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
0 = Sleep Mode 1 = Awake Mode							

La configurazione hardware dei due nodi è riportata nelle Fig. 3a e Fig. 3b.

Mappa Messaggi

Prima di passare ai dettagli software analizziamo un punto fondamentale di ogni rete basata su un protocollo CAN, ossia la mappa messaggi: in un CAN Bus, essa è il documento che descrive le principali caratteristiche dei messaggi che si scambiano i vari nodi all'interno della network. In questo documento sono indicati le tipologie dei messaggi, i relativi indirizzi, il significato dei byte che li compongono, quali nodi trasmettono e quali sono i destinatari di ogni singolo messaggio. Nel nostro caso, abbiamo un totale di due messaggi, uno trasmesso dal nodo sensore e ricevuto dal nodo display, ed uno che transita nel verso opposto. In Tabella 3a e Tabella 3b sono riporta-

te le caratteristiche dei due messaggi. Passiamo ora alla descrizione dell'implementazione software, con la premessa che, data la complessità del progetto, la descrizione di alcune parti non fondamentali verrà omessa o sarà trattata solo marginalmente.

Nodo Sensore

Secondo le specifiche, questo nodo deve acquisire i dati relativi alla distanza dal sensore ad ultrasuoni, effettuare l'auto-diagnosi su livello batteria e stato del bus I²C e comunicare queste informazioni al nodo master tramite messaggio CAN.

Tabella 4 - Periodicità e descrizione dei task del software del nodo sensore.

Task	Periodo [ms]	Descrizione
CanTxTask	250	Gestisce la trasmissione del messaggio CAN del nodo sensore.
CanRxTask	50	Gestisce la ricezione del messaggio CAN dal nodo display.
DiagTask	50	Gestisce la diagnostica del sistema.
SensorRangingTask	10	Gestisce la comunicazione con il sensore ad ultrasuoni SRF02.

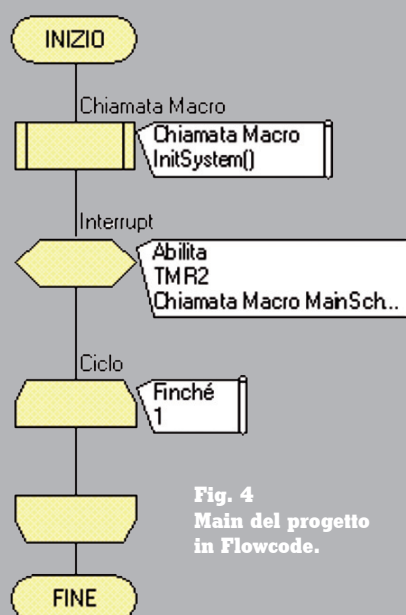
**Fig. 4**
Main del progetto
in Flowcode.

Fig. 5 - Configurazione dell'interrupt del timer2.

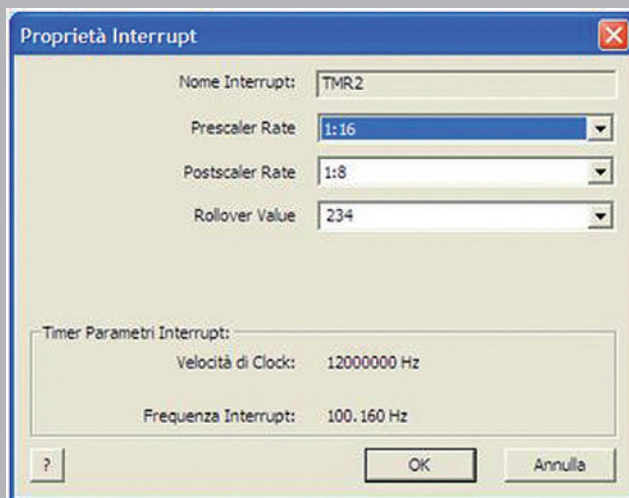
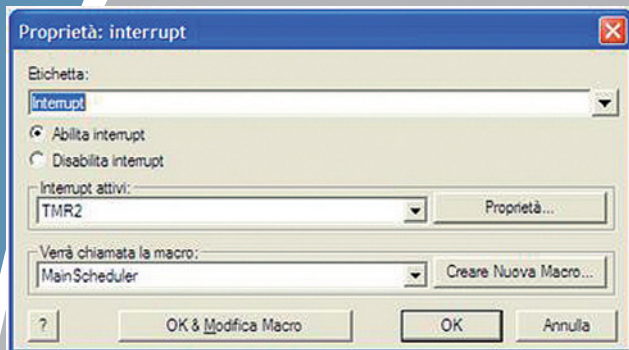


Fig. 6 - Impostazione dei registri del timer 2.

Deve inoltre essere gestita una modalità sleep, su richiesta del nodo master. Per cominciare, esaminiamo l'impostazione dell'architettura di base del sistema. Il firmware in questione ha diversi task da eseguire in maniera simultanea; scegliamo quindi di servirci di uno scheduler per gestire le chiamate ai vari task. Decidiamo di generare uno scheduler con un tick a 10ms e di generare le altre periodicità attraverso dei contatori incrementati ad ogni tick di sistema. In **Tabella 4** sono riportati i task gestiti con le descrizioni e le relative periodicità. Per la generazione del tick base di sistema (sul quale è basato il funzionamento dello scheduler), ci serviamo di un interrupt sul *timer 2*. Come detto nella seconda puntata di questo corso, per inserire un blocco interrupt trasciniamo dalla barra delle icone un blocco interrupt all'interno del nostro main, come visibile in **Fig. 4** (nello schema a blocchi è già stata effettuata la chiamata alla funzione di inizializzazione per i componenti utilizzati).

Per configurare l'interrupt clicchiamo due volte all'interno del blocco, in modo da accedere alle finestre di configurazione, riportata in **Fig. 5**.

In questa finestra selezioniamo l'opzione "Abilita Interrupt", e tra le sorgenti del menu a tendina selezioniamo la voce "TMR2". Poi, nella sezione "Verrà chiamata la macro", inseriamo la macro

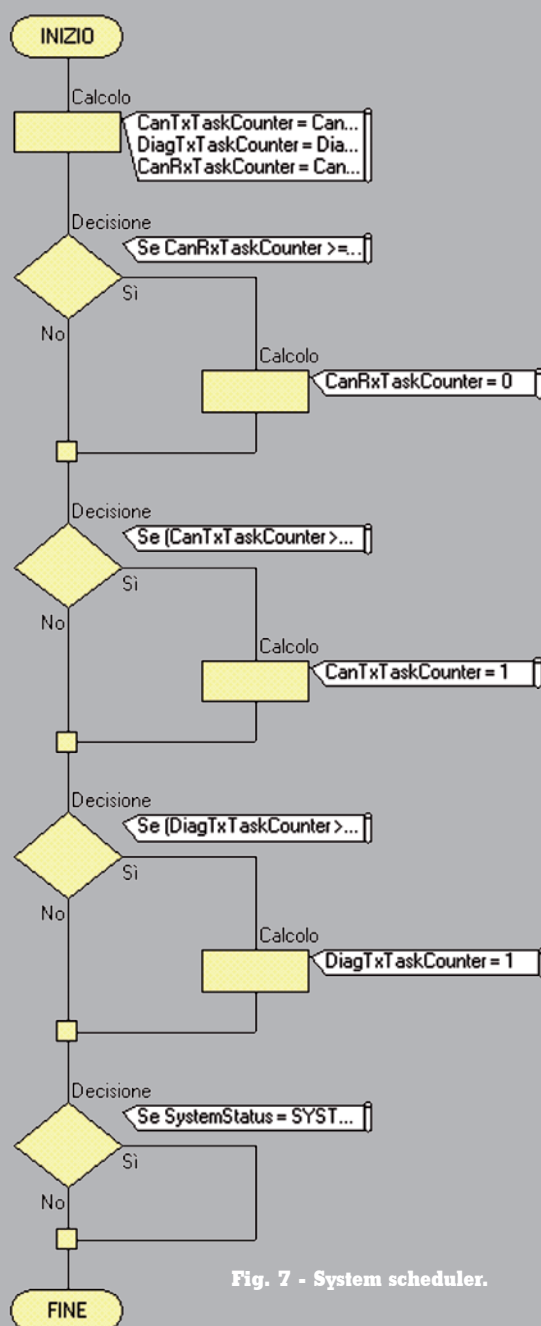


Fig. 7 - System scheduler.

Tabella 5 - Registri interni del sensore SRF02 in modalità I²C.

Localizzazione di memoria	Accesso in lettura	Accesso in scrittura
0	Software Revision	Command Register
1	Unused (reads 0x80)	N/A
2	Range High Byte	N/A
3	Range Low Byte	N/A
4	Autotune Minimum - High Byte	N/A
5	Autotune Minimum - Low Byte	N/A

MainScheduler (che possiamo creare al momento o aver già creato in precedenza). In questo modo, ad ogni occorrenza dell'interrupt verremo indirizzati alla macro contenente lo scheduler di sistema. Non ci rimane che impostare il timer per avere il tick di sistema a 10ms; allo scopo, clicchiamo sul tasto "Proprietà..". Si aprirà la finestra di Fig. 6.

Per avere un interrupt ogni 10ms occorre generare una frequenza di circa 100Hz, per ottenere la quale, con un sistema che gira a 12MHz, selezioniamo un prescaler di 16, un postscaler di 8 e un rollover di 234. Clicchiamo su OK e procediamo. Ora abbiamo il nostro tick di sistema configurato; ciò che rimane da fare è generare il codice dello scheduler in modo da avere le periodicità elencate in Tabella 3 (a e b). Per farlo, generiamo tre variabili globali, ossia *CanRxTaskCounter*, *CanTxTaskCounter* e *DiagTxTaskCounter*, che utilizzeremo come contatori per determinare le periodicità dei task a partire

dal tick base di sistema. Ricordiamoci che alcuni task sono vincolati ad andare in sleep su richiesta del nodo master, quindi in questa condizione non devono essere eseguiti. Per implementare tale caratteristica creiamo una variabile globale che chiamiamo *SystemStatus* (essa può assumere due valori: *SYSTEM_ACTIVE* e *SYSTEM_NOT_ACTIVE*), che sarà, insieme alla periodicità, una delle condizioni di attivazione della chiamata al task. L'unico task che non è controllato da questa variabile è *CanRx*, usato per consentire il risveglio del sistema da messaggio CAN. L'implementazione dello scheduler (senza le chiamate ai task per il momento) è descritta nella Fig. 7.

I contatori sono incrementati ad ogni ciclo e le condizioni di attivazione dei vari task (all'interno dei blocchi decisione) sono:

- *CanRx* → *CanRxTaskCounter* >= *TASK_CAN_RX_TIMEOUT*;
- *CanTx* → (*CanTxTaskCounter* >= *TASK_CAN_TX_TIMEOUT*) && (*SystemStatus* = *SYSTEM_ACTIVE*);
- *Diag* → (*DiagTxTaskCounter* >= *TASK_DIAG_TIMEOUT*) && (*SystemStatus* = *SYSTEM_ACTIVE*);
- *SensorRanging* → *SystemStatus* = *SYSTEM_ACTIVE*.

Come si può vedere, il task *CanRx* non è vincolato alla variabile *SystemStatus* per garantire la possibilità di risvegliare il sistema tramite messaggio CAN, ed il task *SensorRanging* non ha un contatore per la periodicità, in quanto il periodo di chiamata di questo task è proprio 10ms. Analizziamo adesso come è stata implementata la lettura del sensore ad ultrasuoni: per interfacciarsi con il dispositivo che lo utilizza, l'SRF02 ha la capacità di utilizzare sia l'interfaccia seriale che l'I²C. Nel nostro caso, come interfaccia di comunicazione abbiamo scelto l'I²C.

Tabella 6 - Set di comandi dell'SRF02.

Comando (Hex)	Descrizione
0x50	Esecuzione di un ranging - risultato in pollici.
0x51	Esecuzione di un ranging - risultato in cm.
0x52	Esecuzione di un ranging - risultato in microsecondi.
0x56	Esecuzione di un fake ranging - risultato in pollici.
0x57	Esecuzione di un fake ranging - risultato in cm.
0x58	Esecuzione di un fake ranging - risultato in microsecondi.
0x5C	Trasmissione di 8 cicli a 40kHz (ultrasonic burst).

Tabella 7 - Macro scritte per la gestione dell'SRF02.

Nome Macro	Descrizione
<i>Sensor_SendCommand</i>	Invia al dispositivo con indirizzo 0xE0 il comando 0x51 in scrittura.
<i>Sensor_RequestReadResult</i>	Invia al dispositivo con indirizzo 0xE0 la richiesta di lettura a partire dall'indirizzo 2.
<i>Sensor_ReadResult</i>	Legge i byte all'indirizzo 2 (Range High Byte) e 3 (Range Low Byte) e chiude la transazione.

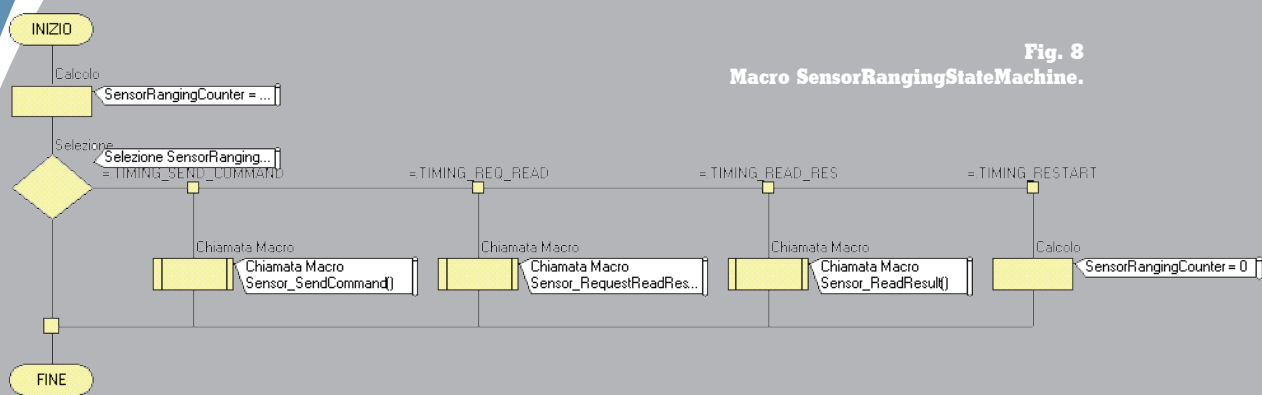


Fig. 8
Macro SensorRangingStateMachine.

Nel caso di interfacciamento di tipo I²C, l'SRF02 funziona in modo molto simile ad una memoria EEPROM della famiglia 24xx: il dispositivo si presenta come un set di registri con accesso in lettura o in scrittura, come riassunto dalla **Tabella 5**. L'accesso in scrittura alla locazione 0 permette di eseguire una delle operazioni riportate in **Tabella 6** (sono riportati solo i comandi principali).

Nel nostro caso utilizziamo i comandi di ranging, che eseguono sia l'operazione di invio del burst ad ultrasuoni che la successiva rilevazione del tempo di volo dell'eco ed il conseguente calcolo della distanza. In questo caso il risultato del ranging viene salvato nei registri 2 e 3 (Range High Byte e Range Low Byte), dai quali può successivamente essere recuperato con un'operazione di lettura. Oltre al comando, sul bus deve essere inviato l'indirizzo del dispositivo (quello predefinito in caso di interfaccia I²C è 0xE0) ed il bit di read/write. Quindi, per ogni trasmissione viene inviato un totale di due byte. La procedura per ottenere il dato sulla distanza è la seguente:

1) invio del comando 0x51 in scrittura (ri-

- chiesta di ranging con risultato in cm);
- 2) invio dell'indirizzo del registro contenente il risultato in lettura;
- 3) ricezione dei due byte del registro contenente il risultato.

Per realizzare queste tre operazioni sono state scritte tre macro, riportate nella **Tabella 7**.

Queste macro sono utilizzate dalla macro SensorRangingStateMachine, che viene chiamata dallo scheduler principale e che fornisce, come output per il resto del sistema, la variabile globale SensorResult più alcune informazioni sulla diagnostica del collegamento I²C. In **Fig. 8** è riportato il diagramma a blocchi Flowcode della macro SensorRangingStateMachine. Questo flowchart implementa una macchina a stati che esegue in sequenza e con le corrette temporizzazioni le tre macro riportate in **Tabella 6**, in modo da ottenere il risultato del ranging. Per farlo, viene utilizzato il contatore SensorRangingCounter, incrementato ad ogni ciclo (ricordiamo che la macro è chiamata direttamente dallo scheduler principale, quindi ha un tempo di chiamata di 10ms).

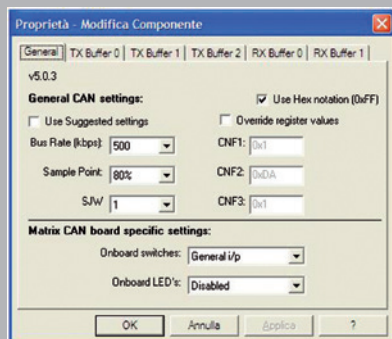


Fig. 9a - Tab general della finestra di configurazione del componente CAN.

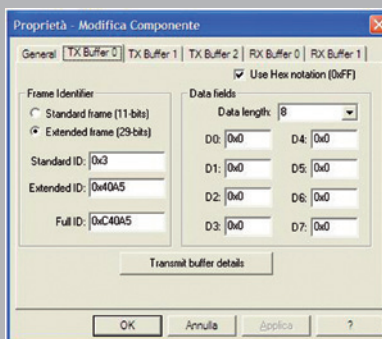


Fig. 9b - Tab tx buffer 0 della finestra di configurazione del componente CAN.

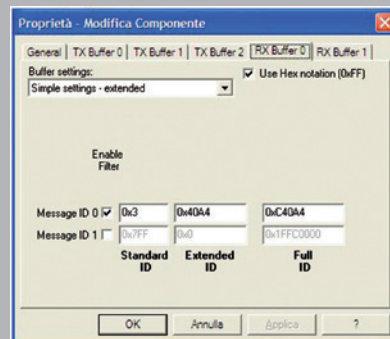
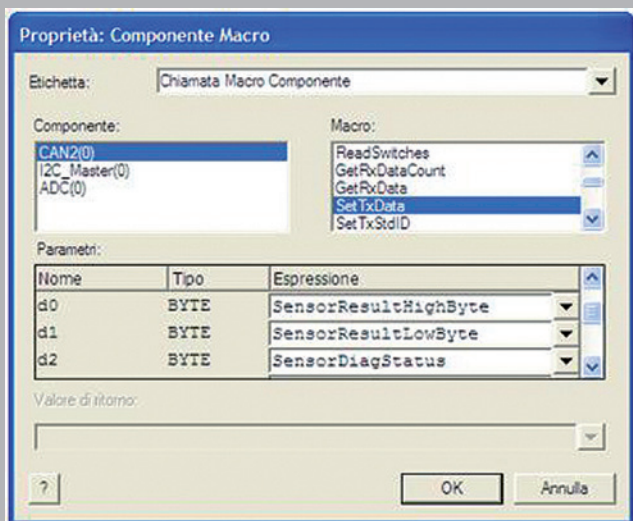


Fig. 9c - Tab rx buffer 0 della finestra di configurazione del componente CAN.



La prima macro chiamata è la Sensor_SendCommand (dopo 10ms dall'avvio della macchina a stati). La seconda chiamata avviene dopo 250ms (ossia quando il contatore raggiunge il valore della costante TIMING_REQ_READ, pari a 25) e viene chiamata la macro Sensor_RequestReadResult. Il ritardo di 250ms serve a fare in modo che l'SRF02 riceva l'eco del burst ad ultrasuoni inviato in seguito alla ricezione del comando 0x51 e faccia i necessari calcoli. A questo punto, dopo altri 100ms viene letto il contenuto dei registri contenenti il risultato, tramite la macro Sensor_ReadResult (contatore uguale a TIMING_READ_RESULT, pari a 35). Infine, dopo altri 150ms la macchina a stati si resetta (semplicemente resettando il contatore) e il ciclo ricomincia dal punto di partenza. In questo modo si ottiene

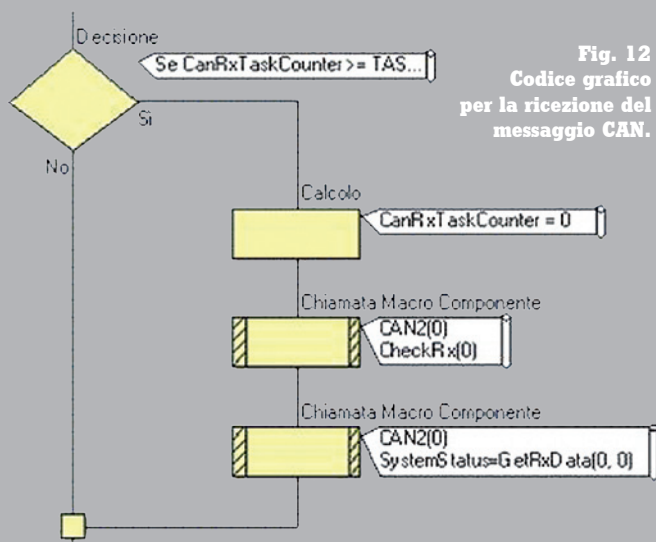
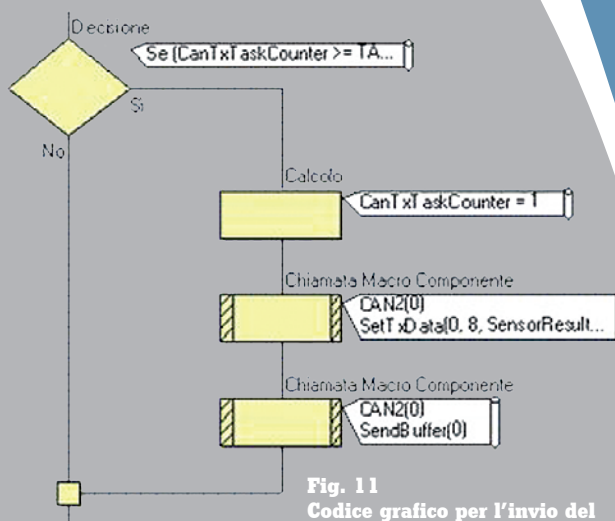


Fig. 10
Proprietà della macro SetTxData del componente CAN.



un aggiornamento della posizione ogni 500ms circa.

Ora che abbiamo compreso la gestione del sensore ad ultrasuoni, esaminiamo come vengono inviati i dati al nodo master tramite il bus CAN. Per prima cosa dobbiamo configurare correttamente il componente, quindi accediamo alla finestra delle proprietà extra. Nel tab principale inseriamo i settaggi principali, selezionando un bau-rate di 500kB, un sample point dell'80% e SJW pari a 1. Passiamo adesso al tab TX Buffer 0, dove, nella sezione frame Identifier, selezioniamo "Extended Frame" ed inseriamo nella casella "Full ID" l'indirizzo 0xC40A5, come riportato nella mappa messaggi. Infine passiamo sul tab RX Buffer 0 e impostiamo un filtro in ricezione, selezionando la spunta "Message ID 0" ed inserendo il seguente ID nella relativa casella "Full ID": 0xC40A4. La configurazione, riportata per comodità in Fig. 9 (a, b e c) è così completata.

A questo punto possiamo inserire, all'interno del nostro scheduler, il codice Flowcode per l'invio del messaggio contenente i dati di distanza e di diagnosi (ID 0xC40A5). L'invio si compone di due macro componente: per prima cosa usiamo la macro SetTxData, per riempire correttamente il messaggio in trasmissione; tale macro prende come parametri il buffer di trasmissione utilizzato, il numero di byte

Tabella 8 - Periodicità e descrizione dei task del software del nodo display.

Task	Periodo [ms]	Descrizione
CanRxTask	500	Riceve e decodifica il messaggio CAN proveniente dal nodo sensore.
DisplayTask	500	Visualizza i dati decodificati dal task CanRxTask.
SwitchTask	200	Legge lo stato del pulsante ed invia il relativo messaggio CAN ad evento per il nodo sensore.

da inviare (DLC) e gli 8 byte del messaggio. I dati da inviare sono le seguenti variabili globali:

- SensorResultHighByte → Byte 0;
- SensorResultLowByte → Byte 1;
- SensorDiagStatus → Byte 2.

Le prime due variabili sono ricavate dalla macro SensorRangingStateMachine, mentre la variabile SensorDiagStatus è calcolata (e codificata come indicato nella mappa messaggi) dalla macro DiagnosticTask. In Fig. 10 è riportata la finestra di proprietà della chiamata alla macro componente.

Una volta riempito correttamente il messaggio in trasmissione, per inviare il messaggio è sufficiente chiamare la macro SendBuffer e passare il buffer di trasmissione. In Fig. 11 è riportato il frammento di codice grafico, così come appare dopo l'inserimento delle chiamate alle macro del componente CAN.

A questo punto la nostra applicazione lato nodo sensore è quasi completa; rimane da gestire l'ingresso/uscita dalla modalità sleep controllata dal messaggio CAN inviato dal nodo display. Analizzando la mappa messaggi, si vede che il parametro che determina lo stato del sistema (sleep o awake) viene impostato dal byte 0 del messaggio con ID 0xC40A4. Bisogna quindi gestire un task di ricezione all'interno del quale viene letto questo byte e viene aggiornata una variabile che usiamo per identificare lo stato del sistema. Il nostro scheduler ci offre già un entry point per la ricezione a 50ms; per completare il codice, introduciamo due chiamate a macro di componente, ed in particolare una CheckRx ed una GetRxData, entrambe sul buffer 0. Il risultato della GetRxData viene salvato all'interno della variabile SystemStatus, la quale, come abbiamo visto in prece-

denza, controlla l'esecuzione di tutti i task (ad eccezione proprio del task CanRx). Il frammento di codice grafico che realizza quanto descritto è riportato in Fig. 12.

Nodo Display

La descrizione del software del nodo sensore è ultimata; passiamo adesso al nodo display, che eredita molti concetti del nodo sensore, come ad esempio lo scheduler (che è praticamente identico). La funzione del nodo display è ricevere il messaggio periodico inviato dal nodo sensore e visualizzare le relative informazioni sul display LCD. Inoltre è il nodo che può mandare in sleep o risvegliare l'intero sistema, rilevando la pressione di un tasto sulla EB018. Il software del nodo display è composto da tre task principali, elencati in Tabella 8.

La filosofia di gestione del sistema è molto simile a quanto visto per il nodo precedente. A titolo di esempio, descriviamo come viene implementato il task CanRxTask; questo task ha il compito di ricevere e decodificare il messaggio con ID 0xC40A5, proveniente dal nodo sensore. Per prima cosa è stato configurato il componente CAN in maniera analoga a quanto fatto per il nodo sensore, con la differenza che l'ID del messaggio in trasmissione in questo caso è 0xC40A4, mentre il filtro di ricezione viene impostato sull'ID 0xC40A5. La ricezione avviene in maniera analoga a quanto visto nel caso precedente; in particolare, viene chiamata la CheckRx e per tre volte la GetRxData, per acquisire i due byte di dati che costituiscono l'informazione sulla distanza ed il byte contenente le informazioni diagnostiche. Infine, il byte 0 ed il byte 1 del messaggio vengono compattati in una word, tramite il seguente codice inserito in un blocco calcolo:

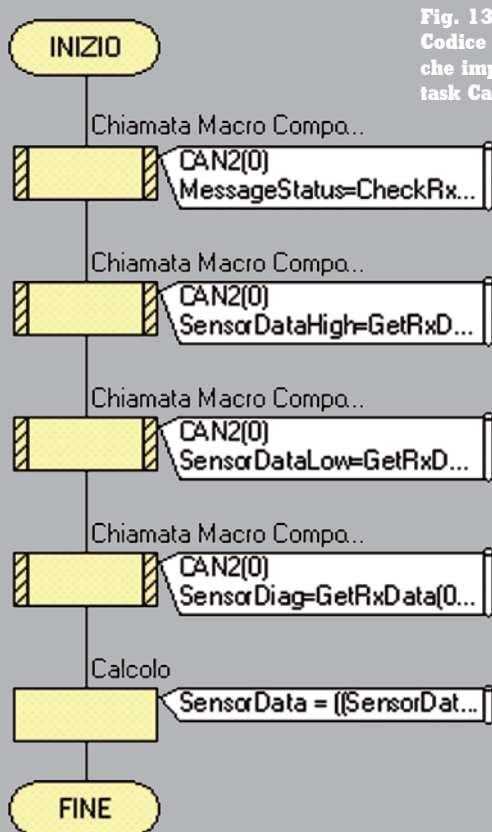


Fig. 13
Codice grafico
che implementa il
task CanRxTask.

ad analizzare il progetto Flowcode e cercare di capirne il funzionamento. A questo punto possiamo anche testare il nostro sistema, collegando tra di loro i due nodi con un doppino intrecciato ed alimentando il sistema con una tensione di circa 12V.

Sul nodo display vedremo apparire le informazioni sulla distanza dal sensore ad ultrasuoni ricevute dal nodo sensore. In Fig. 14 è riportato un tracciato CAN dell'interscambio di messaggi tra i due nodi, eseguito usando il CAN Analyzer Microchip.

Come si può vedere, il messaggio del nodo sensore invia sul bus una distanza rilevata di circa 317 cm (byte 0-1 pari a 0x13D) ed una diagnosi senza fault attivi (byte 2 pari a 0x00). Alla ricezione della richiesta di sleep (byte 0 del messaggio 0xC40A4 pari a 0), il sistema interrompe la trasmissione.

Conclusioni

Concludendo questa puntata, possiamo dire che l'approccio al CAN con Flowcode permette di gestire diverse tipologie di componenti e periferiche con relativa semplicità, oltre a generare codice estremamente facile da comprendere e mantenere.

Nelle prossime puntate esamineremo altre periferiche complesse, come l'USB, lo ZigBee e la porta Ethernet.

$SensorData = ((SensorDataHigh \ll 8) | SensorDataLow)$

Lo schema a blocchi che implementa quanto descritto in precedenza è riportato in Fig. 13.

I task per la gestione del display e per la lettura del tasto di sleep non presentano particolarità implementative di rilievo e non saranno trattati nel dettaglio in questa puntata; vi invitiamo, come esercizio,

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)
RX	0xC40A4x	4	0x00	0x00	0x00	0x00					1921,5274	0,151
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1921,3763	0,250
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1921,1264	0,249
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1920,8773	0,250
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1920,6274	0,249
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1920,3783	0,250
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1920,1284	0,249
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1919,8793	0,250
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1919,6294	0,249
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1919,3803	0,250
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1919,1304	0,249
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1918,8813	0,250
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1918,6314	0,249
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1918,3823	0,250
RX	0xC40A5x	8	0x01	0x3D	0x00	0x00	0x00	0x00	0x00	0x00	1918,1324	0,249

Fig. 14 - Tracciato CAN del sistema durante in funzionamento fino alla ricezione di una richiesta di sleep.