

In omaggio
MICROCONTROLLER PIC a 8 bit
MICROCHIP

Mensile di progettazione elettronica, attualità scientifica, novità tecnologiche.

€ 6,00

Elettronica In

oltre l'elettronica

www.elettronicaonline.it

Anno XXI - n. 196
Giugno 2015

■ Ethernet mp3
il firmware
dell'RTX audio

■ Il mio primo
oscilloscopio
digitale

■ GestIC
incontra
Arduino

■ I dieci anni della
Maker Faire



ANNI

di Elettronica
insieme

■ Da Arduino
a Genuino

■ Radiocontrollo:
prese di rete
con Arduino

■ Robot
Tagliaerba

■ Milliohmometro DIY

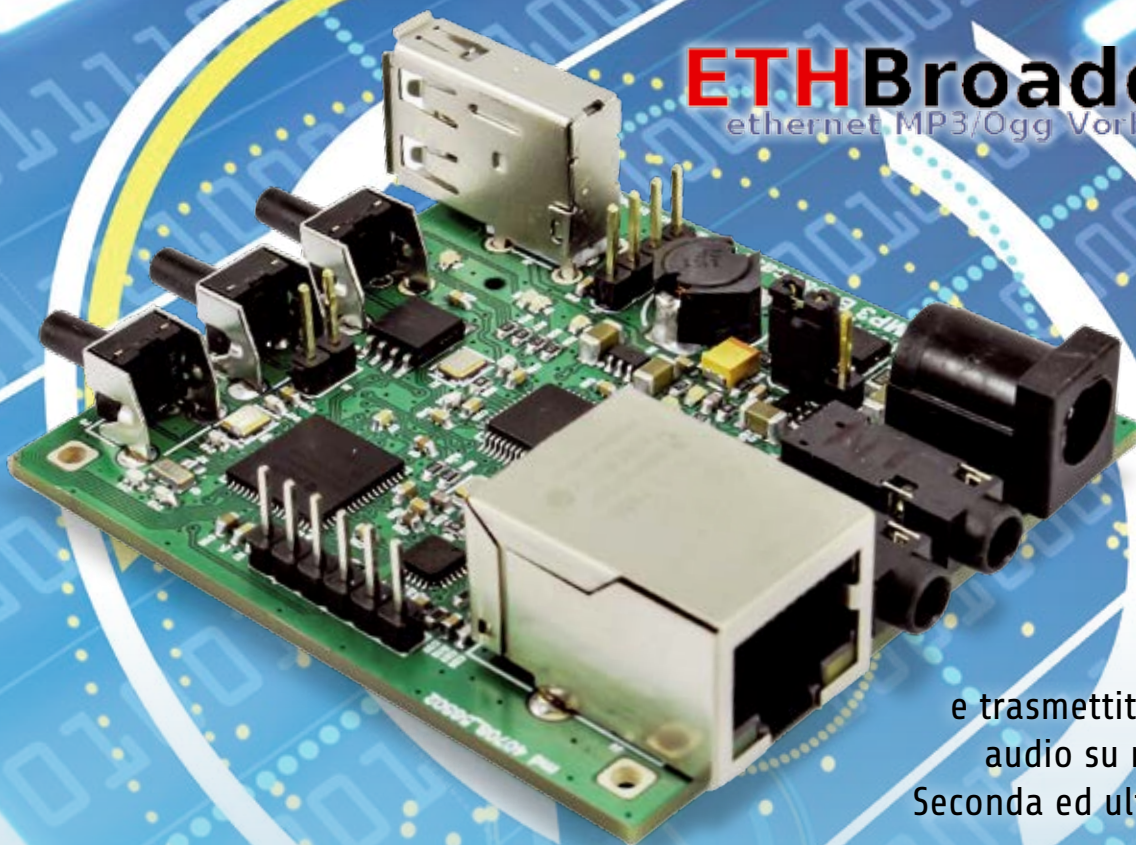
■ Randa
PhotoSharing

■ YUN Shield,
connettività
Internet per
tutte le board
Arduino



ETHBroadcaster

ethernet MP3/Ogg Vorbis streaming



Descriviamo
il firmware
del ricevitore
e trasmettitore di stream
audio su rete ethernet.
Seconda ed ultima puntata.

ETHERNET MP3 OGG VORBIS BROADCASTER

..... • dell'Ing. LUCA PASCARELLA

Nel fascicolo n° 195 abbiamo presentato un'unità per la realizzazione di stream audio su rete ethernet. In particolare, abbiamo mostrato come configurare il circuito per funzionare con altri dispositivi omologhi oppure con VLC Media Player, consentendo la realizzazione di stream punto-punto oppure broadcast in tutte le configurazioni possibili. Inoltre nella prima puntata è stato mostrato il circuito elettrico motivando la scelta dei componenti impiegati e presentato gli scenari d'uso, nonché le diverse modalità di configurazione sia della board che di VLC Media Player. Adesso è giunto il momento di descrivere le opzioni avanzate presenti sulla board, il software e come procedere all'aggiornamento del firmware tramite Internet.

Di seguito facciamo riferimento al programma ETHBroadcaster, già accennato nella prima puntata.

AUTHENTICATION E LOCAL NETWORK

Nella pagina *Authentication* è possibile modificare le proprie credenziali per l'accesso a tutte le aree riservate. Il software prevede due ruoli, ognuno accessibile con una coppia di stringhe: *nome utente* e *password*. Il ruolo con il più alto grado di privilegi è l'amministratore riconosciuto dal software come *Admin*, solo in questo caso è possibile modificare tutti i settaggi della scheda; invece il ruolo con minori diritti è l'utente semplice. Quando alla pagina *Authentication* si accede come amministratore, è presente un *form* come si vede in **Fig. 1**, nel quale è

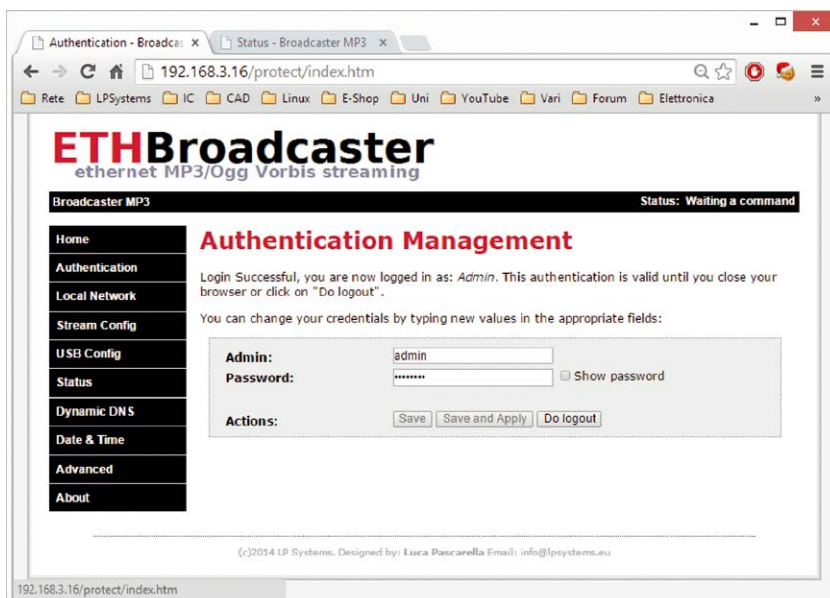


Fig. 1 - Vista della pagina **Authentication Management** che consente di modificare lo username e la password di accesso.

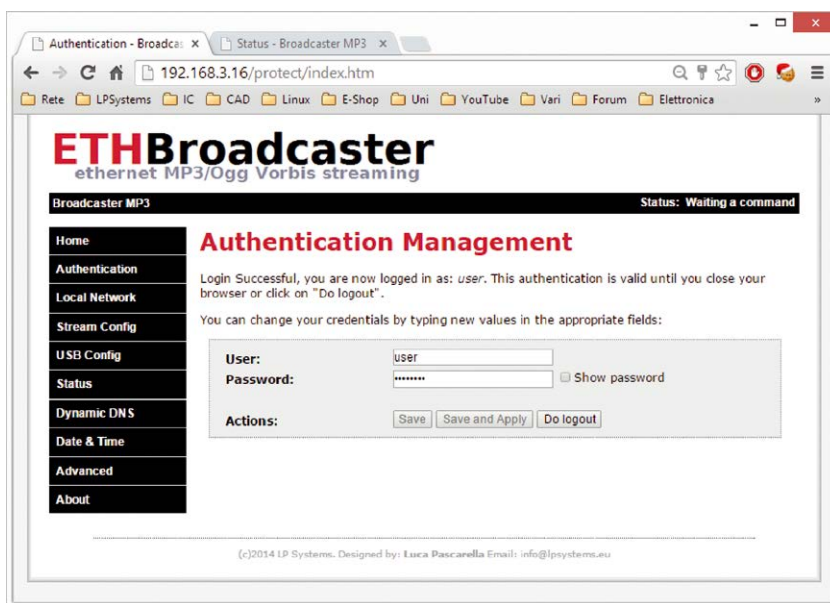


Fig. 2 - Vista della pagina **Authentication Management** nel caso l'accesso avvenga da utente semplice.

possibile cambiare le proprie credenziali; similmente, quando alla pagina si accede come user (come in Fig. 2) è possibile modificare i dati relativi al ruolo utente. Ogni volta che un utente modifica il valore di un campo editabile, i pulsanti *Save* e *Save and Apply* vengono attivati, consentendo il salvataggio delle informazioni modificate. La differenza tra i due comandi sta nel fatto che

mentre il primo (*Save*) consente di salvare i cambiamenti e continuare ad effettuare altre modifiche senza terminare la sessione corrente, il secondo pulsante (*Save and Apply*) effettua anche una chiusura della sessione, richiedendo nuovamente i dati d'accesso a chi desidera proseguire apportando ulteriori modifiche. L'ultimo comando della pagina

Authentication è *Do logout*, che consente di terminare una sessione precedentemente autenticata; lo stesso effetto si ottiene chiudendo il browser.

Dalla pagina *Local Network* di Fig. 3, invece, è possibile configurare le impostazioni di rete: in particolare, si può visualizzare o modificare il MAC Address ed il *Network Basic Input/Output System* della periferica di rete; quest'ultimo, noto come NetBIOS, è un protocollo di livello sessione sviluppato inizialmente da IBM e Sytec, la cui semplicità d'uso lo ha reso oggi uno standard largamente diffuso, soprattutto in ambiente Windows. In realtà non è un protocollo di fondamentale importanza, ma in reti locali, dove gli indirizzi IP sono spesso assegnati dinamicamente dal DHCP server, il suo impiego facilita l'individuazione dei device: infatti è sufficiente effettuare delle ricerche per nome piuttosto che per IP (dinamico).

La procedura d'identificazione dei dispositivi di rete ricorda molto da vicino il *name service* implementato dal DNS e l'idea che sta alla base è molto simile, ovvero offrire un qualche meccanismo in grado di contattare un dispositivo di rete a partire da un nome univoco associato. Nei sistemi Windows, il servizio è spesso attivo per impostazione predefinita, mentre nei sistemi Unix è necessario aggiungere il supporto manualmente.

Concludiamo la descrizione dei comandi disponibili nella pagina *Local Network* con il checkbox *Enable DHCP*, con cui è possibile abilitare il supporto al *Dynamic Host Configuration Protocol* (DHCP): quando il checkbox non è spuntato, la pagina consente la modifica manuale dell'*IP Address*, del *Gateway*, della *Subnet Mask* e dei *server DNS*; invece

quando si opta per l'uso del DHCP (checkbox con la spunta) i parametri non sono modificabili e i campi vengono impiegati per mostrare la configurazione corrente ottenuta tramite DHCP.

USB CONFIG

La pagina USB Config consente di interagire con la chiavetta USB: in particolare, è possibile esplorare il contenuto della memoria di massa connessa al trasmettitore e quindi poter verificare i file presenti al suo interno; una rappresentazione è visibile in Fig. 4.

Inoltre è possibile modificare il nome del file che viene associato alle registrazioni. Il campo *Recording filename* consente di modificare il nome con cui la scheda salverà le registrazioni provenienti dalla rete. Nello stesso *form* è anche possibile scegliere l'azione da compiere nel caso in cui fosse già presente nella chiavetta un file con tale nome. Le azioni possibili sono due e corrispondono alla sovrascrittura del file esistente oppure all'aggiunta di un *time stamp* univoco per ogni registrazione. Il *time stamp* sarà formattato nel modo seguente: *hhmmss_GGM-MYYYY* dove *hh* sta per ora, *mm* sta per minuti, *ss* sta per secondi, *GG* per giorno, *MM* per mese e *YYYY* per anno.

Le azioni associate ai pulsanti *Save*, *Apply* e *Save and Apply* sono rispettivamente salvare le modifiche nella memoria permanente e renderle disponibili solo al prossimo riavvio, applicare le modifiche da subito e non salvarle nella memoria e, infine, salvare ed applicare le modifiche immediatamente.

DYNAMIC DNS

Il *Dynamic DNS* (abbreviato in *DDNS*) è una tecnologia derivata

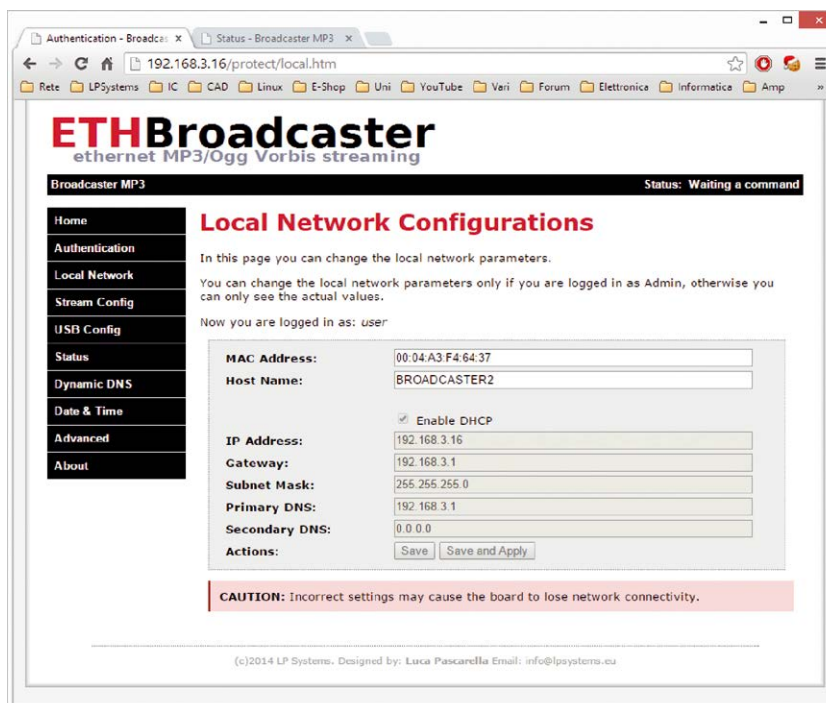


Fig. 3 - Vista della pagina *Local Network Configurations* che consente di modificare le impostazioni relative alla rete LAN.

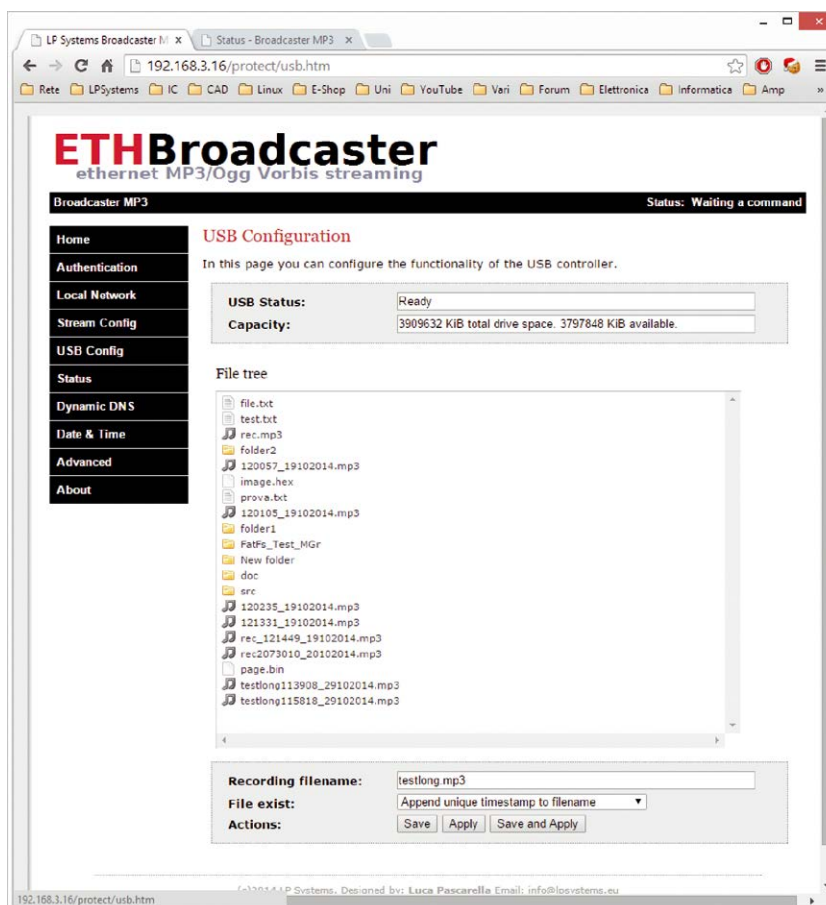


Fig. 4 - Vista della pagina *USB Configuration* e del contenuto di una chiavetta USB collegata alla board.

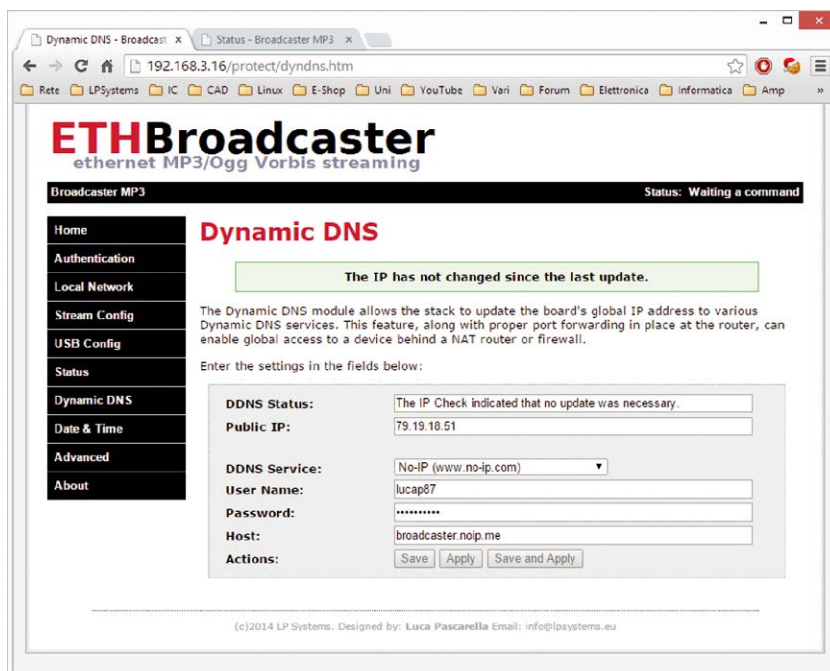


Fig. 5 - Vista della pagina **Dynamic DNS**, in questa pagina è possibile aggiungere un servizio di DDNS.

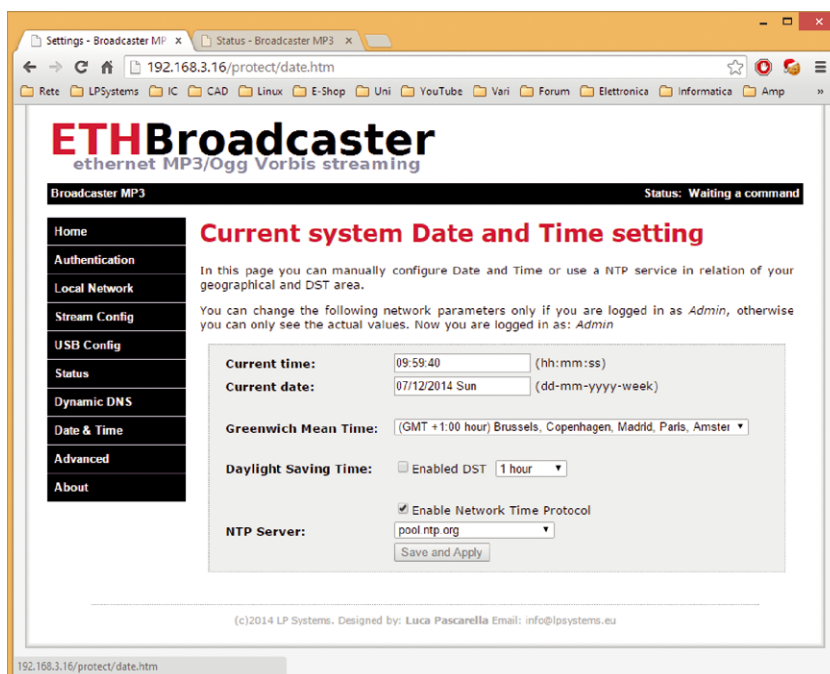


Fig. 6 - Vista della pagina **Date & Time**, in questa pagina è possibile configurare il servizio NTP per avere sempre l'orario della board aggiornato.

dal Domain Name System (DNS), che consente di associare ad un indirizzo IP i nomi della rete o nomi di dominio. La differenza principale che c'è tra i due servizi è che mentre il DNS è gestito a livello mondiale tramite un mec-

canismo di database distribuito e l'inserimento di un nuovo record all'interno del DB è regolato da opportuni meccanismi autorizzativi che rendono l'assegnazione "statica", il DDNS implementa più o meno le stesse funzionalità

per dispositivi che sono soggetti ad un cambio di IP frequente, come possono essere le connessioni ADSL casalinghe. Questa tecnica consente di bypassare le problematiche degli IP dinamici semplicemente impiegando uno dei tanti servizi di DDNS (alcuni dei quali gratuiti).

Per far sì che ci sia sempre la giusta corrispondenza, il dispositivo sul quale gira l'applicazione DDNS client deve contattare il server remoto e aggiornare l'entry nel database con il nuovo valore di IP assegnato in quel momento.

Per poter rendere sempre la board disponibile da remoto, anche se il fornitore di servizi Internet cambia continuamente l'IP assegnato, si può cliccare sulla voce di menu **Dynamic DNS** ed inserire i parametri per uno dei tre servizi di DDNS disponibili: la Fig. 5 propone un esempio in cui è associato l'host name *broadcaster.noip.me* alla board in prova.

Nel campo public IP, come visibile anche dalla figura, viene mostrato l'IP pubblico che viene utilizzato per accedere ad Internet. Qualora la board fosse attiva all'interno di una LAN domestica, occorrerà configurare all'interno del proprio router il *port forwarding*, ovvero bisognerà dire al servizio di DNAT che quando le richieste provenienti dall'esterno della rete LAN coincidono con un particolare numero di porta preassegnato, anziché essere droppate dovranno essere ridirezionate all'host di destinazione inserito nella regola di routing.

DATE & TIME

La scheda che abbiamo realizzato implementa un orologio e un calendario hardware: cliccando sulla voce di menu **Date & Time** è

L'interfaccia ethernet LAN8720

Il chip utilizzato per realizzare l'interfaccia di rete del nostro progetto è un LAN8720 realizzato dalla SMSC. Si tratta di un PHY Ethernet che si colloca all'ultimo livello dello stack ISO/OSI ed implementa le funzionalità del layer fisico, ovvero è il componente che si interfaccia con i segnali elettrici di una rete Ethernet e ne traduce i livelli logici in stringhe di bit. Tipicamente il PHY si interfaccia con un altro componente di più alto livello chiamato MAC (*Media Access Control*). Le specifiche prevedono tre modalità di interfacciamento standard: la MII, l'RMII e la GMII. Le differenze principali che ci sono tra le prime due modalità risiedono nel numero di I/O necessari alla comunicazione e nelle frequenze in gioco, che tipicamente sono multiple di 25 MHz; invece GMII e successive sue varianti con numero di linee ridotte, servono per poter effettuare gli stessi collegamenti logici ma per connessioni Gigabit. Nel caso del LAN8720, essendo questo un componente dalle dimensioni estremamente ridotte (solo 24 pin in un QFN da 4x4 mm) è disponibile la sola modalità RMII; quest'ultima torna utile anche per via del ridotto numero di interconnessioni necessarie, a patto di usare una frequenza di clock (50 MHz) doppia della MII (25 MHz).

Questo componente, per poter funzionare correttamente necessita di un'unica alimentazione a 3,3 V, il che lo rende un'alternativa apprezzata nei progetti embedded. Dal punto di vista del collegamento Ethernet il chip implementa l'*Auto-MDIX* consentendo di utilizzare comuni cavi di rete in qualsiasi configurazione, infatti sarà l'architettura interna a scegliere quale coppia di cavi intrecciati usare per la trasmissione e quale per la ricezione. Infine, non di minore importanza, è il PLL interno che consente di usare come sorgente di clock

un comune cristallo al quarzo da 25 MHz per ottenere i 50 MHz richiesti dall'interfaccia RMII. Molti altri chip PHY RMII disponibili sul mercato necessitano di un clock esterno da 50 MHz il che rende il progetto tipicamente più costoso, in quanto bisogna ricorrere all'uso di un oscillatore quarzato. In **Fig. A** è riportato uno schema logico d'interconnessione del PHY LAN8720, mentre in **Fig. B** viene riportato un esploso delle linee di comunicazione impiegate tra PHY e MAC.

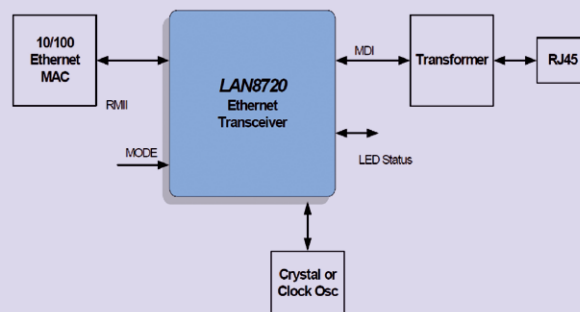


Fig. A - Schema a blocchi del chip LAN8720.

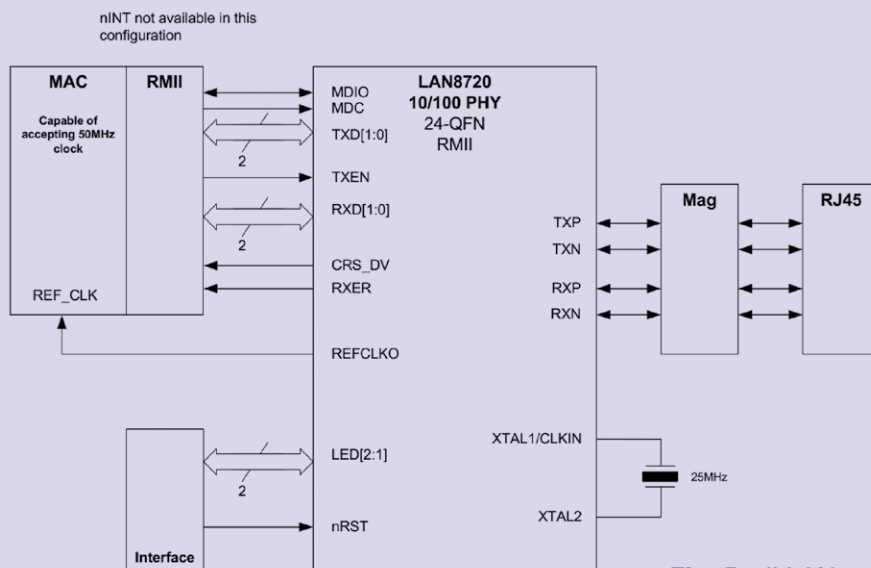


Fig. B - Il LAN8720 genera i 50 MHz necessari al pilotaggio del MAC tramite RMII.

possibile accedere a queste informazioni e configurare l'orologio. Nei primi due campi (*current time* e *current date*) sono riportati i valori di tempo e orario presenti in quel momento nell'RTC della scheda. Inoltre, come visibile dalla **Fig. 6**, è possibile selezionare il fuso orario locale, nel caso dell'Italia GMT+1, ed abilitare l'ora legale (*Daylight Saving Time*).

Infine è possibile scegliere di inserire manualmente l'orario corretto o decidere di affidare l'aggiornamento dell'orario alla rete sfruttando il protocollo *Network Time Protocol* (NTP); in quest'ultimo caso, dato che la scheda non è dotata di una batteria tampone, l'NTP si rivela particolarmente efficace per mantenere l'orario corretto ad ogni

riavvio. La spunta del *checkbox Enable Network Time Protocol* consentirà di selezionare il server NTP, oppure di inserire manualmente l'orario e la data.

FUNZIONALITÀ AVANZATE

Nella voce di menu *Advanced* sono disponibili alcune funzionalità avanzate, come la lettura delle statistiche, il *reboot* della

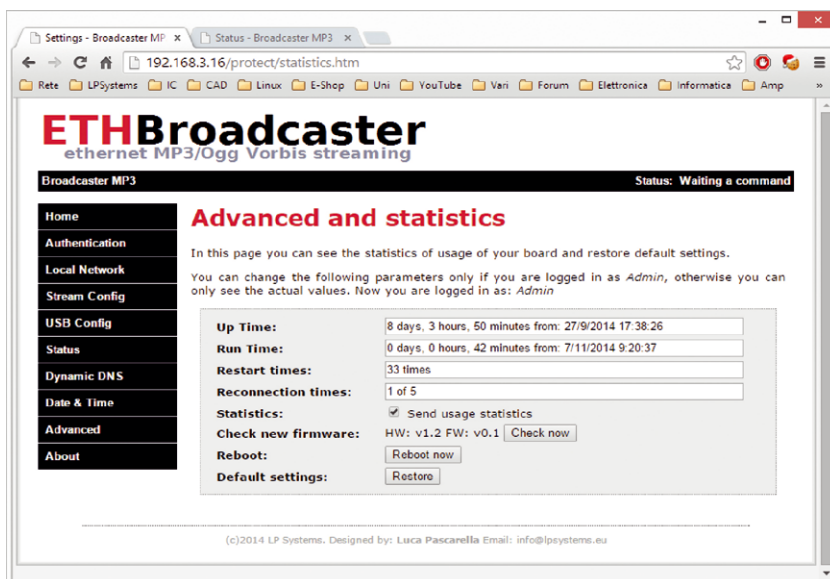


Fig. 7 - Vista della pagina **Advanced and statistics**, in questa pagina è possibile visualizzare le statistiche d'uso della scheda ed eventualmente aggiornare il firmware della scheda.

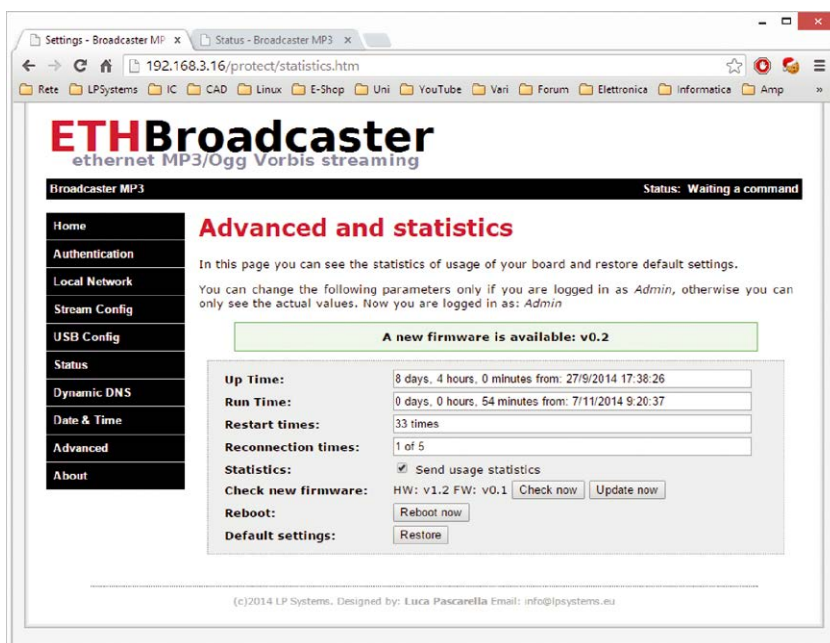


Fig. 8 - La pagina **Advanced and statistics** mostra il messaggio che un nuovo firmware è disponibile per l'aggiornamento della board.

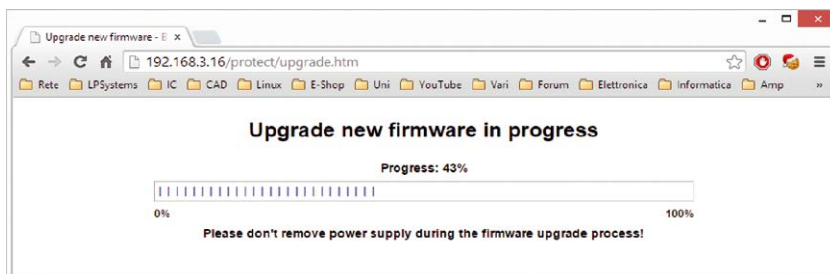


Fig. 9 - La pagina mostra lo stato d'avanzamento dell'aggiornamento del firmware.

scheda, il ripristino dei dati di fabbrica, il controllo e l'aggiornamento del firmware.

Nella parte alta della pagina, come si vede nella Fig. 7, vengono mostrate quattro voci relative alla statistiche. *Up Time* indica il tempo totale di funzionamento espresso in giorni, ore e minuti dall'ultimo azzeramento delle statistiche o dall'ultimo aggiornamento software, questo valore viene aggiornato ogni 10 minuti, frazioni di tempo inferiori vengono ignorate per prevenire il degrado della memoria Flash. *Run Time* indica il tempo di accensione della scheda dall'ultimo riavvio, questo valore è aggiornato ogni minuto di funzionamento essendo un valore presente solo in RAM può essere aggiornato più frequentemente. *Restart times* indica il numero di volte che la scheda è stata riavviata ed infine *Reconnection times* indica il numero di volte (eventualmente il limite superiore) di tentativi di riconnessione. Il valore fa riferimento al numero di tentativi di riconnessione automatica ed è azzerato ogni qual volta la scheda è spenta o si procede con la connessione manuale. Il checkbox *send statistics* abilita le statistiche e l'invio dei dati raccolti al server remoto.

In corrispondenza della voce *check new firmware* viene riportata la versione di firmware e dell'hardware corrente e la possibilità, tramite la procedura descritta nel prossimo paragrafo, di aggiornare il firmware del trasmettitore. Infine, le ultime due voci consentono di riavviare la scheda (in circa 3 secondi) e di resettare i dati di fabbrica. Il reset ai valori di fabbrica può avvenire anche tenendo premuti per più di quattro secondi i pulsanti *left* e *right* all'accensione. Lo stato è segnalato con un lam-

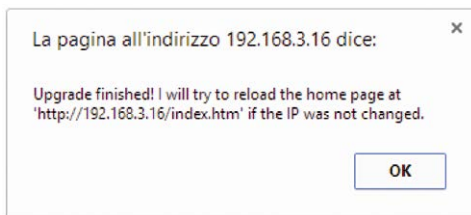


Fig. 10 - Il messaggio indica che il firmware è stato aggiornato con successo.

peggio molto veloce del led blu e del led rosso centrale. Quando il LED blu torna a lampeggiare regolarmente la scheda è di nuovo operativa.

AGGIORNAMENTO DEL FIRMWARE

Per cercare di rendere il progetto longevo abbiamo pensato di realizzare il software in due parti, quindi scomponendolo in un bootloader che viene caricato in fabbrica con l'apposito programmatore In Circuit Serial Programming (ICSP) e in una app che racchiude le funzionalità reali associate al circuito elettronico. Il bootloader viene caricato in un'apposita area di memoria del microcontrollore e non subirà modifiche a meno di non intervenire nuovamente con il programmatore, invece l'app è installata in un'area di memoria riscrivibile dal bootloader, il che consente di effettuare degli aggiornamenti del firmware senza la necessità di disporre dell'apposita strumentazione.

In questa sezione vi descriveremo come aggiornare il firmware per aggiungere nuove funzionalità o per correggere eventuali difetti. In uno scenario d'uso normale, è sufficiente accedere tramite browser all'apposita pagina di configurazione cliccando sulla voce di menu Advanced, questa pagina mostra la versione di firmware e di hardware corrente, come visibile in Fig. 7.

Cliccando sul pulsante *Check now* si avvierà la procedura di verifica della disponibilità di una nuova

versione di firmware. Questa procedura contatterà il server remoto ed in base alla versione di hardware cercherà eventuali aggiornamenti. Nel caso in cui risultasse disponibile un aggiornamento comparirà un messaggio come in Fig. 8 e inoltre apparirà il pulsante con la scritta *Update now*. Prima di procedere con l'aggiornamento bisognerà inserire una USB-Pen formattata con File System FAT16 o FAT32 e con spazio sufficiente di almeno 10 MByte, dopodiché sarà possibile procedere con l'aggiornamento cliccando sull'apposito pulsante *Update now*.

La procedura di aggiornamento del firmware può richiedere alcuni minuti ed è importante che in questa fase non venga né rimossa la chiavetta USB né staccata l'alimentazione elettrica, altrimenti si rischia di compromettere la fase di aggiornamento. Per essere sicuri che l'aggiornamento non pregiudichi il buon funzionamento della scheda, al termine del download del nuovo firmware il microcontrollore provvederà a calcolare l'MD5 del file scaricato comparandolo con quello ottenuto dal server remoto; solo quando ci sarà cor-

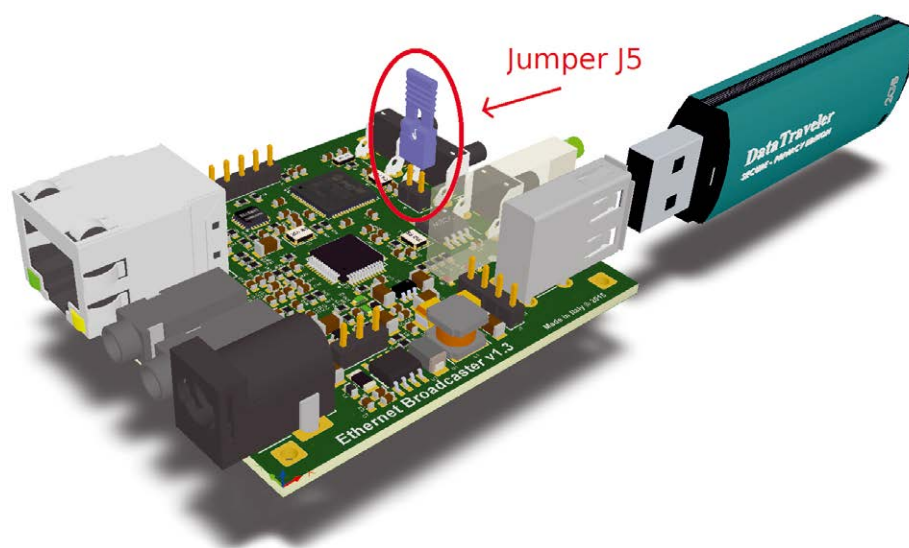


Fig. 11 - Aggiornamento del firmware con procedura manuale.

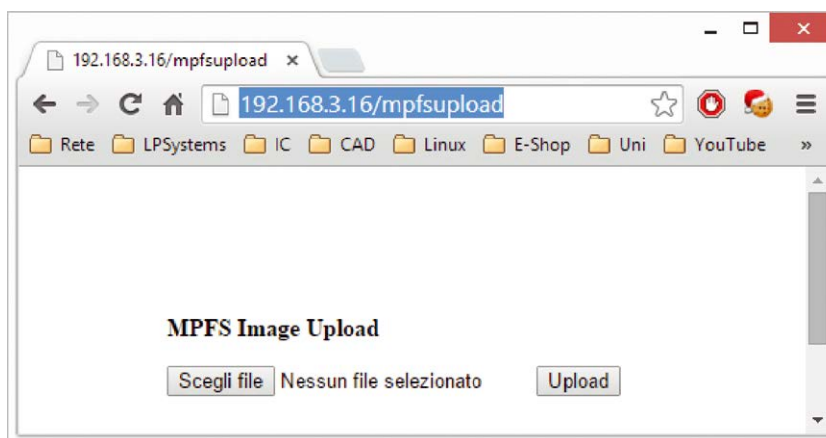
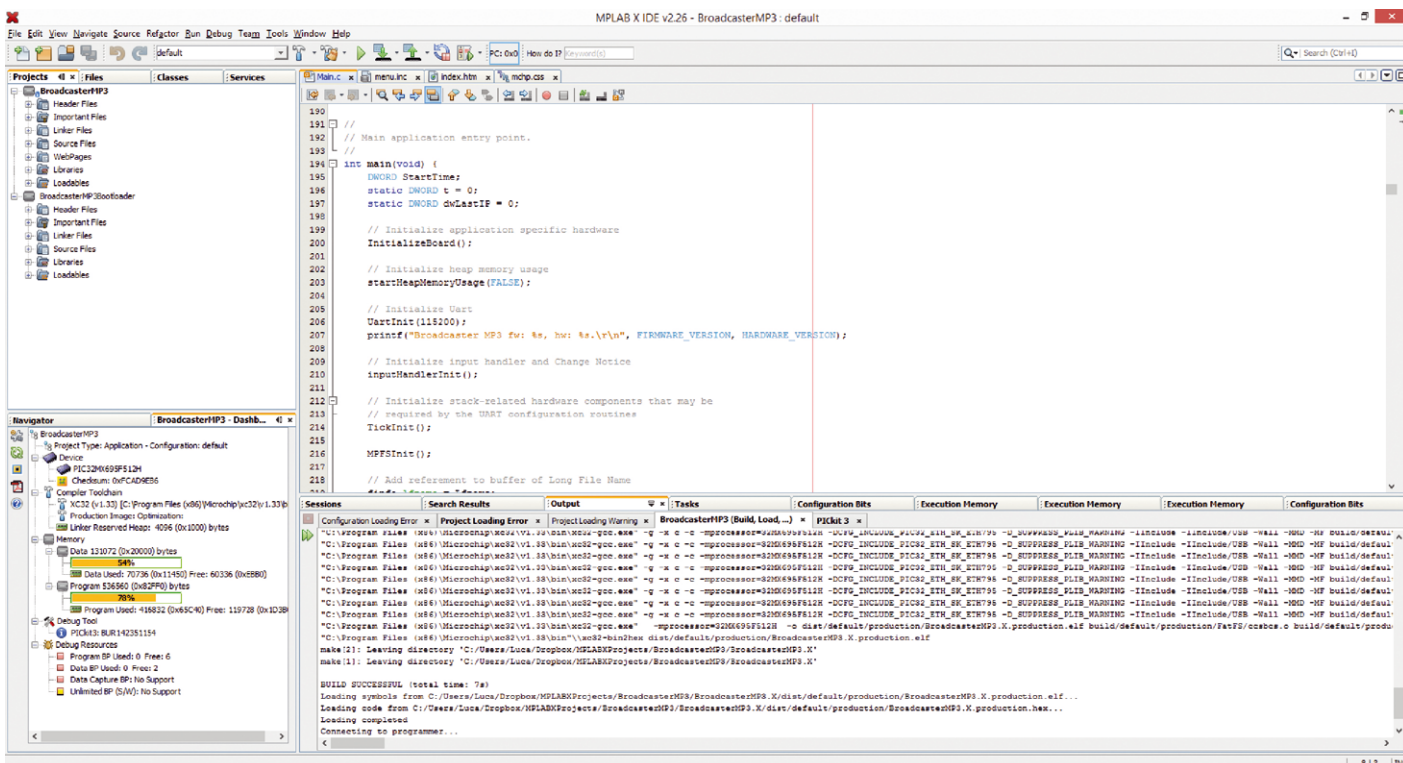


Fig. 12 - Pagina per l'aggiornamento del contenuto della Flash in caso di aggiornamento manuale.

Fig. 13 - Ambiente di sviluppo MPLAB-X, vista del progetto composto dal bootloader e dall'app.



rispondenza tra le due *stringhe di hash*, il software procederà con l'aggiornamento copiando il contenuto del file all'interno della memoria Flash del microcontrollore. In caso contrario, ovvero se qualche bit ha subito alterazioni durante il download del file, il PIC interromperà l'upgrade del firmware e restituirà un messaggio di errore. Durante il download del nuovo firmware nella chiavetta, **Fig. 9**, il microcontrollore continuerà ad essere operativo ed il LED blu continuerà a lampeggiare regolarmente; quando tale fase sarà terminata, il trasmettitore passerà a controllare l'MD5 segnalando l'inizio dell'aggiornamento con l'accensione fissa del LED rosso. Al termine della verifica, se tutto è andato per il verso giusto, il LED rosso inizierà a lampeggiare lentamente; è da questo momento che la procedura di aggiornamento diventa critica ed è importante che fino al completamento non venga interrotta l'alimentazione elettrica.

In questa fase il *bootloader* cancellerà la vecchia app ed in seguito installerà la nuova applicazione. Quando la fase di cancellazione è terminata, il LED rosso rimarrà spento ed inizierà a lampeggiare velocemente il LED rosso centrale. Quando anche quest'ultimo avrà finito di lampeggiare e il LED blu riprenderà a pulsare alla frequenza di 1 Hz, la scheda sarà nuovamente operativa. Durante tutta la fase di aggiornamento la pagina web subirà autonomamente delle evoluzioni che indicheranno lo stato di avanzamento dell'aggiornamento; per avere la certezza che la procedura di aggiornamento è stata completata con esito positivo, conviene attendere la generazione del messaggio mostrato nella **Fig. 10**, che apparirà automaticamente al termine della procedura guidata. Nel caso in cui, invece, non sia più possibile accedere alla pagina web o non sia disponibile una connessione ad Internet, si può procedere all'aggiornamento

manuale del firmware.

In questo caso bisogna innanzitutto impostare i dati di fabbrica, quindi tenendo premuti i pulsanti *left* e *right* all'accensione della board per più di quattro secondi, caricare manualmente il nuovo firmware nella chiavetta USB, rinominandolo esattamente in "*image.hex*"; poi occorre inserire la chiavetta nella scheda spenta dopo il reset dei valori di fabbrica. Ora bisogna inserire un ponticello sul J5 della scheda, come si vede in **Fig. 11**, ed attendere l'aggiornamento, che durerà pochi secondi e sarà indicato dal solito lampeggio dei LED rossi. Terminata questa fase sarà necessario procedere con l'aggiornamento delle pagine web, pertanto bisognerà accedere da browser all'indirizzo IP assegnato dal DHCP; ad esempio, l'URL potrà essere *http://192.168.3.16/mpfsupload*; oppure, più semplicemente, nel caso si utilizzi un browser in ambiente Windows e si abbia abilitato il supporto per il NetBIOS, è

sufficiente andare all'indirizzo <http://MCHPBOARD/mpfsupload>. Dall'apposita pagina, **Fig. 12**, bisogna selezionare il file "page.bin" relativo alla pagine web presenti sul proprio computer e cliccare su *Upload*.

IL SOFTWARE

Di seguito ci occuperemo di descrivere il software e l'ambiente di sviluppo impiegato per la realizzazione del firmware del progetto (app più bootloader). Per quanto riguarda l'ambiente di sviluppo, abbiamo scelto l'Integrated Development Environment (IDE) ufficiale della Microchip, ovvero MPLAB-X, e lo abbiamo aggiornato all'ultima versione disponibile al momento della pubblicazione di questo articolo, vale a dire la v2.26; ricordiamo che il software è liberamente scaricabile direttamente dalla pagina web www.microchip.com/mplabx. Ottenuto il file autoinstallante ed avviata l'installazione, è sufficiente, al termine della procedura guidata, effettuare un import dei nostri progetti, per poter navigare e consultare il codice che abbiamo realizzato per l'app ed il bootloader (**Fig. 13**). Per poter compilare il progetto è invece necessario disporre del compilatore MPLAB-XC32 v1.30 o superiore. Ricordiamo che con l'attuale versione v1.33 è necessario lanciare il gcc con l'opzione -D_SUPPRESS_PLIB_WARNING, allo scopo di nascondere i warning generati dal compilatore che ricordano agli sviluppatori che a partire dalla versioni future di XC32 alcune funzioni di libreria saranno fruibili solo tramite l'impiego del middleware *Microchip Harmony* (disponibile la v1.1 stabile). Il cuore del progetto, però, si basa ancora sull'uso delle MAL,

Listato 1

```
while (1) {
    // Blink LED0 (right most one) every second.
    if (TickGet() - t >= TICK_SECOND / 2ul) {
        t = TickGet();
        invertBlueLED();
    }

    // Stream handlers
    TCPClientHandler();
    UDPClientHandler();

    TCPServerHandler();
    UDPServerHandler();

    // NTP Handler
    UDPClientNTPHandler();

    // Statistics Handler
    statisticsHandler();
    firmwareHandler();

    // Input Handler Task
    inputHandlerTask();

    // This task performs normal stack task including checking
    // for incoming packet, type of packet and calling
    // appropriate stack entity to process it.
    StackTask();

    // This tasks invokes each of the core stack application tasks
    StackApplications();

    #if defined(STACK_USE_SMTP_CLIENT)
    SMTPDemo();
    #endif

    #if defined(STACK_USE_ICMP_CLIENT)
    PingDemo();
    #endif

    #if defined(STACK_USE_TFTP_CLIENT) && defined(WF_CS_TRIS)
    TFTPGetUploadStatus();
    #endif

    #if defined(STACK_USE_BERKELEY_API)
    BerkeleyTCPClientDemo();
    BerkeleyTCPServerDemo();
    BerkeleyUDPClientDemo();
    #endif

    // If the local IP address has changed (ex: due to DHCP lease change)
    // write the new IP address to the LCD display, UART, and Announce
    // service
    if (dwLastIP != AppConfig.MyIPAddr.Val) {
        dwLastIP = AppConfig.MyIPAddr.Val;

        #if defined(STACK_USE_UART)
        putsUART((ROM_char*) "\r\nNew IP Address: ");
        #endif

        DisplayIPValue(AppConfig.MyIPAddr);

        #if defined(STACK_USE_UART)
        putsUART((ROM_char*) "\r\n");
        #endif

        #if defined(STACK_USE_ANNOUNCE)
        AnnounceIP();
        #endif
    }

    //USB stack process function
    USBTasks();

    //if thumbdrive is plugged in
    if (deviceAttached == FALSE) {
        if (USBHostMSDSCSIMediaDetect()) {
            deviceAttached = TRUE;
            // Initialize (Mount) File System structure
            if (f_mount(&Fatfs, "0", 1) != FR_OK)
                FlashLight(250, 50, TRUE);
        }
    }
}
```

Listato 2

```
void __ISR(_CHANGE_NOTICE_VECTOR, ipl2) ChangeNotice_Handler(void) {
    unsigned int tempD;

    // clear the mismatch condition
    tempD = mPORTDRead();

    // clear the interrupt flag
    MCNClearIntFlag();

    buttonLeft = (tempD >> 5) & 0x01;
    buttonCenter = (tempD >> 6) & 0x01;
    buttonRight = (tempD >> 7) & 0x01;
    buttonInternal = (tempD >> 11) & 0x01;

    signalNewInputState = TRUE;
}
```

non più supportate per i microcontrollori a 32 bit; tuttavia abbiamo preferito continuare ad usare le MAL a discapito di Harmony, perché nonostante quest'ultimo sia rilasciato come versione stabile, presenta ancora delle carenze e delle problematiche pratiche, mentre le MAL si rivelano molto robuste. Anche il compilatore, come MPLAB-X, è liberamente scaricabile per l'impiego come versione minimale o trial per la valutazione della versione completa. Per compilare questo progetto è sufficiente la versione minimale, in quanto

non è strettamente necessaria l'ottimizzazione. Detto ciò, passiamo ad analizzare il codice sviluppato per il progetto: andando per ordine troviamo il **Listato 1**, in cui è presente il cuore dell'applicazione; l'intero firmware è realizzato, come avviene in molte applicazioni embedded, con un approccio multitasking cooperativo, ovvero le diverse routine software si contendono la CPU rilasciando quest'ultima solo volontariamente al termine di un task. Tipicamente i task (le parti in cui è divisa una partico-

lare elaborazione) sono realizzati con del codice inserito all'interno di una funzione di handler chiamata regolarmente dalla routine principale, come nel caso del **Listato 1**, dove si possono vedere i differenti handler per ogni compito specifico. Ogni handler implementato suddivide il lavoro in task per mezzo di una struttura a *switch*, dove ognuno dei case rappresenta un sotto-compito da eseguire; ogni volta che termina un task, ovvero si è raggiunta l'istruzione *break*, la CPU viene rilasciata (volontariamente) per tornare nel *while(TRUE)* ed assolvere a un altro compito.

Dato che con l'approccio multitasking cooperativo non è previsto un meccanismo di prelazione, è bene porre molta attenzione alla scrittura del software, perché un eventuale squilibrio in termini di durata dei task può far ottenere dei risultati pessimi; una taratura

Listato 3

```
function TCPServerGetStatus() {

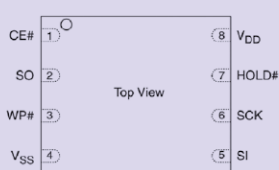
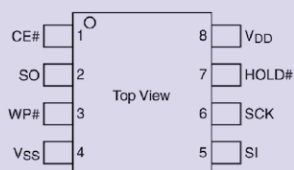
    $.ajax({
        type : "GET",
        url : 'server.xml',
        dataType : "xml",
        cache : false,
        timeout : 30000,
        success : function(data) {
            // Extracts Connect/Disconnect button label
            $('#serverButtonConnection').val(getXMLValue(data, "buttonConnection"));
            if ($('#serverButtonConnection').val() == 'Listening')
                $('#serverButtonConnection').attr("disabled", true);
            else
                $('#serverButtonConnection').attr("disabled", false);
            // Extracts the textual connection status
            $('#connectionStatus').val(getXMLValue(data, "connectionStatus"));
            // Extracts the textual IP address
            $('#serverAddress').val(getXMLValue(data, "address") + ':' + getXMLValue(data, "port"));
            // Set the buffer filling
            $('#serverBufferFilling').val(getXMLValue(data, "bufferFilling"));
            setTimeout("getClientRequest($('#connectionStatus').val())", 2000);
        },
        error : function(jqXHR, textStatus, errorThrown) {
            // Disable button if an error occurs
            $('#serverButtonConnection').val('Connect');
            $('#serverButtonConnection').attr("disabled", true);
            // Insert an error message in connection status
            $('#connectionStatus').val('Server connection error');
            // Set the buffer filling
            $('#serverBufferFilling').val(0);
            //alert("Some error is occurred during the updating of the new values.\nError type: " + textStatus + ".\n Error: " + errorThrown);
        }
    });
};
```

La Flash SPI

SST25VF016B

La SST25VF016B è una memoria Flash seriale appartenente alla famiglia *SPI-compatible interface*. Utilizza solo quattro linee di comunicazione per consentire l'accesso ai dati memorizzati al proprio interno ed è inserita in un package estremamente compatto da soli 8 pin. Questa tipologia di Flash presenta buone qualità in fatto di consumi energetici (l'assorbimento è minore di 5 μ A in standby) e *data retention* (supera i 100 anni ed i 100.000 cicli di scrittura); si basa sulla tecnologia CMOS SuperFlash, ideata dalla *Silicon Storage Technology* (anche essa acquisita dalla *Microchip*

Technology Inc.), che consente di memorizzare grandi quantità di byte allo stato solido in maniera non volatile. Il modello SST25VF016B impiegato nel nostro progetto consente di memorizzare fino a 16 Mbit, ovvero 2 Mbyte di informazioni. Le celle di memoria sono organizzate in blocchi da 4 Kbyte; la cancellazione di un blocco richiede meno di 18 ms e la scrittura di un byte solo 7 μ s, quindi con un clock dell'interfaccia SPI che può raggiungere gli 80 MHz, garantisce prestazioni in fatto di velocità di accesso ai dati irraggiungibili con le tradizionali memorie EEPROM.



Piedinatura del chip SST25VF016B in versione 8-SOIC e 8-WSO.

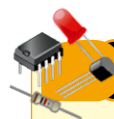
ad-hoc dei singoli task garantisce il raggiungimento di risultati impeccabili. Per scongiurare questi pericoli (per cui solo un hard reset è in grado di sbloccare la CPU ad esempio dall'attesa di un evento infinito) è bene adottare qualche meccanismo di difesa, ad esempio impiegando il *Watch Dog Timer*, così che l'applicazione venga resettata alla condizione iniziale nel caso estremo si verificano degli stalli indesiderati.

Per aumentare l'efficienza delle routine ed evitare polling su eventi "rari" o esterni, sono stati scelti gli interrupt, come ad esempio nel caso della gestione dei pulsanti **Listato 2**. Il vantaggio degli interrupt risiede nella possibilità di interrompere momentaneamente una normale routine ed assolvere ad un compito speciale o settando un flag

(possibilmente di breve durata) ritornando poi da dove si era interrotto. Impiegando adeguatamente gli interrupt sugli eventi hardware e adoperando un'opportuna suddivisione dei task in micro attività non troppo lunghe, si riesce ad ottenere un buon sistema concorrentiale, che non risente dell'assenza di un sistema operativo anche quando ci sono molti task da gestire.

Fin qui abbiamo descritto il comportamento del firmware del microcontrollore, tipicamente realizzato in C o C++; per la parte grafica delle pagine web, abbiamo scelto HTML5 in combinazione con jQuery, e i CSS per definire la formattazione delle pagine. jQuery è una libreria di funzioni JavaScript, per le applicazioni web based, che si propone come obiettivo il

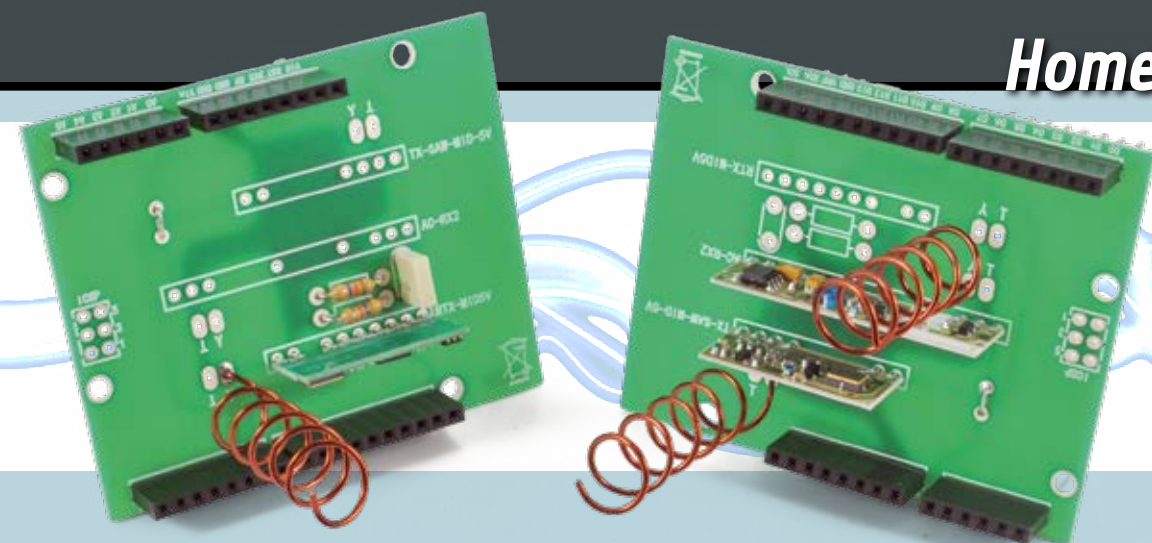
semplificare la programmazione lato client delle pagine HTML. È un software liberamente distribuibile e gratuito, pubblicato per la prima volta nel 2006. La sintassi di jQuery è studiata per semplificare la navigazione dei documenti, la selezione degli elementi DOM, creare animazioni, gestire eventi e implementare funzionalità AJAX. La selezione degli elementi del DOM ricorda molto da vicino le regole adottate per la definizione dei fogli di stile nei file CSS, consentendo di semplificare la vita ai programmatori che possono abbreviare le lunghe dichiarazioni JavaScript. L'impiego di *Asynchronous JavaScript and XML* abbreviato in AJAX consente, inoltre, all'applicazione lato client di automatizzare l'aggiornamento dei contenuti della pagina web, ad esempio nel **Listato 3** viene mostrata una chiamata temporizzata a cadenza regolare invocata dall'interprete del browser per poter contattare il server web implementato all'interno del microcontrollore e poter visualizzare lo stato aggiornato della connessione.



per il MATERIALE

La scheda Modulo per filodiffusione audio su ethernet (Cod. FT1185M) viene fornita montata e collaudata. Può essere acquistata presso Futura Elettronica al prezzo di Euro 78,00. Il Contenitore in alluminio Tekal 11.29 nero, da forare, (Cod. TEKAL1129) è disponibile separatamente al costo di Euro 9.50.

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



dell'Ing. DANIELE DENARO

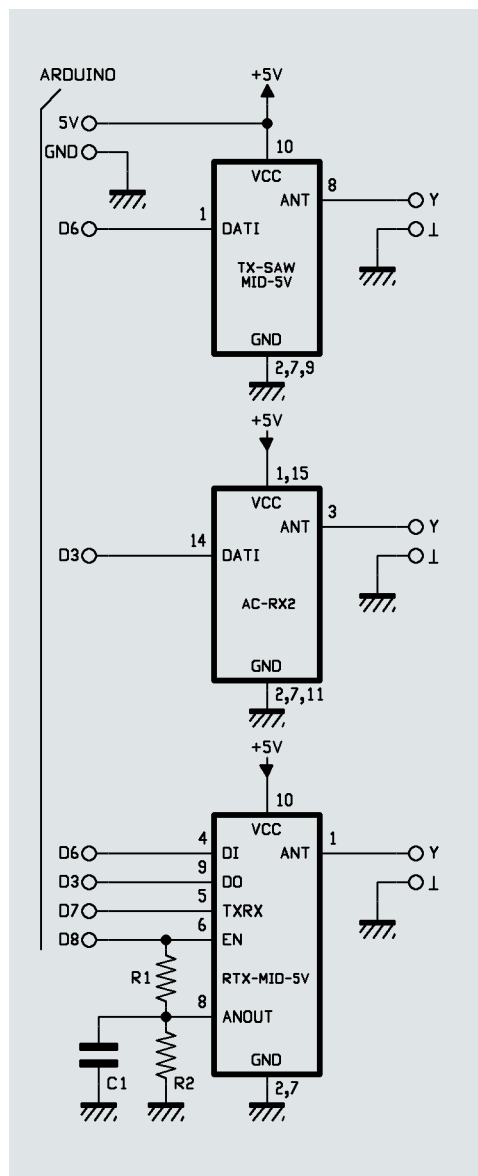
Comandiamo a distanza delle prese di rete a 220 volt in modo da gestire l'attività di utilizzatori come piantane e lampade da tavolo, ma anche stufette elettriche ed altri apparati.

RADIOCONTROLLO PRESE DI RETE



Presso la Grande Distribuzione, è facile trovare dei sistemi radiocomandati di basso costo composti da un telecomando ed alcune prese di alimentazione a 220V capaci di sopportare carichi fino a 2.000 W. Le prese sono passanti (spina+presa) e vanno collocate in una norma-

le presa dell'impianto elettrico, dalla quale traggono l'alimentazione per l'impianto di ricezione, a bassissimo consumo, che è sempre in ascolto di un eventuale comando di attivazione/disattivazione. La commutazione della tensione sulle prese è effettuata tramite un relé collegato all'uscita



di un decoder, che a sua volta riceve gli impulsi da un ricevitore, a modulazione di ampiezza, sintonizzato su 433,92 MHz. I numerosi prodotti del genere, commercializzati con vari marchi (per esempio Velleman), condividono un'unica struttura implementativa basata su una coppia di encoder-decoder corrispondente alle sigle HX2262 (PT o SC) per l'encoder (trasmettitore) e HX2272 (PT o SC) per il decoder (ricevitore). Encoder e decoder possono anche essere siglati, rispettivamente, 5262 e 5272. Questa coppia di encoder-decoder utilizza un protocollo di comunicazione che prevede una codifica con un massimo di 3^{10} combinazioni che, per l'applicazione cui i sistemi sono destinati, è più che sufficiente (basta infatti che non ci siano interferenze tra apparati analoghi all'interno di un condominio); infatti ciascun componente dispone di 12 piedini di impostazione, a tre stati (connesso a massa, connesso alla tensione di alimentazione o non connesso), 10 dei quali sono per definire l'indirizzo e gli ultimi due pin in genere usati come dati per gestire la commutazione del relè.

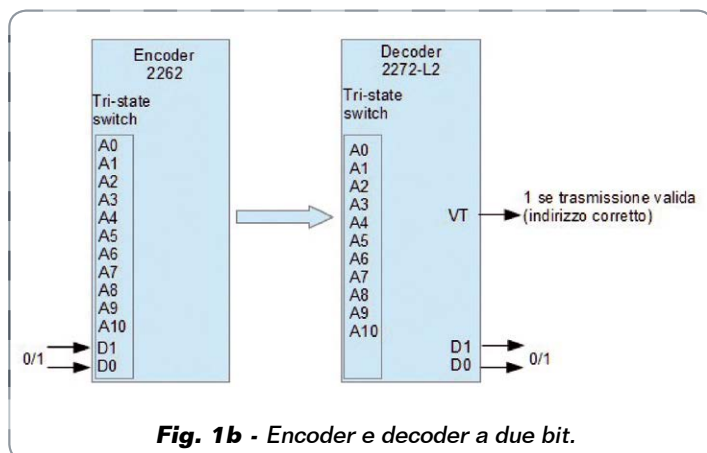
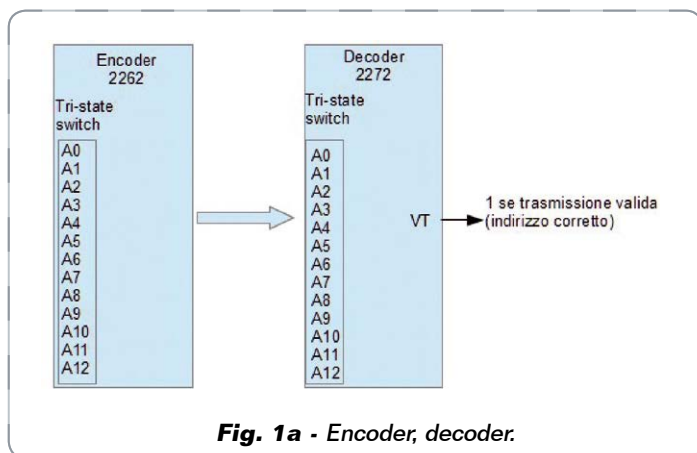
Il data-sheet dell'encoder prevede la possibilità di utilizzare

tutti i 12 pin di indirizzo, ma il decoder associato può prevedere una diversa suddivisione; infatti esistono diverse varianti dello stesso che differiscono per quanti di tali pin sono utilizzati come uscita digitale (possono essere un massimo di sei).

In sostanza l'encoder è unico, mentre il decoder prevede una serie di varianti che differiscono in base al numero di bit da trasportare. I bit di dati sono però a due stati (0/1), diversamente da quelli di indirizzo: in pratica gli ultimi sei pin dell'encoder possono essere usati come bit di dati se il formato degli stessi è binario e a patto che sull'unità ricevente sia installato un decoder di tipo compatibile, vale a dire che preveda 6 bit di indirizzamento e 6 di dati da ricevere (**Fig. 1**).

In genere i prodotti commerciali utilizzano una configurazione a due bit di dati per comandare lo stato ON/OFF della presa, che viene indirizzata dai 10 switch tri-state. Spesso l'indirizzo è composto da un indirizzo di base e da un indirizzo che individua le singole prese, per semplicità costruttiva e di utilizzo.

Sul mercato esistono, in particolare, due tipologie di prese radiocomandate, che per semplicità chiameremo tipo Avidsen e



tipo Velleman, dal nome di due marchi che le implementano; la prima è completamente configurabile tramite microswitch, mentre la seconda ha gli indirizzi cablati, e non a caso, le prese riportano un numero stampigliato. Nel primo tipo, l'indirizzo di base è composto da 5 microswitch a due stati (non collegato/collegato a massa), mentre le singole prese sono indirizzate dal singolo microswitch commutato; per questa ragione, ci sono al massimo 5 prese per ogni trasmettitore del telecomando. Chiaramente questo sottodimensionamento dello spazio indirizzabile è dovuto a criteri di semplificazione. Ancora più semplificato è l'approccio del secondo tipo, dove tutto è predisposto in fabbrica e le prese sono al massimo 3, tutte solitamente comprese nella confezione; però le singole confezioni possono avere indirizzi base diversi.

METTIAMO IN MEZZO ARDUINO

In Internet sono descritte alcune soluzioni per l'uso di Arduino come controllore di queste prese radiocomandate, tuttavia nei relativi esempi viene utilizzato un approccio semplificato ed artigianale che prevede un intervento hardware sul trasmettitore. In pratica, smontando il trasmettitore vengono simulati i pulsanti mediante il collegamento ai pin di Arduino. Il collegamento con Arduino è però mediato dall'interposizione di un buffer a transistor che cortocircuita i pulsanti, anche a causa della differente tensione di alimentazione (Fig. 2). Noi vogliamo invece utilizzare direttamente Arduino come trasmettitore codificato che simula il circuito HX2262; in questo modo potremo comandare un

considerabile numero di prese anche di marca diversa.

IL PROTOCOLLO DI COMUNICAZIONE

Per utilizzare Arduino come telecomando dobbiamo conoscere il protocollo di comunicazione, oltre a dotarci di un modulo radio a 433,92 MHz a modulazione di ampiezza come quelli disponibili nel catalogo della Futura Elettronica (www.futurashop.it). Vediamo allora di approfondire il protocollo di comunicazione usato dalla coppia HX2262 e HX2272. La codifica dei valori a tre stati si basa su una sorta di modulazione di larghezza degli impulsi; più precisamente, da una sequenza di due tipi di impulso: uno base ed uno tre volte più

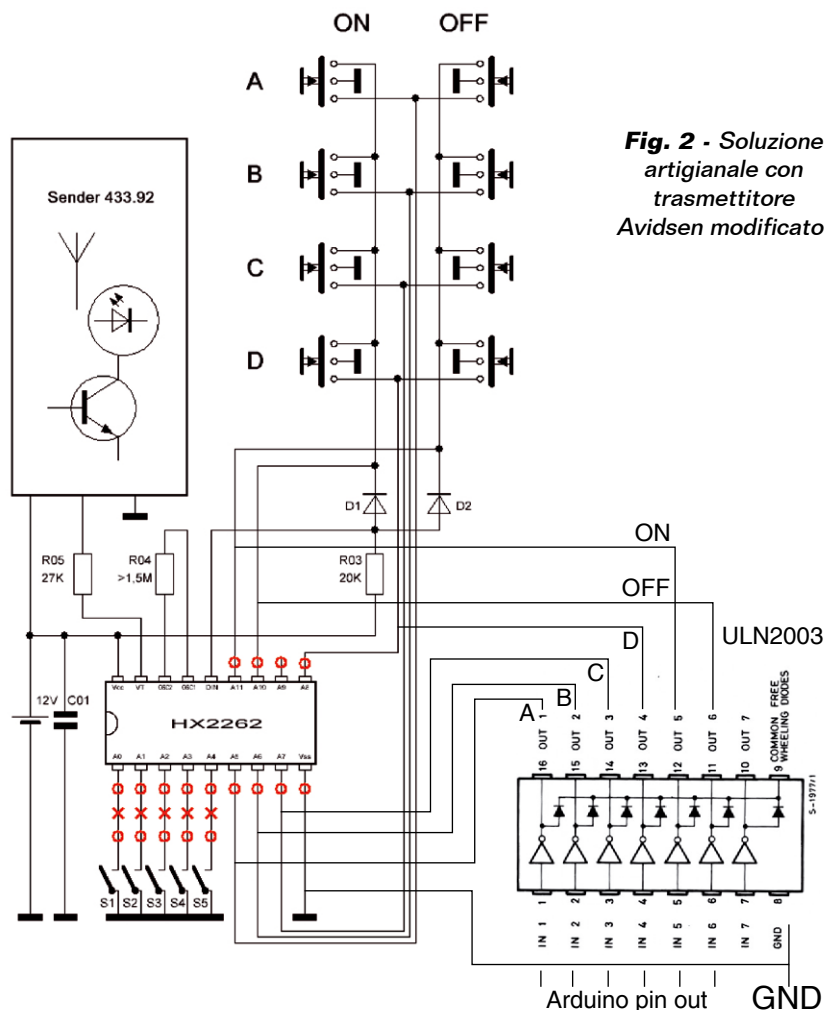


Fig. 2 - Soluzione artigianale con trasmettitore Avidsen modificato.

largo. In Fig. 3 è mostrata la rappresentazione dei tre valori: 0, 1, float. È definito anche un segnale di sincronismo che chiude la spedizione dei 12 bit di indirizzo e dati.

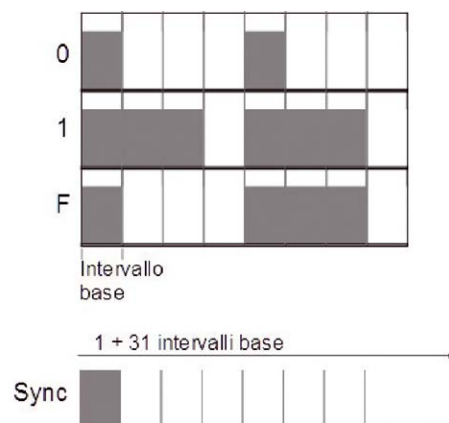


Fig. 3 - Codifica dei valori tri-state.

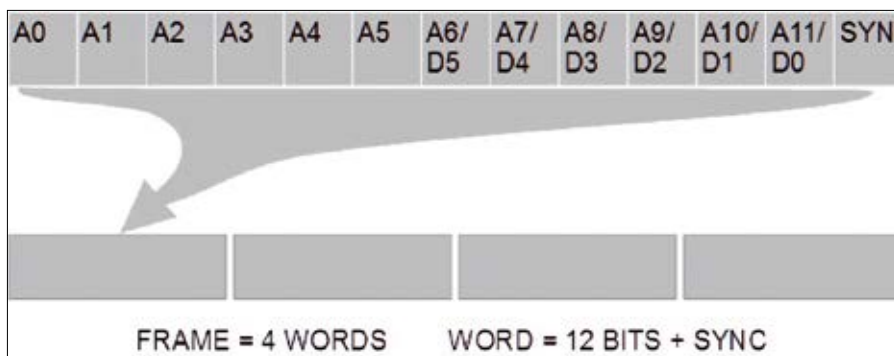


Fig. 4 - Word e frame.

Come si vede, il valore è codificato sulla base di otto intervalli temporali uguali; questo intervallo base è legato alla frequenza dell'oscillatore compreso nel chip. L'oscillatore è regolato dal valore della resistenza applicata ai due suoi piedini dedicati. Per maggiori dettagli, rimandiamo al data-sheet del componente scaricabile da Internet (basta usare, in un motore di ricerca, come chiave HX2262 e HX2272). Da alcune prove fatte nel nostro laboratorio, l'intervallo base

risulta essere circa 250 μ s per il sistema Avidsen e circa 110 μ s per il sistema Velleman. L'insieme della sequenza dei valori di indirizzo più il segnale di sincronismo costituisce una "word" che viene inviata ripetutamente per quattro volte, a formare una struttura detta "frame" (vedere la Fig. 4).

A questo punto si tratta di scrivere lo sketch per Arduino che codifichi la sequenza da inviare al modulo radio. In **Listato 1** sono descritte le funzioni che inviano

gli impulsi secondo la codifica descritta prima.

Come vedete, avendo noi scelto di adottare un approccio semplificato, non sono state utilizzate funzioni C con accesso diretto ai timer e alle porte, ma semplicemente funzioni standard dell'IDE Arduino, che comunque risolvono la problematica in maniera adeguata.

Tutta la word viene costruita mettendo insieme l'indirizzo base, l'indirizzo corrispondente alla singola presa e due bit di dati. L'indirizzamento della presa può essere implementato in modo diverso tra i diversi prodotti (Fig. 5). Per esempio:

- per il sistema Avidsen le prese sono identificate da un unico bit a 0 logico tra cinque (gli altri sono floating);
- per il sistema Velleman le prese sono identificate con una codifica su tre bit.

Costruita la "word" secondo le diverse modalità, questa verrà inviata al pin di trasmissione dell'encoder scandendola bit per bit, utilizzando le funzioni descritte precedentemente e, infine, aggiungendo il bit sync. Inoltre la "word" verrà ripetuta quattro volte, come richiesto dal protocollo.

Il modulo radio dovrà essere capace di discriminare impulsi della durata minima di circa $100 \div 150 \mu$ s, ovvero dovrà presentare una "banda passante" corrispondente a circa 9 kbps. Questa semplice implementazione permette di gestire carichi a 220V in tutta sicurezza, essendo le prese conformi alla normativa vigente e comandate via radio. In realtà, il sistema non è utile solo per comandare a distanza, ma anche -semplicemente- per gestire carichi significativi senza l'impiccio dovuto alla presenza

Listato 1

```
#define TBASE 250 //intervallo base in uSec = osc*4
#define TBASE3 TBASE*3
#define TBASE31 (TBASE*31)/1000
#define TBASE31M (TBASE*31)-TBASE31

void sendBit(byte b)
{
  switch (b)
  {
    case 0 :
      digitalWrite(PDOUT,1);delayMicroseconds(TBASE);
      digitalWrite(PDOUT,0);delayMicroseconds(TBASE3);
      digitalWrite(PDOUT,1);delayMicroseconds(TBASE);
      digitalWrite(PDOUT,0);delayMicroseconds(TBASE3);
      break;
    case 1 :
      digitalWrite(PDOUT,1);delayMicroseconds(TBASE3);
      digitalWrite(PDOUT,0);delayMicroseconds(TBASE);
      digitalWrite(PDOUT,1);delayMicroseconds(TBASE3);
      digitalWrite(PDOUT,0);delayMicroseconds(TBASE);
      break;
    case 2 :
      digitalWrite(PDOUT,1);delayMicroseconds(TBASE);
      digitalWrite(PDOUT,0);delayMicroseconds(TBASE3);
      digitalWrite(PDOUT,1);delayMicroseconds(TBASE3);
      digitalWrite(PDOUT,0);delayMicroseconds(TBASE);
      break;
  }
}

void sendSync()
{
  digitalWrite(PDOUT,1);delayMicroseconds(TBASE);
  digitalWrite(PDOUT,0);delay(TBASE31);delayMicroseconds(TBASE31M);
}
```

di cavi elettrici.

Con i sistemi tipo Avidsen è possibile acquistare diverse prese (anche più di 5) e comandarle agendo sulla parte dell'indirizzo di base; infatti in ogni presa è definibile, tramite microswitch, tutto l'indirizzo e non solo la parte relativa all'identificazione della presa stessa.

Se ad Arduino colleghiamo anche uno shield con RTC, possiamo realizzare una gestione temporizzata di diversi carichi.

LIBRERIA SOFTWARE

Anche se, come abbiamo visto nei paragrafi precedenti, implementare uno sketch Arduino per spedire la corretta sequenza di impulsi è abbastanza semplice, abbiamo voluto semplificare la vita a chi si accinge a mettere in pratica questo tipo di radio-comandi ed abbiamo perciò preparato una libreria software corredata da un paio di sketch di esempio.

L'archivio che contiene il tutto è in formato compresso "zip" e lo potete scaricare dal nostro sito www.elettronica.in insieme agli altri file del progetto.

Una volta scompattato l'archivio, troverete una cartella "RTXlib", dentro la quale sono presenti i due sketch *TestRTXlib* e *ScannerRX*; nella cartella troverete anche la libreria vera e propria. Poiché l'IDE Arduino ha bisogno di avere i file di libreria tutti nella stessa cartella, a meno di non installare la libreria nella directory "libraries" dell'IDE, i riferimenti ad essa si troveranno inseriti, nei due sketch a corredo, con il path assoluto, che quindi andrà modificato inserendo quello corrispondente alla posizione attuale.

Nel caso vogliate utilizzare la libreria con un vostro sketch, potete inserire il path completo,



Fig. 5 - Tipi di indirizzamento.

oppure copiare la libreria nella cartella dello sketch, oppure installare definitivamente la libreria in "libraries".

Le parti da modificare sono:

```
#include "D:\Arduino\ArduinoWork\ _  
RTXlib\RTXlib\RTX.h"  
#include "D:\Arduino\ArduinoWork\ _  
RTXlib\RTXlib\RTX.cpp"
```

Le funzioni base della libreria sono le seguenti:

- inizializzazione del protocollo HX2262-HX2272;
- trasmissione di una stringa di 12 valori three-state (0, 1, 2).

Va comunque detto che la libreria ha altre funzionalità, che sono le seguenti:

- inizializzazione ricezione protocollo HX2262-HX2272;
- ricezione di una stringa di 12 valori three-state.

Per questa ragione, oltre che trasmettere permette anche di ricevere.

Inoltre la libreria ha funzioni predisposte per trasmettere e ricevere con codifiche diverse come quelle a due stati e numerose funzioni di basso livello.

Gli sketch inclusi nel nostro archivio sono due: *TestRTXlib* e *ScannerRX*. Lo sketch *TestRTXlib* implementa la trasmissione e ricezione continua dei dati.

Il comando viene impartito tramite la console seriale e su di essa è evidenziato l'eventuale comando arrivato da un trasmettitore del sistema (Fig. 7).

Il "setting" delle caratteristiche di funzionamento è predisposto da una serie di "define" che stabiliscono la tipologia del sistema (sono definiti i sistemi Avidsen e Velleman), la lunghezza dell'impulso base espressa in microsecondi e, infine, l'indirizzo di riferimento.

La funzione "loop" dello sketch non fa altro che scandire l'input

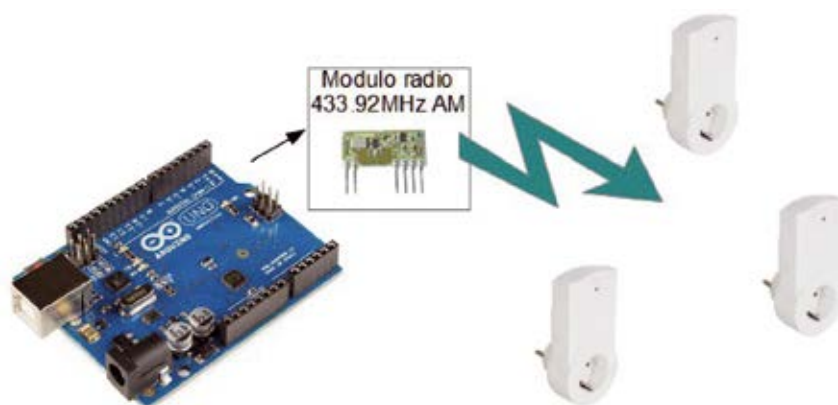


Fig. 6 - Arduino controlla via radio carichi alimentati a 220V.

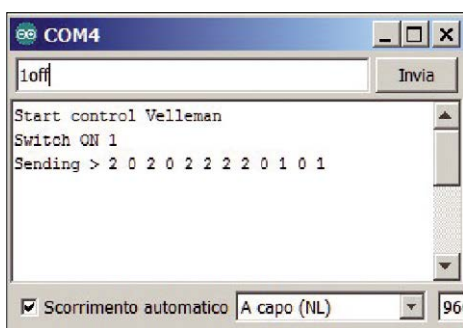


Fig. 7 - Trasmissione dei comandi.



Fig. 8 - Rilevazione dell'indirizzo base nei sistemi Avidsen.

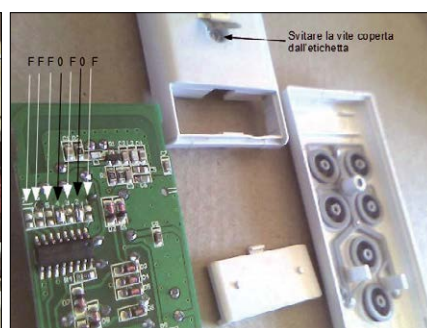


Fig. 9 - Rilevazione dell'indirizzo base nei sistemi Velleman.

della console seriale per vedere se è stato inserito un comando da decodificare e spedire, e per accertarsi che non stia arrivando un comando inviato da un trasmettitore terzo:

```
void loop()
{
  if (Serial.available() > 0) decode();
  checkReceive();
}
```

La funzione di ricezione è stata inserita per completezza. Se siete interessati alla sola trasmissione dei comandi, potete tranquillamente commentare la linea corrispondente nella funzione loop. Inoltre la funzione di

ricezione è inutile se disponiamo solo di un modulo radio trasmettente. Ma dei moduli radio parleremo in un secondo momento, vale a dire a conclusione di questo articolo.

Nel caso siate interessati anche alla ricezione di comandi, dovrete implementare voi una funzione che decodifica la stringa arrivata in base ad un indirizzo da voi predisposto per il confronto, e che lancia la conseguente azione che avrete predisposto. Lo sketch *ScannerRX*, invece, prevede obbligatoriamente l'utilizzo di un modulo radio ricevente; questo perché è stato pensato per ricevere il treno di impulsi e

fornire indicazioni a riguardo. Lo sketch si pone in ascolto ed alla ricezione di 12 valori visualizza la durata dell'impulso base e la stringa ricevuta. In questo modo è possibile determinare le caratteristiche del sistema trasmettitore-repre anche quando non sono presenti gli switch, come nel caso di sistemi Velleman. (Fig. 9) Oltre alla lunghezza (in microsecondi) dell'impulso base (calcolata come media), vengono visualizzati anche i singoli impulsi con segno positivo per indicare un impulso ON e con il segno negativo per indicare l'impulso OFF (o pausa). Successivamente, per comodità,

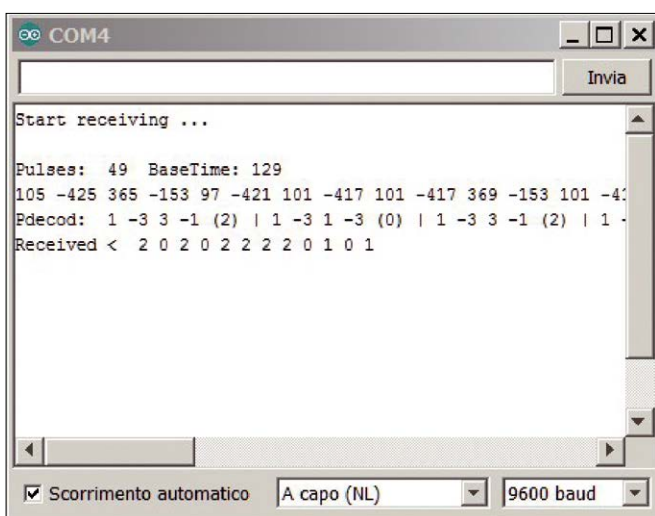


Fig. 10 - Ricezione degli impulsi del sistema Velleman.

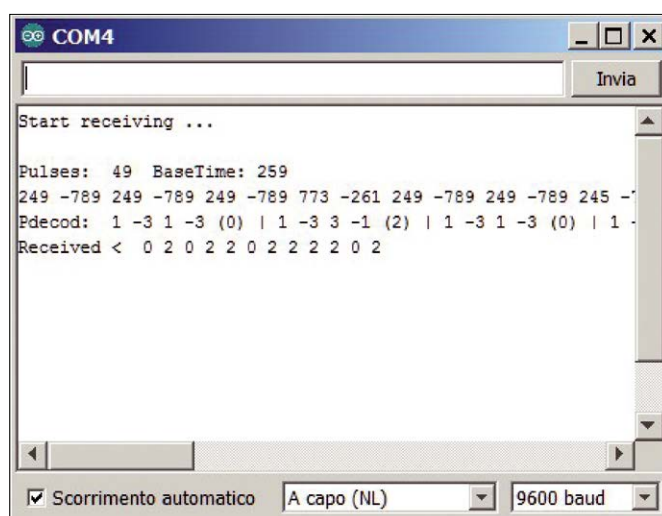


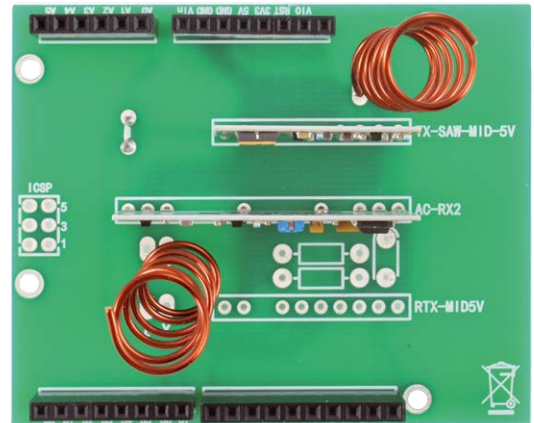
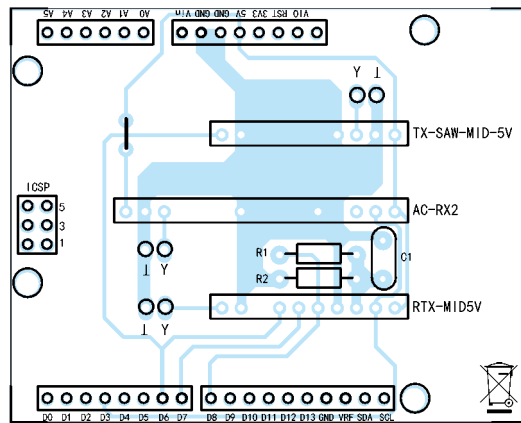
Fig. 11 - Ricezione degli impulsi del sistema Avidsen.

Elenco Componenti:

R1, R2: 47 kohm
C1: 1 nF ceramico

Varie:

- Strip M/F 6 vie
- Strip M/F 8 vie (3 pz.)
- Strip Maschio /Femmina 10 vie
- Spezzone filo 17cm
- Modulo RF (RTX-MID5V /AC-RX2/TX-SAW MID-5V)
- Circuito stampato S1212



vengono rappresentati gli stessi impulsi codificati in termini di rapporto con l'impulso base (1 o 3). Poiché ogni dato è codificato con un gruppo di quattro impulsi (2 ON e 2 pause), ad ogni "quartetto" viene associato il valore "three-state" decodificato. I dodici valori sono infine visualizzati tutti insieme. Come si vede, il "timing" è diverso nei due sistemi: più corto per il Velleman e più lungo nell'Avidsen (Fig. 10 e 11).

Lo sketch *ScannerRX* costituisce anche il modo più semplice per verificare il "timing" cioè la lunghezza dell'impulso base, oltre a verificare gli indirizzi.

Se non si può utilizzare lo sketch *ScannerRX* perché si è sprovvisti di modulo radio ricevente, bisogna rilevare l'indirizzo base utilizzando i microswitch -se presenti, come nel caso del sistema Avidsen- oppure è necessario aprire il telecomando (è il caso del sistema Velleman). Il tutto va fatto tenendo presente che l'impulso base non dovrebbe cambiare nell'ambito di un prodotto ed è di circa 110 microsecondi per il sistema Velleman e di circa 250 microsecondi per l'Avidsen.

L'IMPLEMENTAZIONE DELL'HARDWARE

Per realizzare il progetto bisogna, per prima cosa, dotarsi di un modulo radiotrasmettente; questo deve essere, come si è detto, un modulo TX sintonizzato sui 433,92 MHz e operante in modulazione di ampiezza operata da segnali digitali: è una AM detta anche OOK perché la modulazione non è lineare ma di tipo on (portante presente)/off (portante assente). In alternativa è possibile procurarsi un modulo ricetrasmettitore, sempre operante in AM/OOK a 433,92 MHz.

Noi abbiamo provato tutte e due le configurazioni, impiegando moduli RF Aurel presi dal nutrito catalogo della Futura Elettronica (www.futurashop.it): il primo è un trasmettitore a bassissimo prezzo (codice 8110-TX-SAW MID-5V: costa circa 6 euro) cui abbiamo affiancato un ricevitore (facoltativo) anch'esso a bassissimo prezzo (codice 8110-AC-RX2: costa intorno ai 5 euro) (Fig. 12). In alternativa abbiamo lavorato con un ricetrasmettitore siglato 8110-RTX-MID5V, ovvero un modulo contenente sia un'unità trasmittente, sia una ricevente (Fig. 13) sintonizzate entrambe sulla stessa frequenza e aventi in comune il piedino d'antenna. Tutti questi moduli vanno alimentati a 5V, quindi si prestano ad essere abbinati ad Arduino e a ricevere l'alimentazione dal pin 5V dei connettori di quest'ultimo. Naturalmente potete utilizzare qualunque altro modulo AM a 433,92 MHz purché supporti la modulazione d'ampiezza OOK. Il modulo trasmettitore può essere sempre alimentato in modo continuo, perché l'irradiazione della portante avviene solo in corrispondenza degli impulsi ON, e quindi in mancanza di trasmissione non consuma praticamente energia. Invece per il ricevitore si può pensare a

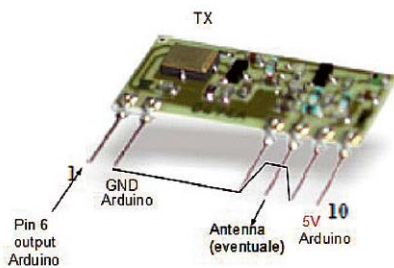
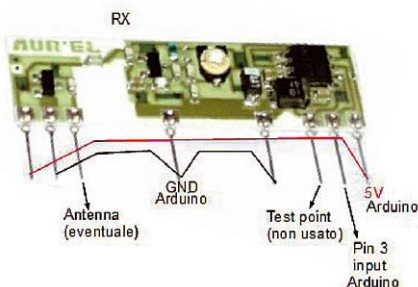
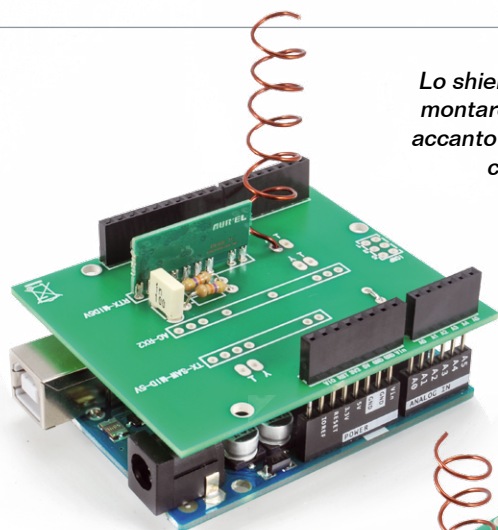


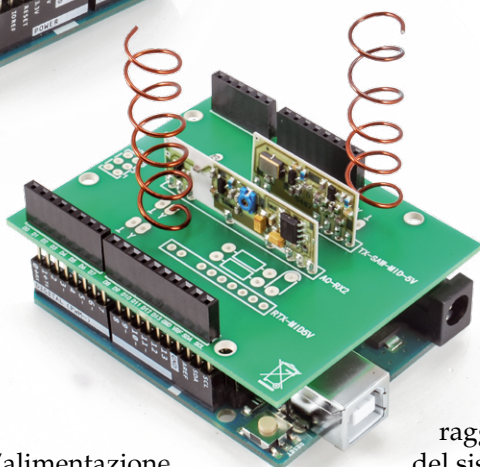
Fig. 12 - Trasmettitore TX-SAW MID-5V e ricevitore AC-RX2.



Lo shield proposto consente di montare tre diversi moduli: qui accanto lo vedete equipaggiato con il solo RTX-MID5V e relativa antenna.



A lato, lo shield con montati il ricevitore AC-RX2 e il trasmettitore ibrido TX-SAW MID-5V: in questo caso ognuno ha la sua antenna.



somministrare l'alimentazione tramite un transistor pilotato da un pin di Arduino. Optando per il ricetrasmittente, esso deve commutare dalla modalità trasmissione alla modalità ricezione seguendo una determinata sequenza temporale di impulsi su PWR (Enable) e TX/RX. Questa sequenza è descritta nel data-sheet del modulo ed è implementata nella libreria; infatti un apposito "define" nello sketch permette di commutare tra i precedenti moduli e quest'ultimo ibrido.

Il ricetrasmittente RTX-MID5V è preferibile se si vuole avere anche la possibilità di ricevere sequenze da decodificare oltre a trasmettere comandi di accensione/spegnimento. Riguardo alla copertura del segnale radio, con ambedue i trasmettitori si possono tranquillamente raggiungere diversi metri e stanze non troppo distanti anche in presenza di muri. Con il ricevitore Aurel AC-RX2 il sistema presenta una copertura leggermente inferiore. Nel caso vogliate estendere il

raggio di azione del sistema di prese radiocomandate, dovete dotarvi di trasmettitori e ricevitori più performanti. La parte RF del progetto può essere realizzata su una basetta millefori, collegata ai pin di Arduino tramite fili, ma nessuno vi vieta di preparare un c.s. adatto, con la forma e dei contatti disposti per combaciare con la piedinatura di Arduino: in altre parole, uno shield dedicato come quello di cui in queste pagine proponiamo lo schema elettrico e il piano di montaggio.

Il nostro shield può ospitare tutti e tre i moduli proposti, anche se non vanno montati tutti e tre insieme; per l'esattezza, le opzioni disponibili prevedono il montaggio dell'RX e del TX separati (in questo caso ci vuole un'antenna distinta per ciascun modulo) o, in alternativa, dell'RTX (in questo caso basta un'unica antenna). Nello shield qui proposto, la linea D3 di Arduino riceve i dati in arrivo dall'uscita del modulo RX (o dall'uscita DO dell'RTX: le due sono in parallelo) e la D6

Circuito per la riduzione del rumore in ricezione

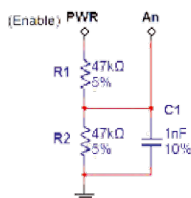
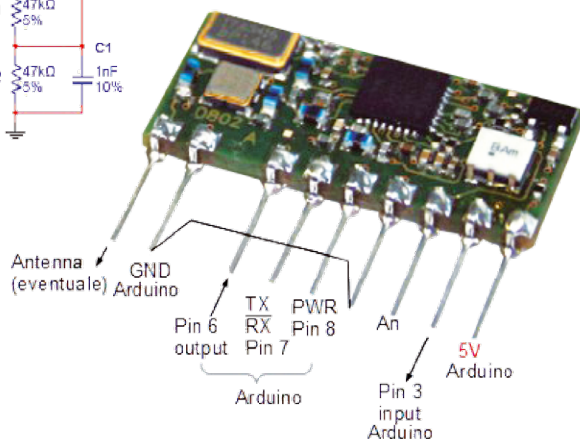
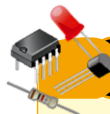


Fig. 13
Ricetrasmittente RTX-MID5V.



porta i dati da trasmettere da Arduino al modulo TX (ovvero all'ingresso DI dell'RTX). Il partitore R1/R2 serve alla polarizzazione della linea ANOUT del modulo ricetrasmittente RTX-MID5V.



per il MATERIALE

Il materiale utilizzato in questo progetto può essere acquistato presso Futura Elettronica. Il master del circuito stampato è scaricabile dal sito della rivista. La board Arduino Uno REV3 (cod. ARDUINOUNO-REV3) costa 24,50 Euro. Il modulo trasmettitore AM - 5V (cod. TX-SAW MID-5V) è disponibile al prezzo di 6,00 Euro. Il modulo ricevitore AM - 5V (cod. AC-RX2) è in vendita a Euro 5,00. Il mini ricetrasmittente dati (cod. RTX-MID5V) costa Euro 16,00. La confezione di 3 prese radiocomandate + radiocomando (cod. WRS4) è disponibile a Euro 29,00.

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

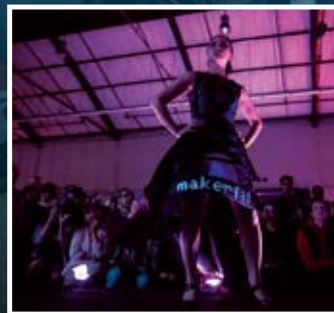
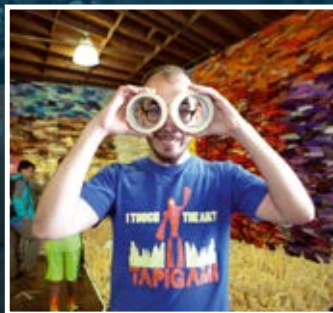
I DIECI ANNI DELLA

Maker Faire®

di ARSENIO SPADONI

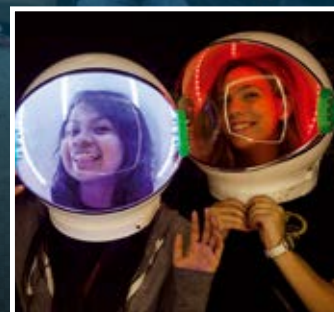
Anche quest'anno siamo tornati dove tutto è nato, in quella enorme area al centro di San Mateo, un po' parcheggio un po' fiera di paese, dove nel 2006 si svolse la prima Maker Faire, e che quest'anno ha ospitato la decima edizione. L'anno prima, nel 2005, nacque la rivista Make, manifesto e simbolo di quel Maker Movement che in poco tempo

Ancora una volta la prima e più importante Maker Faire, quella di San Mateo, in California, non ha deluso le aspettative degli oltre 145 mila visitatori (nuovo record) che hanno affollato i padiglioni e gli spazi all'aperto del County Event Center.



avrebbe dato nuova dignità al mondo degli hacker, degli smanettoni e degli sperimentatori, a quegli artigiani tecnologici che oggi vengono identificati con la parola makers e

che, grazie soprattutto a Internet, hanno imparato a condividere conoscenze e competenze. E che ora, come afferma Dale Dougherty, ideatore della Maker Faire, sono pronti ad hackerare il mondo nello stesso modo con cui mettono mani ai computer ed alle schede a microcontrollore. Ovvero con un atteggiamento più creativo e più innovativo, cercando di dominare la tecnologia anziché esserne dominati; in altre parole cercando di utilizzarla come elemento abilitante della propria creatività. Un fenomeno culturale che si è diffuso rapidamente in tutto il mondo e che è stato favorito dalla disponibilità di tecnologie sempre più





semplici, più a buon mercato e, soprattutto, completamente "aperte" sia nell'hardware che nel software. Tecnologie attorno alle quali si sono formate delle comunità di appassionati (pensiamo a DIY Drones di Chris Anderson, ad Arduino e a Raspberry Pi) che hanno portato in pochi anni alla nascita di nuovi prodotti e nuovi mercati, dai droni alle stampanti 3D. Un movimento culturale che è anche alla base di numerosi altri fenomeni, dai FabLab - luoghi dove i makers condividono attrezzature ed esperienze - ai coworking, spazi e servizi comuni dove iniziare attività imprenditoriali, fino agli hackerspace, luoghi dove i makers si ritrovano per condividere attività e progetti di interesse comune.

I confini tra queste iniziative sono piuttosto labili e spesso all'interno di spazi di coworking nascono anche dei FabLab che magari sono semplici hackerspace. Sicuramente il Maker Movement ha anche molto in comune con fenomeni quali il crowdfunding e con la nascita di nuove imprese di tipo innovativo ovvero col fenomeno delle start-up. Spesso, infatti, molte delle attività e dei progetti realizzati per passione dai makers si trasformano in vere e proprie

attività di tipo imprenditoriale. In fondo Steve Jobs, Dave Packard e Bill Hewlett altri non erano che dei makers. Questi nomi ci ricordano altri nomi, quelli delle località che troviamo nei dintorni di San Mateo: Palo Alto, Cupertino, Menlo Park, Mountain View, Sunnyvale, San José, Redwood Shores, Santa Clara e tanti altre ancora, dove hanno sede le più innovative società al mondo, da Google a Intel, da Apple a Facebook. La Maker Faire ed il Maker Movement non potevano che nascere qui, in quell'area compresa tra San Francisco e Santa Clara che si percorre in meno di un'ora di macchina lungo la 101 e che attraversa la Bay Area e la Silicon Valley.



Un ecosistema dove continuano a nascere e svilupparsi imprese particolarmente innovative, con una visione molto lucida e puntuale del futuro. Guardando all'attività di queste aziende possiamo intravedere come sarà il nostro mondo tra 10 o 20 anni. Società come Uber e Airbnb (entrambe di San Francisco) stanno cambiando il modo con cui ci avviciniamo alla mobilità e alle città; Tesla (che ha sede a Palo Alto) sta guidando l'innovazione in campo automobilistico e non solo (i sistemi di accumulo domestici lanciati recentemente potrebbero cambiare radicalmente l'assetto delle reti energetiche); SpaceX (con sede a Hawthorne) sta rivoluzionando la tecnologia spaziale.

Per non parlare delle iniziative di Google in campo genetico: i progetti Sens e Calico intendono sconfiggere la vecchiaia e, addirittura, regalarci l'immortalità. Sembra fantascienza ma il ragionamento dei ricercatori è molto semplice: basta bloccare la degenerazione delle cellule e il gioco è fatto. Queste iniziative danno il senso dell'aria che si respira da queste parti dove la cultura del "non è possibile" è bandita e dove tutto è ispirato ad un misto di follia e fiducia nel futuro; dove la tecnologia non è fine a se stessa ma un mezzo per migliorare la nostra vita e il mondo.

La stessa aria si respira anche alla Maker Faire, dove non si va – come in una tradizionale fiera - a vedere questa o quella applicazione, per poi magari acquistarla e portarla a casa. Alla Maker Faire si va a vedere quello che i makers o le community sono riusciti a realizzare con la loro creatività sfruttando le tecnologie disponibili, alla Maker Faire si va – come dicono da queste parti – per farsi "ispirare".

Camminando tra i padiglioni della fiera e nelle vaste aree all'aperto abbiamo visto come la creatività dei makers abbia declinato nelle forme più incredibili tutte le tecnologie disponibili: abbiamo incontrato mostri meccanici realizzati con ruspe in disuso o improbabili velocipedi, bellissimi giochi di luce o lampade dalle forme più strane. Alcune di queste realizzazioni sono diventate nel tempo delle vere e proprie attrazioni, come le fontane a base di CocaCola & Mentos o i mostri sputafuoco della Fire Art.

Alcuni di questi oggetti si possono trovare solamente alla Maker Faire di San Mateo, proprio per lo spirito unico che caratterizza quest'area della California. Girando tra gli stand e incrociando nei viali le attrazioni, abbiamo avuto la netta sensazione del futuro che ci aspetta, indefinibile ma precisa allo stesso tempo. La certezza che quelle persone con la loro creatività, il modo giocoso di porsi, i marchingegni spesso completamente inutili ci stessero inviando un



Makers on the beach: a Rimini la prima Mini Maker Faire da spiaggia!

Sabato 14 e domenica 15 novembre 2015 appuntamento con la prima edizione di Rimini Beach Mini Maker Faire, un evento dedicato ai "makers", un termine universale che unisce persone di diversa formazione ma che condivide l'interesse per l'apprendimento e lo sviluppo creativo. Grazie ad una nuova dimensione sociale, creata dai Fab Lab, luoghi in cui si progetta, si sperimenta, si costruisce, i makers creano continuamente momenti di aggregazione e collaborazione, aprendo nuove frontiere al fai-da-te. Rimini Beach Mini Maker Faire è affiliata al circuito "Maker Faire", un brand nato in America, dedicato al mondo della creatività tecnologica di proprietà di Maker Media, editore della rivista MAKE, portavoce e punto

di riferimento del movimento Makers in tutto il mondo. Un'occasione imperdibile per conoscere questa realtà così dinamica fatta di ricerca, innovazione e condivisione; i visitatori potranno testare il proprio talento di artigiano del terzo millennio grazie ai tantissimi stimoli proposti nella due giorni riminese: workshop, presentazioni, laboratori didattici, "pillole" e naturalmente materiali, strumenti, componenti, elettronica per la realizzazione di progetti DIY (do-it-yourself). Cuore dell'evento sarà l'officina creativa del Temporary Fablab, circondato dai progetti partecipanti alla CALL Makers on the Beach, una "chiamata" rivolta a singolo individui, scuole, associazioni, start-up e

imprese, a proporre progetti innovativi dedicati al mare, alle coste e alla spiaggia.

La Blue Economy – economia del mare – italiana, con 7859 chilometri di coste, è un settore importante e vitale sia sul piano occupazionale che imprenditoriale. Un dinamismo che si traduce in un apporto del mare al valore aggiunto prodotto dal Paese pari a 41,5 miliardi di euro che, grazie all'effetto moltiplicatore del comparto, diventano 120 se si considera anche l'indotto. (Dati 3° Rapporto Unioncamere – Aprile 2014).

I progetti partecipanti potranno riguardare turismo, servizi, sport, benessere, divertimento, trasporti, pesca, architettura, nautica, sicurezza marittima, costiera, natatoria e balneare.



preciso messaggio, al quale non è estranea la curiosità del pubblico che ha partecipato alla fiera (tantissimi i bambini!). Creatività e curiosità che fanno passare in secondo piano l'aspetto tecnologico, ma che proprio per questo motivo sono in grado di metterne in luce le caratteristiche più significative.

Anche tutti i produttori di semiconduttori che da al-

cuni anni partecipano alla Maker Faire (Texas Instruments, Microchip, Atmel, ecc.), spingono molto sull'aspetto creativo delle applicazioni presentate, piuttosto che sull'aspetto tecnologico. Negli ultimi tempi tutti i produttori di semiconduttori si sono avvicinati al mondo dei maker proponendo prodotti e demo specifiche, semplificando dove possibile i sistemi di sviluppo e supportando tutte quelle aziende e associazioni che progettano e realizzano prodotti embedded e schede di sviluppo per questo mercato (Arduino, RaspberryPi, Beagle Bone, Banana Pi, Udoo, ecc.).

Ovviamente tutte queste realtà più tante altre erano presenti alla Maker Faire insieme a



L'obiettivo dichiarato è quello di riunire in un unico spazio tutte le idee, visioni e opportunità offerte dall'applicazione delle tecnologie, digitali ma non solo.

Alla Rimini Beach Mini Maker Faire troveranno spazio anche idee e prodotti per ambiti diversi dal tema principale della call. Rispondi subito alla CALL! Il termine per l'iscrizione è stato fissato al 30 giugno 2015.

Nell'ambito dell'evento sono previsti settori commerciali e focus specifici sui trend del momento: stampanti 3D, droni o, più correttamente, UAV - Unmanned aerial vehicle - videogame, fotografia, robotica, videomaker, Internet of Things, musica.

Completano la rassegna Expo Elettronica, la mostra mercato di elettronica professionale e di consumo più famosa d'Italia e CAB - Recycle & Co, un'esposizione di artigiani e artisti contemporanei che

recuperano e trasformano prodotti e materiali destinati alla distruzione, facendoli rivivere in piccole opere d'arte uniche e in oggetti utili, una best practise per limitare i rifiuti, ridurre i consumi energetici ma soprattutto per la salvaguardia del nostro pianeta.

Per saperne di più:

www.makerfairerimini.it



RIVIERA DI RIMINI

Make Magazine, col suo stand centrale dove si sono svolte i seminari, le presentazioni e i workshop, tutti sempre particolarmente affollati.

Dopo l'esperienza della Maker Faire di New York, per la prima volta anche a San Mateo è stata organizzata - a cura dell'ICE (Istituto per il Commercio Estero) - un'area destinata ai maker, ai FabLab, alle start-up e alle aziende italiane che operano in questo campo. E per la prima volta anche noi di ElettronicaIn/Futura Elettronica abbiamo partecipato come espositori (dopo averla frequentata per tanti anni come visitatori) a questa manifestazione, presentando l'ultima versione della Choco 3Drag e l'applicazione per

PhotoSharing realizzata con la board RandA e descritta sulle pagine di questo numero della rivista.

Non sappiamo quanto siamo stati propositivi o innovativi, non sta a noi dirlo; quello che possiamo affermare è che abbiamo sicuramente suscitato stupore, curiosità e interesse nelle tantissime persone che si sono fermate al nostro stand.

Le immagini che pubblichiamo, riprese dal nostro sistema di PhotoSharing, sono solamente

una piccola parte delle centinaia di ritratti che si sono fatti i nostri visitatori e che sono stati pubblicati sulle loro e sulle nostre pagine Facebook. Per quanto riguarda l'area espositiva organizzata dall'ICE, questa volta lo spazio a disposizione è sembrato addirittura esagerato (rispetto alla Maker Faire di New York dello scorso settembre), anche perché le aziende che hanno aderito sono state davvero pochine, con "Fonderie Digitali" che ha occupato la





metà dello spazio disponibile. Per quanto riguarda gli altri italiani presenti, da segnalare le due organizzazioni che si contendono il marchio Arduino: da una parte lo stand di Arduino.cc in collaborazione con Atmel e con la presenza di Massimo Banzi e, a poche decine di metri di distanza, quello di Arduino.org di Gianluca Martino e Federico Musto con la presenza di Dog Hunter e Linino.

Al primo stand erano presenti anche i ragazzi di Sharebot, quelli di Primo e una demo Sigfox funzionante con la board SmartEverything di Amel; nello stand di Arduino.org erano invece presenti i ragazzi dell'Università di Messina che stanno lavorando all'IDE e al software dei nuovi prodotti della società di Musto. Sabato, come riportato nell'articolo che segue, Massimo Banzi ha annunciato in un affollato speech, la nuova strategia di Arduino.cc che prevede un accordo produttivo con Adafruit relativo al mercato statunitense e il lancio del nuovo marchio Genuino per quanto riguarda il resto del mondo.

La decima edizione della Maker Faire Bay Area ha visto la presenza di 145.000 visitatori (12% in più rispetto all'edizione 2014), grazie anche all'apertura pomeridiana del venerdì e ad un'area che è stata aggiunta "rubandola" al parcheggio principale. In considerazione dell'estensione attuale della fiera, a nostro avviso, questo è il numero massimo di visitatori: in alcuni momenti del sabato e della domenica era infatti praticamente impossibile muoversi e le code ai punti di ristoro sono lunghissime.

Il sabato sera, come tradizione, l'organizzazione ha offerto Paella e birra a tutti makers e gli espositori presenti, con Dale Dougherty in persona a riempire i piatti.

In sottofondo, la musica del complesso Bike the Rock, con le biciclette che alimentano l'impianto elettrico del gruppo e tantissimi ragazzi a pedalare per garantirsi la musica. Sicuramente un altro momento significativo della manifestazione, anche questo in sintonia con lo spirito della Maker Faire.

In occasione della decima edi-





zione della Maker Faire, le città di San Mateo, San Francisco, e Oakland hanno dichiarato la settimana precedente "Maker

Week" organizzando attività, incontri, ed eventi collaterali. Questi gli altri numeri della manifestazione:

- I visitatori sono arrivati da 46 Stati degli USA e da 45 paesi di tutto il mondo
- Oltre il 40% dei visitatori ha preso parte alla Maker Faire per la prima volta
- Il 59% dei visitatori era composto da famiglie
- 1.200 sono stati i progetti dei maker presentati quest'anno
- Oltre 500 I maker che hanno organizzato attività col pubblico
- 278 sono state le presentazioni di progetti, esperienze e attività presso l'apposita area di Make Magazine
- Il 24% dei maker presenti ha partecipato per la prima



volta alla Maker Faire

- I maker provenienti dall'estero sono stati 59

I numeri delle Maker Faire nel resto del mondo sono ancora più impressionanti: nel 2014 si sono svolte 131 manifestazioni sotto l'egida di Make Media con la presenza di 2,3 milioni di visitatori.

L'appuntamento per la undicesima edizione della Maker Faire Bay Area è per il weekend del 21-22 maggio 2016. ■



Due siti, quotazione in Borsa, cause legali in mezzo mondo: cosa sta succedendo in casa Arduino? Ce lo spiega in questa intervista Massimo Banzi, poche ore dopo il lancio del nuovo logo e del nuovo brand "Genuino".

DA ARDUINO A GENUINO, LE RAGIONI DI UNA SCELTA

di ARSENIO SPADONI

Quanti si occupano solamente di tecnologia e negli anni passati hanno utilizzato le schede Arduino unicamente per dare sfogo alla propria creatività, saranno rimasti sicuramente sorpresi nell'apprendere che proprio Massimo Banzi – da tutti riconosciuto come leader indiscusso del team Arduino – ha presentato alla Maker Faire di San Mateo che si è svolta il weekend del 16/17 maggio scorso, un nuovo brand e un nuovo logo: Genuino. Un repentino cambio di strategia nella spinosa questione dell'utilizzo del marchio Arduino che

vede Banzi (e gli altri fondatori del team) contrapposto all'unico esponente italiano del gruppo, Gianluca Martino, col quale ha definitivamente interrotto qualsiasi forma di collaborazione dopo che quest'ultimo ha cambiato nome – da Smart Projects a Arduino srl – alla società che in Italia produce le board a marchio Arduino affidando al CEO Federico Musto la guida della società. La strategia illustrata nella presentazione alla Maker Faire dovrebbe superare l'impasse legato alle varie cause legali in corso, dando a Massimo



I due stand "Arduino" presenti alla Maker Faire di San Mateo: in alto quello di Arduino.cc con Massimo Banzi, i ragazzi di Sharebot, quelli di Primo e la nuova board SmartEverything con demo Sigfox. In basso, lo stand di Arduino.org con i ragazzi di Linino, quelli dell'Università di Messina e Federico Musto.



Banzi la possibilità di continuare il percorso intrapreso nel 2005; una sorta di "Independence Day", come lo ha definito lo stesso Banzi.

Il giorno successivo Massimo Banzi ci ha concesso una video intervista nella quale spiega alla community italiana le ragioni di questa scelta, dovuta sostanzialmente ai tempi troppo lunghi della giustizia che avrebbero tarpato le ali all'evoluzione del progetto.

Già un paio di mesi fa, Banzi aveva rotto il silenzio sull'argomento, portando a conoscenza dei fans di Arduino quanto sta-

va succedendo e rivendicando la paternità del marchio, anche a seguito delle dichiarazioni del nuovo CEO di Arduino srl Federico Musto.

LA STORIA

Il lancio del nuovo marchio e le strategie annunciate non si spiegano se non si conosce la storia del progetto Arduino e le vicende che sono accadute negli ultimi mesi.

Il progetto Arduino nasce nel 2005 presso l'Interaction Design Institute di Ivrea, fondato da Olivetti e Telecom Italia, dall'esigenza di mettere a disposizio-

ne degli studenti che seguivano i corsi di Interactive Design e che non avevano alcuna competenza in campo elettronico e informatico uno strumento semplice per lavorare con i microcontrollori. Come ci ha spiegato lo stesso Banzi (allora docente di quella scuola) in una precedente intervista, dopo varie prove con i prodotti disponibili in commercio allora, ed in considerazione delle difficoltà che gli studenti incontravano, un gruppo di docenti e studenti decise di realizzare un prodotto proprio ed un software di sviluppo particolarmente semplice. Prese così vita il progetto Arduino (dal nome di un bar nelle vicinanze della scuola).

Per far sì che anche altri studenti nelle stesse condizioni di quelli di Ivrea potessero utilizzare questo lavoro, il team decise di mettere a disposizione di tutti sia il software che l'hardware, distribuendo il progetto nei termini delle licenze GPLv2 e Creative Commons Attribution-ShareAlike; l'unica cosa che il team volle proteggere - mantenendolo per sé - fu il marchio e il logo Arduino. Le prime schede vennero prodotte dalla società di Gianluca Martino che allora era titolare di una piccola azienda di produzione di Strambino (TO) e che contribuì al progetto portando la propria esperienza lavorativa nel manufacturing.

Gli altri componenti del team erano David Cuartielles, Tom Igoe e David Mellis.

Il progetto, grazie alle sue qualità uniche, ha avuto un successo planetario, contribuendo alla diffusione della cultura dell'open hardware e della condivisione delle conoscenze, in questo favorito anche dallo sviluppo di Internet. Ma il vero



I componenti del team Arduino in un'immagine di alcuni anni fa.

motore del successo è stato sicuramente Massimo Banzi, l'unico del gruppo a "metterci la faccia" e a portare avanti con costanza lo spirito dell'open-source hardware, coinvolgendo in questa sua visione tutti i principali produttori di semiconduttori e di software, da Intel a Microsoft, società tutt'altro che "aperte".

E dando vita ad una comunità che ha sviluppato e condiviso in rete migliaia e migliaia di progetti.

Questa sua visione, con la delocalizzazione non solo della progettazione ma anche della produzione di alcune board, non poteva piacere a chi in Italia produceva le schede, ovvero a Gianluca Martino.

Per quanto riguarda il marchio, questo risulta registrato negli Stati Uniti nel 2008 da Arduino LLC (partecipata dai 5 fondatori) mentre in Italia, è stato registrato (all'insaputa degli altri componenti del team, almeno questa è la versione di Banzi) da Gianluca Martino il quale, dopo la definitiva rottura con gli altri componenti, ha cambiato nome alla società, da Smart Projects ad Arduino srl, ha aperto il nuovo sito di

e-commerce arduino.org (molto simile nella grafica ad arduino.cc) ed affidato a Federico Musto il ruolo di CEO della nuova società. A Musto ha anche venduto il 95% della nuova società. Il tutto senza più riconoscere agli altri fondatori del progetto Arduino alcuna royalties (sempre secondo Banzi).

Sono così iniziate le dispute legali – ancora in corso – col tentativo di Banzi & Co. di rientrare in possesso del marchio. Dal canto suo Federico Musto



Il marchio col quale verranno commercializzati i prodotti fuori dagli USA.



L'altro logo a supporto dei prodotti di arduino.cc

rivendica la piena disponibilità del marchio Arduino a cui vuole dare, come spiegato al Corriere della Sera pochi mesi fa, "Una dimensione industriale e globale" con un incremento del fatturato nel giro di un paio d'anni dagli attuali 15 milioni di euro a circa 50 milioni, e con la possibilità di quotarsi in Borsa per trovare nuove occasioni di crescita.

Attualmente, dunque, chi vuole acquistare una scheda col marchio Arduino deve rivolgersi alla società di Musto-Martino, la quale continua nello sviluppo del progetto appoggiandosi agli ingegneri della società Dog Hunter (quella di Linino) e ad una start-up dell'Università di Messina.

Probabilmente, se le cose andranno come annunciato da Banzi, qualcosa nei prossimi

mesi cambierà, a iniziare dal mercato americano.

DA ARDUINO A GENUINO

Ma cosa ha detto esattamente Banzi a San Mateo?

- 1) Che per quanto riguarda il mercato USA (dove il marchio Arduino appartiene al vecchio team) e che rappresenta almeno il 50% del business globale, è stato stipulato un accordo con Adafruit per la produzione di tutte le board Arduino, a iniziare da Arduino Uno. Effettivamente nei giorni scorsi sul blog di Adafruit sono apparse le foto delle prime schede prodotte, autografate da Massimo Banzi e Lady Ada.
- 2) Nel resto del mondo i prodotti vecchi e nuovi verranno commercializzati col marchio Genuino e saranno prodotti

da aziende locali con le quali si stanno stringendo accordi. Per quanto riguarda l'Italia sono in corso trattative con un paio di società.

- 3) Lo sviluppo dei nuovi prodotti e dell'IDE continuerà negli uffici sparsi un po' in tutto il mondo dove Arduino ha alcune sedi, a Torino, negli USA, in Svizzera, Svezia e in India.

Banzi ha anche annunciato il primo prodotto progettato e realizzato nella sede di Bangalore, in India: Modulino, mentre la tanto attesa board Arduino Zero (la versione a 32 bit) dovrebbe essere disponibile negli USA dal 9 giugno.

Nelle prossime settimane sono attesi altri annunci, sia relativi a nuovi prodotti che a partnership produttive.

A breve sarà anche disponibile la nuova versione dell'IDE che dovrebbe essere aperta a prodotti di terze parti.

Questa strategia consente a Massimo Banzi di prescindere dalle azioni legali in corso dal momento che l'utilizzo del marchio Genuino non infrange alcuna legge.

Le parole di Banzi sono state seguite con grande attenzione da una sala stracolma e accom-



Massimo Banzi e Limor Fried, proprietaria e fondatrice di Adafruit, mostrano una delle prime board Arduino Uno uscite dalle linee di produzione di Adafruit e pronta per essere commercializzata sul mercato USA.



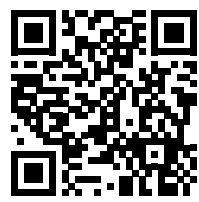
pagnate con applausi d'approvazione da parte del pubblico. *"Alla fine la presentazione è andata molto bene, c'era un sacco di gente, ... a dir la verità mi sono anche un po' commosso."* Ci ha detto Massimo Banzi il giorno dopo *"In questi mesi pensi sempre a come reagirà la comunità a queste notizie, poi quando vedi che la sala si riempie molto più del solito e le persone ti urlano delle parole d'incoraggiamento capisci che la comunità è ancora con te e questa è la cosa importante"*. A pochi passi da noi, i due stand, quello di Arduino.org e quello di Arduino.cc con la maggior parte dei visitatori della Maker Faire all'oscuro di questa disputa. Dal canto suo, Federico Musto presente allo stand di arduino.org, non dà per scontata la soluzione prospettata da Banzi: *"Anche negli USA è in corso una vertenza legale relativa al marchio e non è detto che Banzi possa agire come ha appena annunciato"*.

Nell'intervista che ci ha concesso, Massimo Banzi ribadisce con forza quale sarà la nuova strategia per il futuro: *"Ieri abbiamo fatto questo annuncio per chiarire cosa faremo nei prossimi mesi; molti nella comunità di Arduino sanno che ci sono dei problemi legali che vanno avanti da un po' di tempo e che noi stiamo cercando di risolvere. Da un certo punto di vista però, non è possibile farsi condizionare la vita dagli avvocati, è necessario che le cose vadano avanti."*

Questa storia è molto complessa ma ciononostante sono convinto che alla fine i fatti e la giustizia ci daranno ragione. Tuttavia i tempi della giustizia sono lenti, per cui nel frattempo che la giustizia fa il suo corso noi abbiamo chiarito la situazione per il prossimo futuro:



Massimo Banzi ha tenuto uno speech sullo "Stato di Arduino" sabato 16 maggio presso la Maker Faire di San Mateo annunciando l'accordo con Adafruit e il lancio del nuovo brand Genuino, una strategia in grado di superare i problemi legati al contenzioso legale sul marchio con la società Arduino srl (ex Smart Projects). Il giorno seguente ci ha concesso un'intervista nella quale spiega alla community italiana le ragioni di questa scelta. Il video è disponibile all'indirizzo: <https://youtu.be/wdzL-toqa4I>



qui negli USA dove possediamo il marchio abbiamo stretto un accordo con Adafruit che si occuperà della produzione e della commercializzazione delle schede Arduino; questa società di Manhattan offre le massime garanzie, ha una storia di grandi successi alle spalle ed è stata fondata da una donna che ancora oggi possiede il 100% della società.

Lady Ada, della quale sono molto amico, è partita da zero 10 anni fa, come noi, e non ha mai preso soldi da società di venture capital o altre sovvenzioni. Con Adafruit abbiamo stabilito una partnership strategica molto importante per il mercato USA.

Al di fuori degli Stati Uniti la situazione del brand è un po' più complessa, spesso difficile da capire. Per questo motivo abbiamo pensato ad un nuovo marchio che sia riconducibile al team che ha dato vita al progetto Arduino. Abbiamo così dato vita al marchio

Genuino che utilizzeremo per produrre fuori dagli USA e che i clienti potranno acquistare sul nostro sito arduino.cc; daremo così la possibilità agli appartenenti alla nostra comunità, in qualsiasi paese del mondo vivano, di acquistare i nostri prodotti originali. Abbiamo ricevuto un sacco di mail in questo senso e abbiamo pensato che questa fosse la soluzione per soddisfare al più presto le esigenze di queste persone.

Per quanto riguarda la produzione delle board, ieri ho letto su alcuni social dei commenti relativi al fatto che la produzione verrà spostata fuori dall'Italia. Vorrei precisare che non ho spostato io il manufacturing fuori dall'Italia, è il manufacturing dell'Italia che ha preso un'altra strada e che legalmente mi impedisce di produrre in Italia una board col marchio Arduino. Nel frattempo questa è la soluzione alla quale abbiamo pensato".

GESTIC INCONTRA ARDUINO



Con la board qui descritta, interfacciamo la scheda elettrodi per il riconoscimento dei gesti ad Arduino e Raspberry Pi. Vediamo per prima l'applicazione con Arduino.

di MATTEO DESTRO

Nella precedente puntata abbiamo descritto nei dettagli le elettroniche degli elettrodi e della scheda demo che abbiamo progettato per gestire gli stessi, nonché la scheda di interfaccia tra gli elettrodi e il PC da usare per la parametrizzazione. In questa puntata parleremo del nostro software da interfacciare alla scheda demo e del nuovo elettrodo sviluppato per interfacciarsi ad Arduino Uno Rev. 3 e Raspberry Pi. Per studiare la piattaforma Gestic abbiamo pensato di realizzare un software con cui interfacciarsi alla nostra scheda demo, grazie al quale possiamo vedere in tempo reale le gesture riconosciute dall'integrato MGC3130. Il software GesticTester, come mostrato

dalla **Fig. 1**, è suddiviso in due TAB distinti: il primo (Gesture) monitorizza e visualizza tutte le gesture intercettate dall'integrato MGC3130, mentre il secondo (Firmware) serve per visualizzare la revisione firmware caricata nell'integrato MGC3130. La comunicazione tra PC e scheda demo avviene tramite interfaccia USB; per instaurare una comunicazione si deve prima di tutto selezionare dal menù "*Communication*" la voce "*Open communication*" la quale richiama la finestra di selezione della COM virtuale cui è connessa la scheda demo. La **Fig. 2** mostra la finestra di selezione della COM virtuale. Notate che la COM può essere selezionata manualmente, oppure, più

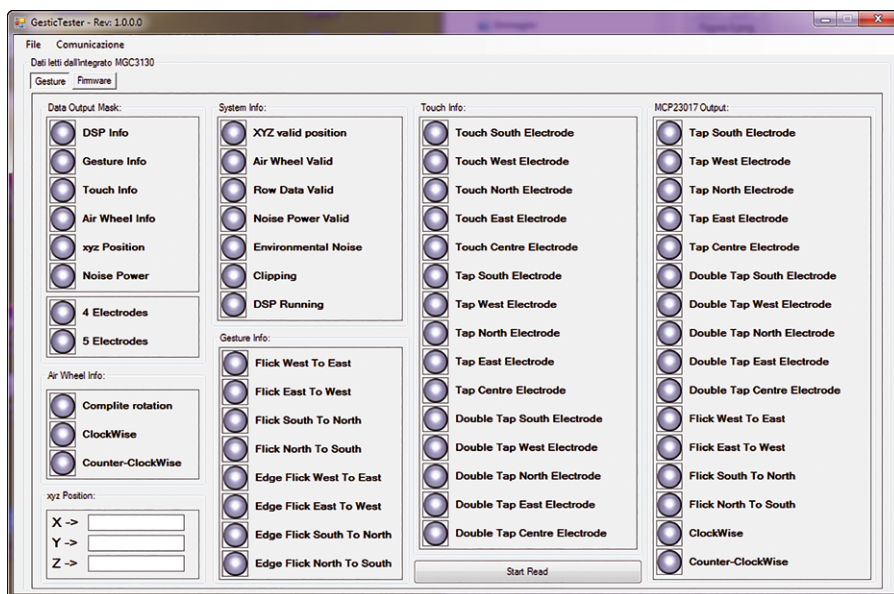


Fig. 1

gesture riportate sulle uscite digitali gestite dall'integrato MCP23017.

UN NUOVO ELETTRODO, ARDUINO E RASPBERRY PI

Per sfruttare le potenzialità dell'integrato MGC3130 abbiamo pensato di sviluppare un nuovo elettrodo con la possibilità di collegare, oltre che la nostra scheda demo vista nella precedente puntata, anche la scheda Arduino Uno Rev3 (oppure la Arduino Leonardo Rev3). Inoltre il nuovo elettrodo è stato pensato e progettato per poter collegare anche le schede Raspberry Pi e in particolare la Raspberry Pi B+ o Raspberry Pi 2. Ovviamente solo una delle schede sopra citate può essere direttamente collegata alla scheda elettrodo; in altre parole, se decidiamo di lavorare con il mondo Arduino non possiamo collegare anche la Raspberry Pi e viceversa. Resta sottinteso che il discorso di unicità di connessione verso l'elettrodo vale anche per la nostra scheda. In questo articolo descriveremo l'accoppiata con Arduino, rimandando al prossimo quella con la Raspberry Pi. Iniziamo con il descrivere lo schema elettrico del nostro nuovo elettrodo partendo dall'integrato MGC3130, il quale è configurato per gestire cinque

comodamente, con la ricerca automatica della scheda che si avvia cliccando sulla voce **"Auto Find Device"**. Sulla destra compare una **"Text box"** dove vengono listati i comandi che il driver di comunicazione invia alla scheda, mentre in basso, se attiva, c'è una seconda **"Text box"** in cui sono listati i byte dei dati che passano sulla COM virtuale (in pratica sono i pacchetti dati che dal PC vengono inviati alla scheda demo e le relative risposte). Per attivare la seconda **"Text box"** si deve cliccare su **"Start Logger"**.

Se la comunicazione viene instaurata con successo, comparirà il nome del device collegato (in questo caso **"Gestic DemoBoard"**) e il relativo FW caricato in esso.

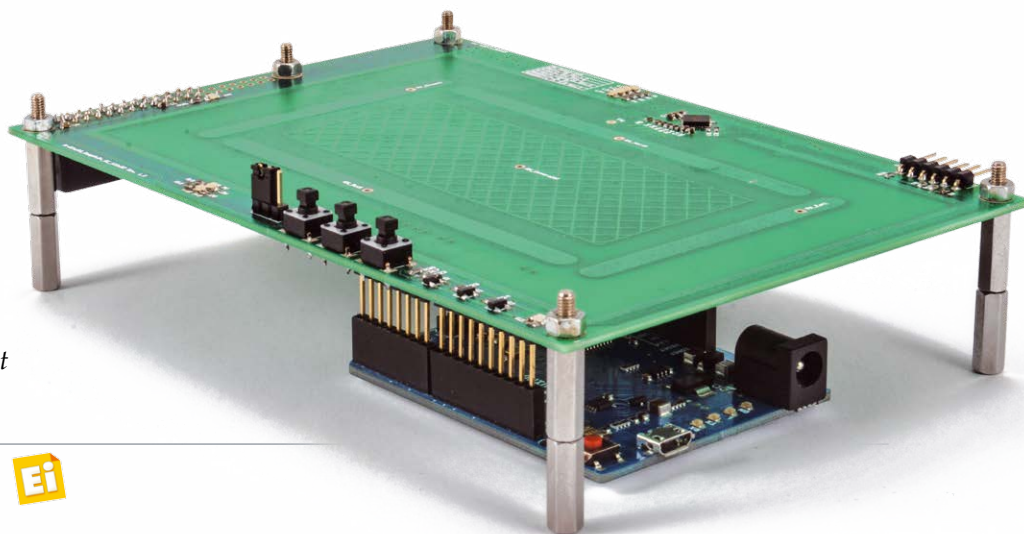
Instaurata la comunicazione con la scheda demo, si può iniziare a monitorare le gesture riconosciute dall'integrato MGC3130. Per attivare la scansione, cliccare sulla voce **"Start Read"** (Fig. 1).

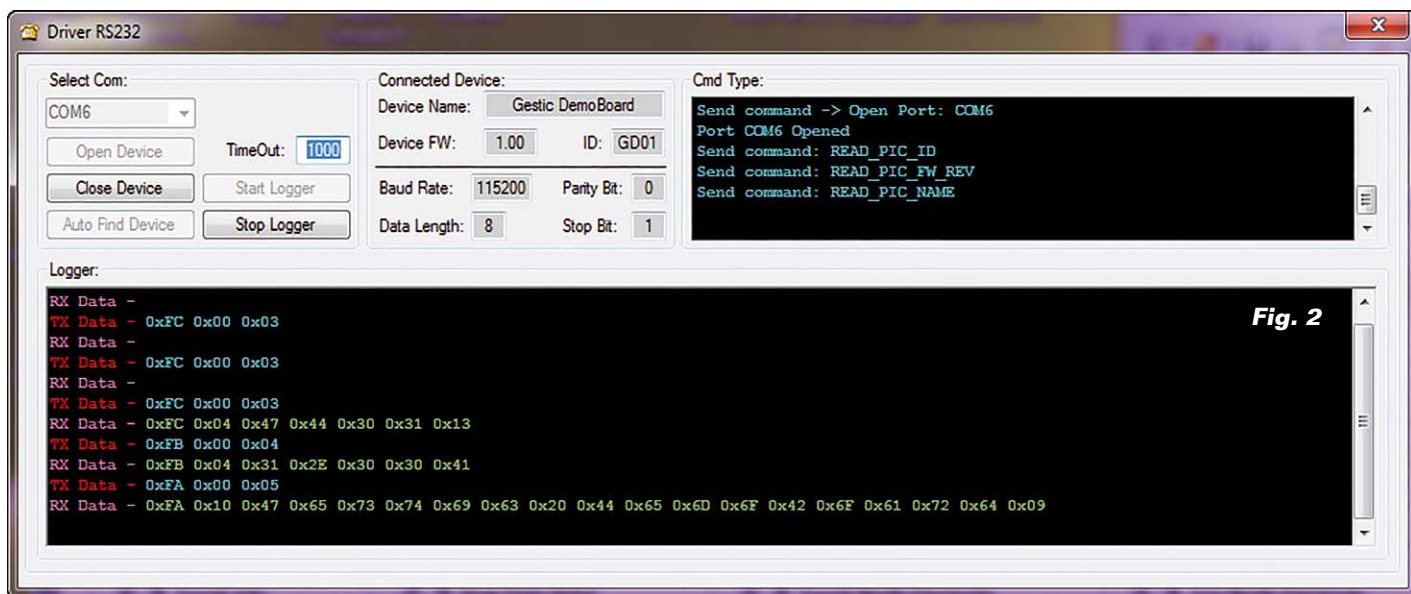
La sezione **"Gesture Info"** evidenzia le gesture di movimento della mano ovvero:

- *movimento da Ovest a Est, compreso movimento Ovest – Est sul bordo (Edge Flick);*

- *Movimento da Est a Ovest, compreso movimento Est – Ovest sul bordo (Edge Flick);*
- *Movimento da Sud a Nord, compreso movimento Sud – Nord sul bordo (Edge Flick);*
- *Movimento da Nord a Sud, compreso movimento Nord – Sud sul bordo (Edge Flick).*

La sezione **"Touch Info"** evidenzia le gesture di tocco sugli elettrodi; in questo caso si identificano le gesture **"Touch"**, **"Tap"** e **"Double Tap"** per un totale di quindici diverse gesture. Vi ricordiamo che con il termine Tap si intende toccare l'elettrodo con un colpetto veloce, che è diverso dalla gesture Touch, la quale è un movimento più lento. Infine la sezione **"MCP23017 Output"** la quale evidenzia le



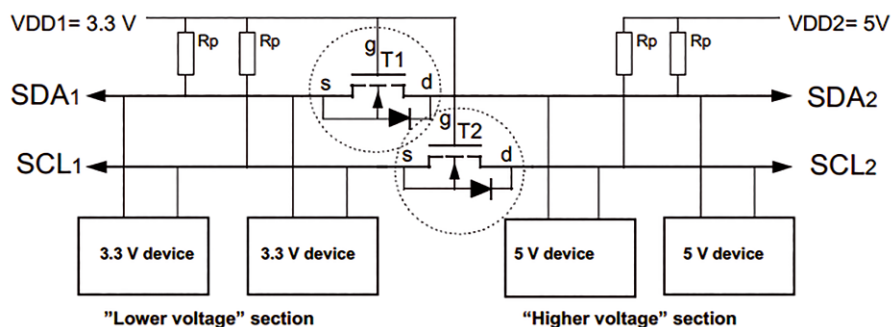


elettrodi di ricezione (R_x) e un elettrodo di trasmissione (T_x). Per ognuno degli elettrodi di ricezione è stata predisposta una resistenza da 10 k Ω , la quale, grazie alla capacità di ingresso dei pin R_{x0} , R_{x1} , R_{x2} , R_{x3} e R_{x4} crea un filtro passa basso eliminando le interferenze ad alta frequenza. Anche per l'elettrodo T_x è stata predisposta una resistenza, che attualmente è un ponticello (0 Ω). L'interfaccia di comunicazione utilizzata dall'integrato MGC3130 è il consueto bus I²C, più le due linee di comunicazione TS (EIO0) e RESET, il quale viene portato sia all'interfaccia di connessione verso le schede Arduino (U2) che all'interfaccia di connessione della scheda Raspberry Pi (U3). Tali linee vengono ovviamente portate anche ai connettori CN1 e CN2 per la connessione della nostra scheda demo. Ricordiamo che la scheda è collegabile sia utilizzando un cavo Wurth da 10 poli (passo 1mm, lunghezza 50 mm) al connettore CN2 oppure direttamente tramite connettore CN1. Dato che l'integrato MGC3130 è alimentato a +3,3 Vcc si rende necessario adattare i segnali verso l'elettronica delle schede

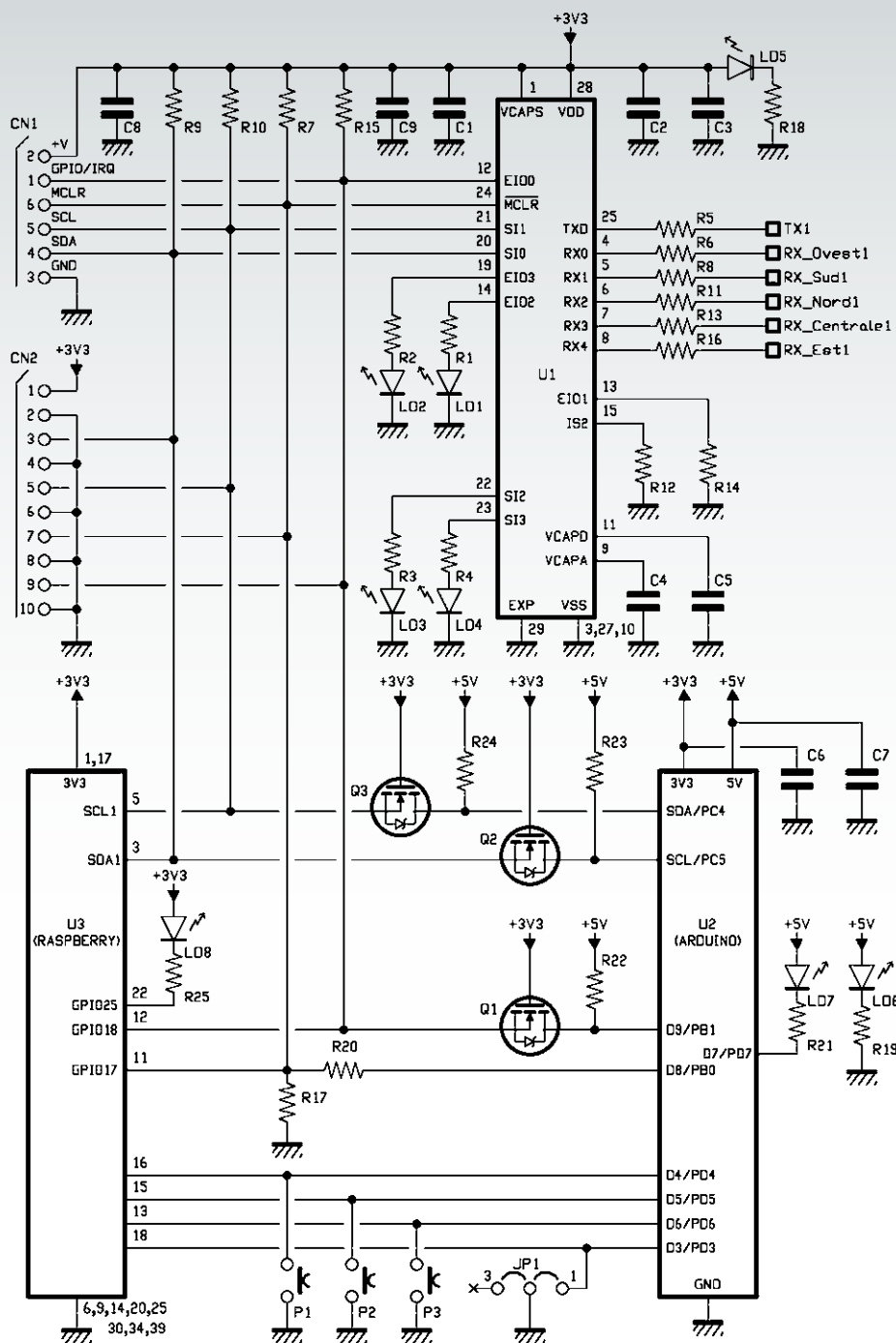
Arduino in quanto alimentate a +5 Vcc. L'adattamento di linea viene fatto sfruttando i MOSFET BSS123 a canale N (Q1, Q2 e Q3). Vediamo in pratica come funziona l'adattamento di linea commentando la Fig. 3. Un traslatore di livello bidirezionale è utilizzato per interconnettere due sezioni di un BUS I²C (vale anche per SPI o qualsiasi linea bidirezionale) dove ogni sezione è alimentata con un livello di tensione differente, nel nostro caso +3,3 Vcc per l'elettronica dell'elettrodo e +5 Vcc per quanto riguarda le schede Arduino. La sezione a bassa tensione, a sinistra, prevede due resistenze di pull-up collegate al source del MOSFET, per le linee SDA e SCL, mentre il gate viene

collegato direttamente alla tensione di alimentazione più bassa, nel nostro caso +3,3 V. La sezione ad alta tensione prevede anch'essa due resistenze di pull-up collegate al drain del MOSFET. Per entrambe le sezioni le periferiche devono essere configurate come uscite open-drain. Il MOSFET utilizzato per entrambe le linee è un MOSFET ad accrescimento canale N con il substrato internamente connesso al source. Inoltre si devono scegliere MOSFET con già integrato il diodo che connette il substrato con il drain creando una giunzione n-p. I MOSFET BSS123 da noi scelti soddisfano queste condizioni. Vediamo ora in dettaglio i tre stati di funzionamento (il ragionamento lo applichiamo a

Fig. 3



[schema ELETTRICO]



una sola linea ma ovviamente vale per entrambe).

1. Nessuna periferica impone un livello di tensione basso sulla linea. Quindi sul lato bassa tensione la resistenza di pull-up impone un livello logico alto, di conseguenza la V_{GS} non supera la soglia del

MOSFET, in quanto V_G e V_S hanno la medesima tensione, quindi il MOSFET non entra in conduzione; di conseguenza sul lato ad alta tensione la linea si ritrova a un livello logico alto imposto dalla rispettiva resistenza di pull-up.

2. La periferica a bassa tensione

impone sulla linea un livello di tensione basso di conseguenza il source del MOSFET va a livello logico basso. A questo punto la V_{GS} supera la soglia e manda in conduzione il MOSFET; con il MOSFET in conduzione, il livello logico basso viene imposto anche alla sezione ad alta tensione.

3. La periferica ad alta tensione impone sulla linea un livello di tensione basso; sulla linea a bassa tensione abbiamo la resistenza di pull-up che impone un livello logico alto e quindi manda in conduzione il diodo presente nel MOSFET. La V_S scende e di conseguenza la V_{GS} supera la soglia mandando in conduzione il MOSFET; a questo punto le due sezioni hanno il medesimo livello di tensione sulla linea.

Il concetto appena esposto è stato applicato alle due linee I²C, SDA e SCL, e alla linea TS. Per quanto riguarda la linea RESET, è sufficiente la R20 assieme alla resistenza di pull-up R7. La R17 non deve essere montata. Le linee del BUS I²C che vengono portate alla nostra scheda demo e alla scheda Raspberry Pi non serve che siano adattate, in quanto le elettroniche lavorano a +3,3 V e sono compatibili con l'integrato MGC3130.

Sia alla scheda Arduino che alla Raspberry Pi vengono collegati tre pulsanti (P1, P2 e P3) e un jumper; questi servono per riprodurre le funzioni implementate nella nostra scheda demo.

Non sono state predisposte le resistenze di pull-up perché sono già integrate sia nella scheda Arduino che nella scheda Raspberry Pi, infatti

si possono attivare da codice quando si configurano i pin del microcontrollore come ingressi. Sia per la scheda Arduino che per la scheda Raspberry Pi abbiamo predisposto un LED di segnalazione, LD7 per Arduino e LD8 per Raspberry Pi, utile durante il riconoscimento delle Gesture o durante la gestione della pressione dei pulsanti P1, P2 e P3.

Con questa nuova scheda abbiamo anche colto l'occasione per sfruttare gli I/O estesi EIO2, EIO3, EIO6 e EIO7 ai quali abbiamo collegato dei LED per segnalare le gesture riconosciute. Durante la parametrizzazione è quindi possibile decidere quale gesture deve essere monitorata e riportata su una delle possibili uscite o una combinazione di esse. Per la nostra applicazione abbiamo scelto di riportare sui LED le seguenti gesture:

- *Flick West – East;*
- *Flick East – West;*
- *Flick North – South;*
- *Flick South – North;*
- *Single Tap North;*
- *Single Tap South;*
- *Single Tap West;*
- *Single Tap East;*
- *Single Tap Centre;*
- *Clock Wise;*
- *Counter Clock wise.*

La **Fig. 4** evidenzia la configurazione delle uscite per ogni gesture di cui sopra. Ad ogni gesture riconosciuta l'integrato MGC3130 genera un impulso sulla rispettiva uscita. Tuttavia vogliamo fare notare che è possibile scegliere e configurare diversamente il comportamento delle uscite rispetto alla gesture riconosciuta. Infatti oltre all'impulso è possibile scegliere:

- *uscita permanentemente alta;*
- *uscita permanentemente bassa;*
- *toggle.*

Oltre alla libreria scritta per la gestione dell'integrato MGC3130 è stata pensata e realizzata una libreria dedicata per la gestione dell'integrato MCP23017, che è un I/O expander della Microchip su bus I²C. Questo componente è lo stesso montato sulle schede d'espansione a relé FT1079K descritte in queste pagine. La libreria è composta da tre file distinti "**MCP23017.c**", "**MCP23017.h**" e "**keyword.txt**". La libreria mette a disposizione una serie di funzioni utili alla gestione dell'integrato MCP23017 tra cui:

```
void begin(void);
void ToggleSingleBit(uint8_t RegAdd, uint8_t Bit, uint8_t i2caddr);
void SetSingleBit(uint8_t RegAdd, uint8_t Bit, uint8_t i2caddr);
void ResetSingleBit(uint8_t RegAdd, uint8_t Bit, uint8_t i2caddr);
void WriteSingleReg(uint8_t RegAdd, uint8_t RegData, uint8_t i2caddr);
void ClearReg(uint8_t RegAdd, uint8_t i2caddr);
uint8_t ReadSingleReg(uint8_t RegAdd, uint8_t i2caddr);
```

La funzione "**Begin**" serve per inizializzare la periferica I²C, la funzione "**ToggleSingleBit**" permette di eseguire il toggle su un singolo bit del registro desiderato. Infatti i parametri da passare alla funzione sono l'indirizzo del registro su cui eseguire il toggle, il bit da modificare e infine l'indirizzo hardware dell'integrato. La funzione "**SetSingleBit**" setta un singolo bit del registro desiderato, i parametri da passare sono gli stessi della precedente funzione. Viceversa la funzione "**ResetSingleBit**" che come suggerisce il nome resetta un singolo bit.

Diversamente le funzioni "**WriteSingleReg**" e "**ClearReg**" le quali rispettivamente scrivono un byte sul registro desiderato oppure resettano il registro desiderato. I parametri da passare sono i soliti: indirizzo del registro, dato che si vuole scrivere e indirizzo hardware dell'integrato.

Per ultimo "**ReadSingleReg**" la quale permette di leggere un registro dell'integrato. La funzione ritorna il byte letto, i parametri da passare sono l'indirizzo del registro da leggere e l'indirizzo hardware dell'integrato.

Oltre alle funzioni appena descritte ce ne sono altre necessarie a configurare l'integrato per il suo corretto funzionamento. A seconda della configurazione dei banchi, vedere datasheet, la sequenza di configurazione cambia. Supponendo di ragionare sul BANK0 le funzioni di inizializzazione sono le seguenti:

```
void SetAllRegBank0(uint8_t i2caddr);
void SetIntPortABank0(uint8_t Iocon, uint8_t Intcon, uint8_t Defval,
uint8_t GpintEn, uint8_t i2caddr);
void SetIntPortBBank0(uint8_t Iocon, uint8_t Intcon, uint8_t Defval,
uint8_t GpintEn, uint8_t i2caddr);
```

La prima funzione setta i registri dell'integrato; il parametro da passare è l'indirizzo hardware dell'integrato. Seguono poi due funzioni per configurare l'interrupt sia per la porta A che per la porta B, a patto che siano configurate come ingressi. I byte di configurazione da passare a ogni singolo registro si trovano nel file .h (MCP23017.h), l'utente può configurarli a seconda delle sue necessità.

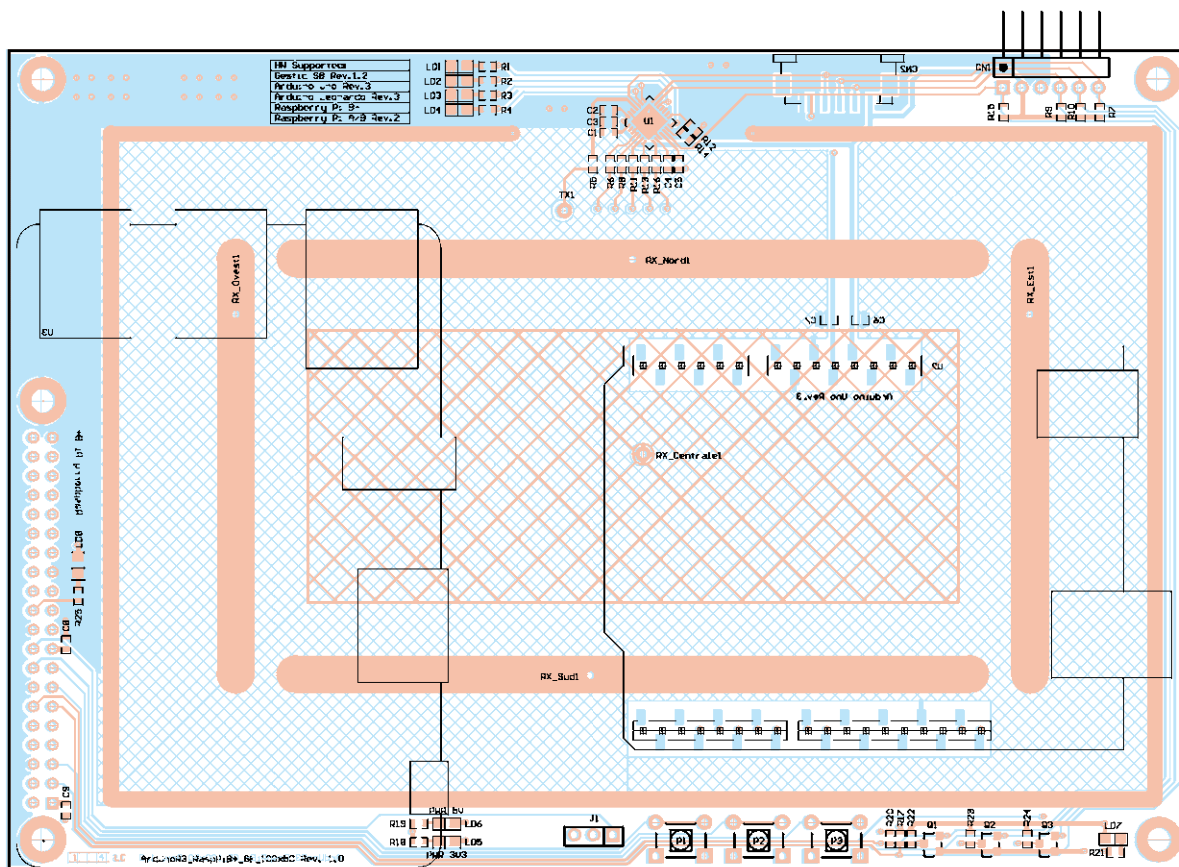
Sarà poi la funzione "**SetAllRegBank0**" a passare ogni singolo valore ai registri da configurare. Solitamente si usa sempre la configurazione dei registri come BANK0. Nel nostro sketch, di cui parleremo a breve, abbiamo fatto uso di questa libreria, durante la fase di configurazione dei due integrati (due schede FT1079K) abbiamo usato le seguenti:

```
mcp23017.Begin();
mcp23017.SetAllRegBank0(0x00);
mcp23017.ClearReg(MCP23017_BNK0_OLATA, 0x00);
mcp23017.SetAllRegBank0(0x01);
mcp23017.ClearReg(MCP23017_BNK0_OLATA, 0x01);
```

Mentre per gestire le uscite abbiamo usato le seguenti funzioni:

```
mcp23017.ToggleSingleBit(MCP23017_BNK0_OLATA, 0, 0x00);
mcp23017.SetSingleBit(MCP23017_BNK0_OLATA, 0, 0x00);
mcp23017.ResetSingleBit(MCP23017_BNK0_OLATA, 0, 0x00);
```

Per chi volesse cimentarsi con la nostra libreria per studiarla e usarla nei propri sketch può contare su quattro sketch di esempio distribuiti assieme alla libreria.



Elenco Componenti:

R1 ÷ R4: 470 ohm 1% (0603)
 R5: 0 ohm (0603)
 R6 ÷ R8: 10 kohm 1% (0603)
 R9, R10: 1,8 kohm 1% (0603)
 R11 ÷ R16: 10 kohm 1% (0603)
 R17: -

R18: 470 ohm 1% (0603)
 R19, R21: 680 ohm 1% (0603)
 R20: 5,2 kohm 1% (0603)
 R22: 10 kohm 1% (0603)
 R23, R24: 1,8 kohm 1% (0603)
 R25: 680 ohm 1% (0603)

C1, C2: 100 nF ceramico (0603)
 C3: 10 μ F 6,3 VL ceramico (0603)
 C4, C5: 4,7 μ F 6,3 VL ceramico (0603)
 C6 ÷ C9: 10 μ F 6,3 VL ceramico (0603)
 Q1 ÷ Q3: BSS123
 U1: MGC3130-I/MQ

LIBRERIA ARDUINO MGC3130

In questo articolo ci concentreremo sull'accoppiata del nostro nuovo elettrodo con le schede Arduino Uno Rev.3 e Arduino Leonardo Rev.3. Per entrambe abbiamo scritto due demo che si appoggiano sulla nostra libreria di gestione dell'integrato MGC3130. La demo scritta per la scheda Arduino Uno Rev.3 viene completata dalla scheda di

espansione FT1079K, sviluppata e venduta da Futura Elettronica (www.futuraishop.it) la quale mette a disposizione 8 ingressi digitali e altrettante uscite a relé. Per la nostra demo useremo solo le otto uscite a relé, tuttavia, per chi lo desiderasse, è possibile gestire anche gli ingressi modificando opportunamente il codice della demo. La gestione degli I/O viene fatta sfruttando l'integrato

Microchip MCP23017, il quale viene connesso alle schede Arduino tramite il bus I²C come l'integrato MGC3130. Anche per l'MCP23017 abbiamo scritto una libreria di supporto che andremo a descrivere in breve durante l'articolo. Grazie alle uscite a relé messe a disposizione, possiamo riportare le gesture riconosciute su una delle possibili uscite. Per la nostra demo abbiamo

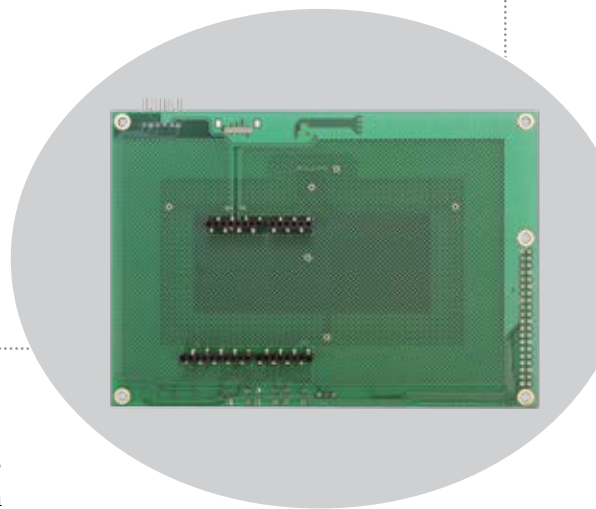


P1 ÷ P3: Microswitch

Varie:

- Connettore 6 vie
- Connettore 8 vie (2 pz.)
- Connettore 10 vie

- Connettore WR-FPC 10 vie
- Pin strip femmina 2x20 vie
- Pin strip maschio 3 vie
- Jumper
- Pin strip maschio 6 vie 90°
- Circuito stampato S1214



deciso di riportare fino a un massimo di sedici gesture, per un totale di due schede FT1079K. Ovviamente è possibile aggiungere altre FT1079K per riportare più gesture possibili sulle uscite a relé (l'aggiunta di più di due schede FT1079K necessita di apportare modifiche allo sketch da noi scritto e può essere visto come un esercizio didattico interessante). Invece la demo che riguarda

la scheda Leonardo Rev.3 ci permette di interagire con il PC e in particolare con un programma di visualizzazione delle immagini. Grazie alle gesture riconosciute dall'integrato, sarà possibile sfogliare le immagini presenti sul PC. Iniziamo a descrivere la libreria Arduino per la gestione dell'integrato MGC3130: essa è composta da un file con estensione .cpp (*MGC3130.cpp*)

e da un file .h (*MGC3130.h*) contenente le dichiarazioni delle variabili e delle funzioni presenti nel file .cpp. Oltre a questi c'è un file .txt (*keywords.txt*) con i riferimenti alla funzioni pubbliche utilizzate negli sketch Arduino. Le funzioni pubbliche messe a disposizione

Event	Gesture/Electrode	Pin	Action
Gesture	Flick West -> East	EIO1 ☐ EIO2 ☒ EIO3 ☐ EIO6 ☐ EIO7 ☐	pulse Clear
Gesture	Flick East -> West	EIO1 ☐ EIO2 ☐ EIO3 ☒ EIO6 ☐ EIO7 ☐	pulse Clear
Gesture	Flick North -> South	EIO1 ☐ EIO2 ☐ EIO3 ☐ EIO6 ☒ EIO7 ☐	pulse Clear
Gesture	Flick South -> North	EIO1 ☐ EIO2 ☐ EIO3 ☐ EIO6 ☐ EIO7 ☒	pulse Clear
Single Tap	North	EIO1 ☐ EIO2 ☒ EIO3 ☒ EIO6 ☐ EIO7 ☐	pulse Clear
Single Tap	South	EIO1 ☐ EIO2 ☐ EIO3 ☒ EIO6 ☒ EIO7 ☐	pulse Clear
Single Tap	West	EIO1 ☐ EIO2 ☐ EIO3 ☐ EIO6 ☒ EIO7 ☒	pulse Clear
Single Tap	East	EIO1 ☐ EIO2 ☒ EIO3 ☒ EIO6 ☐ EIO7 ☐	pulse Clear
Single Tap	Center	EIO1 ☐ EIO2 ☐ EIO3 ☒ EIO6 ☒ EIO7 ☐	pulse Clear
AirWheel	Clock Wise	EIO1 ☐ EIO2 ☒ EIO3 ☒ EIO6 ☐ EIO7 ☐	pulse Clear
AirWheel	Counter Clock Wise	EIO1 ☐ EIO2 ☐ EIO3 ☒ EIO6 ☒ EIO7 ☐	pulse Clear
select	select	EIO1 ☐ EIO2 ☐ EIO3 ☐ EIO6 ☐ EIO7 ☐	select Clear

Pin Configuration during Start-Up/Sleep					
	EIO1	EIO2	EIO3	EIO6	EIO7
Level at Startup Up	Low	Low	Low	Low	Low
State during Self-Wakeup / Deep Sleep	Unchanged	Unchanged	Unchanged	Unchanged	Unchanged

Fig. 4

dalla libreria sono le seguenti:

```
void SetSerial( uint8_t Baud, uint8_t Config)
void SetAdd(uint8_t Addr)
void ResetDevice(uint8_t Rst)
void ExitResetDevice(uint8_t Rst)
void Begin( uint8_t Ts, uint8_t Rst)
boolean GetTsLineStatus( uint8_t Ts)
void ReleaseTsLine( uint8_t Ts)
void GetEvent(void)
void DecodeGesture(void)
```

La funzione *“SetSerial”* serve semplicemente per inizializzare la comunicazione seriale da utilizzarsi con il *“monitor seriale”* presente nello IDE per eseguire il monitoraggio delle gesture riconosciute dall’integrato MGC3130. I parametri da passare alla funzione sono la velocità di comunicazione che si vuole utilizzare e la configurazione dei dati, ovvero lunghezza pacchetto dati, bit di parità bit, di stop ecc. La funzione *“SetAdd”* serve per assegnare l’indirizzo hardware all’integrato MGC3130: nel nostro caso è sempre e solo 0x42. La funzione *“ResetDevice”* serve per mantenere resettato l’integrato MGC3130, fintantoché non viene rimossa la condizione di reset. L’unico parametro da passare alla funzione è il pin cui è connessa la linea di

RESET. La funzione opposta è *“ExitResetDevice”* la quale rimuove la condizione di RESET per l’integrato MGC3130; come per la precedente funzione, l’unico parametro è il pin a cui è collegata la linea di RESET. La funzione *“Begin”* serve per inizializzare le linee di comunicazione tra la scheda Arduino e MGC3130, come parametri necessita soltanto i riferimenti per la linea TS e RESET. La funzione inizializza la periferica I²C con la consueta funzione *“Wire.begin”* cui si deve passare l’indirizzo hardware dell’integrato MGC3130. La nostra libreria per gestire la comunicazione I²C sfrutta la libreria standard di sistema *“Wire”* includendo nel file .cpp il file di definizione *“Wire.h”*. Oltre all’inizializzazione dell’interfaccia I²C, si definiscono e configurano le linee RESET (Output) e TS (Input). La linea di RESET viene mantenuta a livello basso per un tempo di 250 ms, consentendo così la stabilizzazione dell’alimentazione al power-up.

Per indicare l’inizio e la fine della sequenza di inizializzazione, vengono stampati sul monitor seriale una serie di stringhe di testo.

Per scrivere del testo sul monitor seriale si utilizza la direttiva *“print”* o *“println”* presenti nella libreria *“Serial”*. Tuttavia ogni qual volta si vuole scrivere una stringa di testo utilizzando queste direttive si spreca della memoria SRAM in quanto la stringa viene memorizzata proprio in questo tipo di memoria.

Per ovviare a questo inconveniente si può pensare di salvare le stringhe di testo nella memoria Flash, per poi andare a rileggerle e passarle alla funzione *“print”* come array di byte evitando di sprecare memoria SRAM. Quindi per memorizzare in Flash le stringhe di testo si può usare la seguente direttiva:

```
const char MGC3130Ready[] PROGMEM =
“MGC3130 device is ready”;
```

Per rileggere la stringa memorizzata utilizziamo una funzione detta *“pgm_read_byte_near”* alla quale dobbiamo passare l’indirizzo di memoria Flash in cui è memorizzato il dato da leggere. Abbiamo quindi scritto una nostra funzione che, tramite un loop, rilegge tutti i byte che compongono la stringa e la stampano sul monitor seriale. Alla funzione bisogna quindi passare un puntatore in Flash che identifica l’inizio della stringa, la lunghezza della stringa e un boolean che indica alla funzione se alla fine deve stampare anche il carattere *“\n”*. Quindi, concludendo l’esempio della stringa di cui sopra, avremo:

```
ReadStringFLASH((uint8_t *)MGC3130Ready,
strlen(MGC3130Ready), TRUE);
```

La funzione *"GetTsLineStatus"* serve per monitorare la linea TS e rilevare quando l'integrato MGC3130 la impegna per avvisare che è stata riconosciuta una gesture. Non appena il sistema si accorge che la linea TS è stata impegnata, cambia lo stato del pin andando esso stesso ad occupare la linea TS, potendo così iniziare la procedura di lettura dei dati dall'integrato MGC3130. I dettagli sulla gestione della linea TS li trovate nell'articolo pubblicato nel fascicolo n° 195. Conclusa la lettura dei dati, si può richiamare la funzione *"ReleaseTsLine"* rilasciando così la linea Ts.

Per entrambe le funzioni appena esposte, l'unico parametro che bisogna passargli è il pin a cui è collegata la linea TS. La funzione *"GetEvent"* serve per leggere i dati memorizzati nel buffer dell'integrato MGC3130 e salvarli in apposite strutture dati utilizzate dalla libreria; sarà poi una successiva funzione, quella che provvederà a decodificare i dati ricevuti e a renderli disponibili all'utente finale per la propria applicazione.

Vi ricordiamo che il pacchetto dati utilizzato per la lettura dei dati caricati nel buffer dell'integrato MGC3130 è del tipo mostrato in Fig. 5. L'indirizzo contenuto nel pacchetto è quello hardware dell'integrato dove il bit meno significativo indica se si sta eseguendo una scrittura ("0") oppure una lettura ("1"). La sezione *"MGC3130 message"* può essere espansa come indicato in Fig. 6 identificando due sezioni *"Header"* e *"Payload"*. Espandendo ancora la sezione *"Header"*, come in Fig. 7, si nota che nei primi quattro byte ritroviamo la lunghezza

del pacchetto dati ricevuto e, cosa molto importante, l'ID che identifica il pacchetto dati. Nel caso di lettura dei dati dopo il riconoscimento di una gesture, l'ID del pacchetto sarà 0x91. Per l'elenco dei codici ID disponibili vi invitiamo a rileggere l'articolo nel fascicolo n° 195.

Dei dati letti dalla funzione *"GetEvent"* durante il riconoscimento delle gesture, solo una parte viene tenuta in considerazione e in particolare le *"GestureInfo"* e le *"TouchInfo"*. Queste vengono ulteriormente filtrate da delle costanti di mascheratura *"MASK_GESTURE_RAW"* e *"MASK_TOUCH_RAW"* eliminando le parti non necessarie. Per completezza vengono anche salvate le informazioni delle coordinate x, y e z. Infine abbiamo la funzione *"DecodeGesture"* la quale decodifica i dati letti dalla precedente funzione e le rende disponibili attraverso una struttura dati pubblica che l'utente può utilizzare nella propria applicazione. Le gesture riconosciute e messe a

disposizione dalla libreria sono:

- *Gesture Touch South;*
- *Gesture Touch West;*
- *Gesture Touch North;*
- *Gesture Touch East;*
- *Gesture Touch Centre;*
- *Gesture Tap South;*
- *Gesture Tap West;*
- *Gesture Tap North;*
- *Gesture Tap East;*
- *Gesture Tap Centre;*
- *Gesture Double Tap South;*
- *Gesture Double Tap West;*
- *Gesture Double Tap North;*
- *Gesture Double Tap East;*
- *Gesture Double Tap Centre;*
- *Flick West to East;*
- *Flick East to West;*
- *Flick South to North;*
- *Flick North to South;*
- *Edge Flick West to East;*
- *Edge Flick East to West;*
- *Edge Flick South to North;*
- *Edge Flick North to South;*
- *Clock Wise;*
- *Counter Clock Wise.*

La libreria si accorge delle variazioni intercettate dall'integrato MGC3130 e le riporta sui rispettivi bit della struttura dati pubblica. La gesture è riconosciuta quando il bit corrispondente va a "1" logico. La

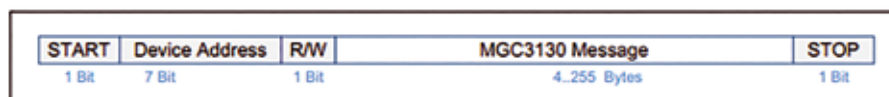


Fig. 5

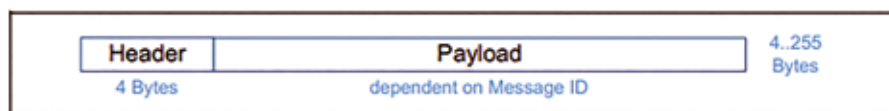


Fig. 6

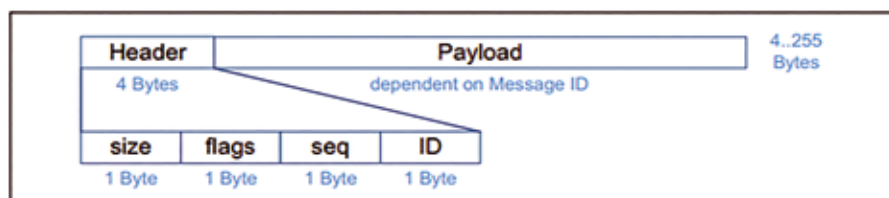


Fig. 7

Listato 1

```

union GestureOutput {
    uint32_t Gesture;
    uint8_t GestArray[4];
    struct {
        uint8_t Byte_0;
        uint8_t Byte_1;
        uint8_t Byte_2;
        uint8_t Byte_3;
    } GestureByte;
    struct {
        uint8_t TouchSouth      :1; // GESTURE_TOUCH_SOUTH      0x00000001
        uint8_t TouchWest       :1; // GESTURE_TOUCH_WEST       0x00000002
        uint8_t TouchNorth      :1; // GESTURE_TOUCH_NORTH      0x00000004
        uint8_t TouchEast       :1; // GESTURE_TOUCH_EAST       0x00000008
        uint8_t TouchCentre     :1; // GESTURE_TOUCH_CENTRE     0x00000010
        uint8_t TapSouth        :1; // GESTURE_TAP_SOUTH        0x00000020
        uint8_t TapWest         :1; // GESTURE_TAP_WEST         0x00000040
        uint8_t TapNorth        :1; // GESTURE_TAP_NORTH        0x00000080
        uint8_t TapEast         :1; // GESTURE_TAP_EAST         0x00000100
        uint8_t TapCentre       :1; // GESTURE_TAP_CENTRE       0x00000200
        uint8_t DoubleTapSouth   :1; // GESTURE_DOUBLE_TAP_SOUTH 0x00000400
        uint8_t DoubleTapWest    :1; // GESTURE_DOUBLE_TAP_WEST  0x00000800
        uint8_t DoubleTapNorth   :1; // GESTURE_DOUBLE_TAP_NORTH 0x00001000
        uint8_t DoubleTapEast    :1; // GESTURE_DOUBLE_TAP_EAST  0x00002000
        uint8_t DoubleTapCentre  :1; // GESTURE_DOUBLE_TAP_CENTRE 0x00004000
        uint8_t GestWestEast     :1; // GESTURE_WEST_EAST        0x00008000
        uint8_t GestEastWest     :1; // GESTURE_EAST_WEST        0x00010000
        uint8_t GestSouthNorth   :1; // GESTURE_SOUTH_NORTH     0x00020000
        uint8_t GestNorthSouth   :1; // GESTURE_NORTH_SOUTH     0x00040000
        uint8_t EdgeGestWestEast :1; // GESTURE_EDGE_WEST_EAST   0x00080000
        uint8_t EdgeGestEastWest :1; // GESTURE_EDGE_EAST_WEST   0x00100000
        uint8_t EdgeGestSouthNorth :1; // GESTURE_EDGE_SOUTH_NORTH 0x00200000
        uint8_t EdgeGestNorthSouth :1; // GESTURE_EDGE_NORTH_SOUTH 0x00400000
        uint8_t GestClockWise    :1; // GESTURE_CLOCK_WISE       0x00800000
        uint8_t GestCounterClockWise :1; // GESTURE_COUNTER_CLOCK_WISE 0x01000000
        uint8_t FreeBit
    } Bit;
} GestureOutput;

```

struttura dati pubblica è distribuita su 32 bit, di cui 7 liberi per sviluppi futuri. Inoltre è possibile leggere e modificare i dati della struttura agendo su ogni singolo bit oppure più brutalmente a livello di byte o double word.

Nel **Listato 1** riportiamo la struttura dati nella sua interezza. Quindi per testare un singolo bit, ad esempio *TapWest*, si dovrà scrivere la seguente:

```
if (mgc3130.GestureOutput.Bit.TapWest != 0) { ... .. }
```

dove *mgc3130* è il riferimento alla libreria incluso nello sketch. Volendo, ma non è necessario, è possibile filtrare le gesture che non si vogliono utilizzare nel proprio progetto. Infatti è possibile configurare una costante di filtro, presente nel file MGC3130.h, che permette di filtrare le gesture

non volute. La costante si chiama *"MASK_FILTER_GESTURE"* ed è un valore a 32 bit configurabile bit a bit. Se il bit è a *"1"* logico significa che si desidera filtrare la gesture viceversa la si vuole mantenere. Nei nostri sketch di esempio non abbiamo fatto uso della costante di filtro, in altre parole l'abbiamo lasciata configurata a 0x00000000, questo vuol dire che nella funzione *"DecodeGesture"* è usata ma non ha alcuna influenza.

Per favorire lo studio della libreria, ci sono delle parti di codice nascoste che possono essere attivate al bisogno; queste sezioni sono racchiuse tra le direttive *"#ifdef identifier"* e *"endif"* dove *identifier* è una etichetta che, quando dichiarata, attiva la sezione di codice viceversa la esclude. Ad esempio la porzione di codice seguente:

```

#ifdef PRINT_RAW_DATA
    PrintMGC3130RawData();
#endif

```

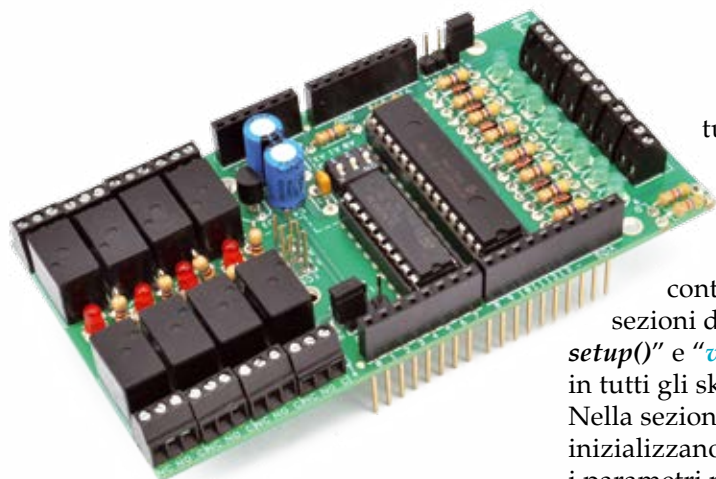
attiva/disattiva il codice della funzione *"PrintMGC3130RawData"*. Tale funzione stampa sul monitor seriale i dati letti dall'integrato prima che vengano decodificati dalla funzione *"DecodeGesture"*. Durante lo studio della libreria, e dell'integrato MGC3130, può risultare molto utile perché mostra i dati esattamente come escono dal buffer di MGC3130. Ad esempio la gesture *"Flick East to West"* è così evidenziata:

```

#####
Row data from MGC3130
Header: 1A081191
Payload: 1F01 | 86 | 80 | 0073 | 03100000
| 00000000 | 0000 | 000000000000
#####

```

Per aumentarne la leggibilità



e interpretazione il pacchetto dati viene diviso tra **"Header"** e **"Payload"**, in più il **"Payload"** viene suddiviso in sotto-pacchetti come evidenziato nel data-sheet della libreria Gestic.

Oltre alla sezione di codice nascosta appena trattata, ce ne sono altre tre, tra cui le gesture riconosciute e decodificate e le coordinate X, Y e Z (rispettivamente **PRINT_GESTURE_DATA** e **PRINT_XYZ**).

Se non attivate queste funzioni, esse non occupano spazio nella Flash.

MGC3130_DEMO

Descriviamo ora il nostro primo sketch, nel quale mostriamo l'utilizzo della nostra libreria e l'interfacciamento delle gesture selezionate rispetto alle uscite a relé messe a disposizione delle schede FT1079K.

Lo sketch è suddiviso in quattro file, di cui quello principale si chiama **"MGC3130_Demo"**.

Gli altri file permettono la gestione degli ingressi digitali messi a disposizione dal nostro elettrodo (**"DigitalInput"**), la gestione delle uscite a relé e del riporto delle gesture sulle uscite (**"DigitalOutput"**), e un ultimo file per la gestione del TIMER1 del microcontrollore Atmel (**"TimersInt"**) utile per gestire

tutte le costanti di tempo utilizzate nello sketch.

Il file **"MGC3130_Demo"** contiene le due classiche sezioni di codice **"void setup()"** e **"void loop()"** presenti in tutti gli sketch Arduino.

Nella sezione **"void setup()"** si inizializzano e configurano tutti i parametri necessari al corretto funzionamento dello sketch,

nonché le schede di espansione FT1079K con indirizzo hardware 0x00 e 0x01, nonché l'integrato MGC3130 (Indirizzo 0x42). Vengono inizializzati i pin di ingresso, richiamando la funzione **"void SetInputPin(void)"**, necessari a gestire i tre pulsanti P1, P2 e P3 e il jumper J1.

Vengono configurati i pin di uscita collegati alla scheda Arduino, in questo caso il solo led LD7 e infine viene configurato il TIMER1 e il relativo vettore di interrupt per la gestione delle costanti di tempo.

Per ultimo, ma non meno importanti, vengono configurate le macchine a stati utilizzate nello sketch. La sezione **"void loop()"** presenta i

richiami a tutte le macchine a stati precedentemente inizializzate nonché il codice per gestire l'integrato MGC3130. In particolare il codice di nostro interesse è il seguente:

```
if (mgc3130.GetTsLineStatus(Ts_MGC3130) == 0) {
    mgc3130.GetEvent(); // Start read data from MGC3130
    mgc3130.DecodeGesture(); // Decode Gesture
    mgc3130.ReleaseTsLine(Ts_MGC3130); // Release TS Line
}
```

Utilizzando la funzione di libreria:

```
boolean GetTsLineStatus( uint8_t Ts)
```

si testa la linea TS in attesa che vada a livello logico basso per poi impegnarla e iniziare la procedura di lettura dei dati presenti nel buffer di MGC3130 tramite la funzione di libreria:

```
void GetEvent(void)
```

Viene poi richiamata la funzione per la decodifica dei dati letti per estrapolare le gesture riconosciute e infine si rilascia la linea TS:

```
void DecodeGesture(void)
void ReleaseTsLine(uint8_t Ts)
```

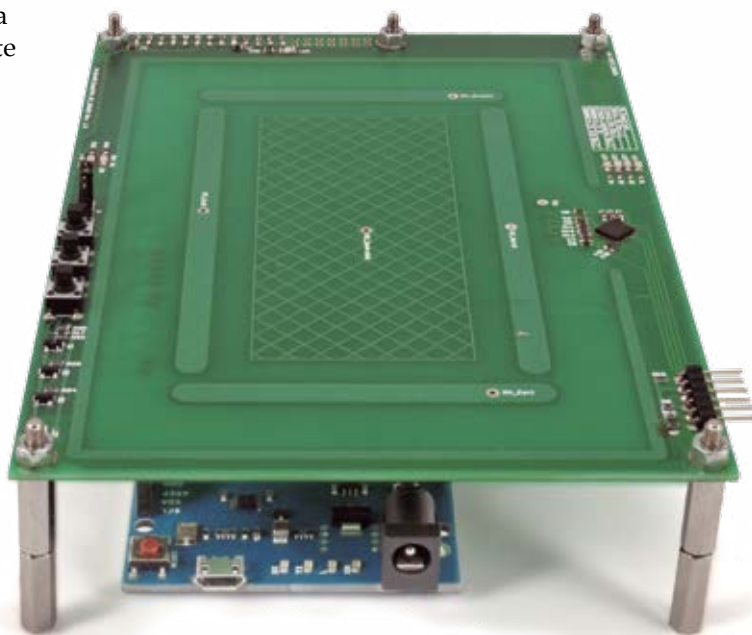


Tabella 1

P1	P2	P3	Tempo selezionato
ON	OFF	OFF	Tempo monostabile 1 secondo. Premere pulsante P1 per più di due secondi
OFF	ON	OFF	Tempo monostabile 5 secondi. Premere pulsante P2 per più di due secondi
OFF	OFF	ON	Tempo monostabile 10 secondi. Premere pulsante P3 per più di due secondi
ON	ON	OFF	Tempo monostabile 15 secondi. Premere pulsanti P1 & P2 per più di due secondi
OFF	ON	ON	Tempo monostabile 20 secondi. Premere pulsanti P2 & P3 per più di due secondi
ON	OFF	ON	Tempo monostabile 25 secondi. Premere pulsanti P1 & P3 per più di due secondi
ON	ON	ON	Tempo monostabile 30 secondi. Premere pulsanti P1 & P2 & P3 per più di due secondi

Tabella 2

Uscita	Indirizzo FT1079K	Gesture
01	0x00	Tap su elettrodo Sud
02	0x00	Tap su elettrodo Ovest
03	0x00	Tap su elettrodo Nord
04	0x00	Tap su elettrodo Est
05	0x00	Tap elettrodo Centrale
06	0x00	Doppio Tap su elettrodo Sud
07	0x00	Doppio Tap su elettrodo Ovest
08	0x00	Doppio Tap su elettrodo Nord
01	0x01	Doppio Tap su elettrodo Est
02	0x01	Doppio Tap elettrodo Centrale
03	0x01	Gesture movimento Ovest verso Est
04	0x01	Gesture movimento Est verso Ovest
05	0x01	Gesture movimento Sud verso Nord
06	0x01	Gesture movimento Nord verso Sud
07	0x01	Gesture movimento circolare orario
08	0x01	Gesture movimento circolare antiorario

Il jumper J1 collegato all'ingresso 3 (PD3) serve per decidere se le uscite digitali si devono comportare come monostabili o bistabili. Se il jumper è inserito, e quindi l'ingresso è a livello logico basso, le uscite devono funzionare in modo monostabile, viceversa come bistabili. Nella modalità monostabile è possibile decidere il tempo di eccitazione dell'uscita sfruttando i tre pulsanti P1, P2 e P3 (pulsanti collegati rispettivamente agli ingressi 4, 5, e 6 ovvero PD4, PD5 e PD6). Tenendo premuto il pulsante P1 per più di due secondi, verrà selezionato come tempo per il monostabile 1 secondo, se viene premuto P2 per più di due secondi verrà selezionato

un tempo di 5 secondi ecc. ecc. Riportiamo nella **Tabella 1** le possibili combinazioni con i pulsanti P1, P2 e P3 e i rispettivi tempi selezionabili (ON → Premuto; OFF → Non Premuto). Il tempo che viene selezionato è globale e quindi viene associato a tutte le uscite digitali pilotate come monostabili. In altre parole una volta selezionato, ad esempio, un tempo di 5 secondi questo sarà associato a tutte le uscite monostabili. Non è possibile associare tempi diversi per ogni uscita. Notate che la condizione bistabile significa che l'uscita inverte la propria condizione ad ogni riconoscimento della gesture associata; quindi un rilevamento della gesture eccita l'uscita, un

successivo rilevamento la porta a riposo e via di seguito. La modalità monostabile, invece, eccita l'uscita associata alla gesture e la mantiene per il tempo selezionato (vedere **Tabella 1**). Se viene intercettata nuovamente la gesture associata all'uscita prima che il tempo sia scaduto, si avrà un riarmo della stessa prolungando l'eccitamento dell'uscita associata.

Come accennato durante la descrizione della libreria, le gesture riconosciute e messe a disposizione dell'utente sono in totale 25; di queste, solo una parte è riportata sulle uscite a relé gestite dalle schede FT1079K. Nella **Tabella 2** riportiamo la corrispondenza tra la gesture riconosciuta e l'uscita a relé indirizzata sulla scheda FT1079K. Volendo, è possibile riportare sulle uscite a relé anche le gesture mancanti, ma per fare ciò è necessario aggiungere almeno un'altra scheda FT1079K con indirizzo 0x02; fatto ciò, bisognerà chiaramente scrivere il codice necessario alla gestione della nuova scheda.

MGC3130_LEONARDO

Descriviamo ora la demo realizzata con la scheda Arduino Leonardo Rev. 3. Lo scopo di questo sketch è di comandare un software di gestione delle immagini attraverso le gesture riconosciute dall'integrato MGC3130. Prima di tutto parliamo del programma utilizzato per la demo in esame: si tratta del programma "*FastStone Image Viewer*" scaricabile gratuitamente dal link web www.faststone.org/FSViewerDownload.htm. Dopo avere installato il software sul vostro PC, procedete con il configurare il suo possibile avvio tramite una combinazione di

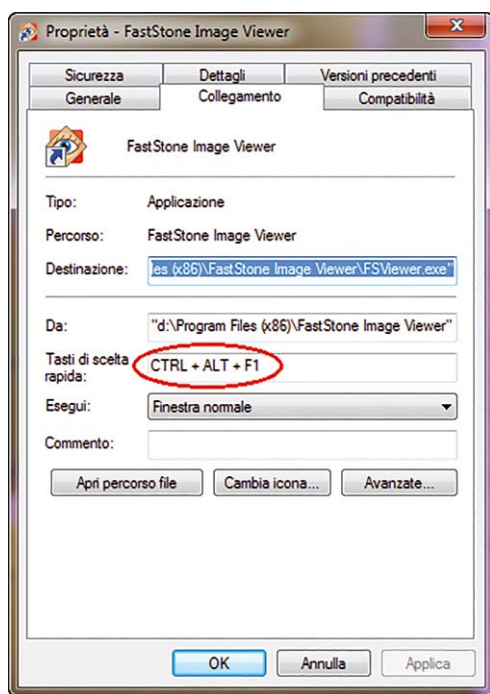


Fig. 8

tasti; per fare ciò bisogna cliccare con il pulsante destro del mouse sull'icona del programma e sotto la voce "*Collegamento*" impostare i tasti di scelta rapida. Nel nostro caso abbiamo impostato la combinazione "*CTRL+ALT+F1*" come evidenziato dalla Fig. 8. Con questo accorgimento possiamo impostare una gesture per l'avvio del software e altrettante gesture per la gestione delle immagini. Il meccanismo di interazione con il software su PC tramite gesture deve essere attivato; per fare ciò abbiamo predisposto due possibili azioni: la prima consiste nell'utilizzare i pulsanti P1 e P2 presenti sulla scheda elettrodo. Il pulsante P1, se premuto per più di due secondi, attiva il meccanismo di gestione del software su PC; se invece si preme il pulsante P2 per più di due secondi, il meccanismo viene disattivato. L'altro metodo consiste nell'utilizzare il monitor seriale,

per attivare il meccanismo inviare il carattere ASCII 'S' viceversa 'P' per disattivarlo. L'attivazione tramite pulsante P1 o carattere ASCII 'S' richiama le funzioni di attivazione dei servizi keyboard e mouse messi a disposizione dalla scheda Leonardo:

```
Keyboard.begin();
Mouse.begin();
```

Viceversa il pulsante P2 o il carattere ASCII 'P' richiama le funzioni di disattivazione dei servizi appena menzionati:

```
Keyboard.end();
Mouse.end();
```

L'attivazione/disattivazione tramite caratteri ASCII viene utile nel caso in cui la scheda Arduino Leonardo Rev.3 venga collegata a delle schede elettrodo diverse da quelle utilizzate per lo sketchup. Ad esempio gli elettrodi presentati nelle precedenti puntate o altro. Eseguita questa manovra preliminare, il sistema è pronto per riconoscere le gesture e di conseguenza di interagire con il software di gestione delle immagini.

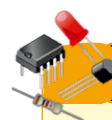
Per aprire il programma è sufficiente un Tap sull'elettrodo centrale: il programma viene aperto e saranno mostrate le immagini a disposizione. Per sfogliare le immagini utilizzare le gesture "*Ovest verso Est*" e "*Est verso Ovest*": la prima sfoglia le immagini verso destra e la seconda verso sinistra. Per vedere le immagini a tutto schermo è sufficiente eseguire un Tap sull'elettrodo "*Ovest*" e per uscire dalla visualizzazione a schermo intero è sufficiente un Tap sull'elettrodo "*Est*". Per ingrandire una immagine (Zoom +) eseguire una gesture "*Edge Ovest verso Est*" e per

ridurre l'immagine (Zoom -) eseguire una gesture "*Edge Est verso Ovest*". Quando si guarda una immagine a schermo intero zoomata è possibile muovere l'immagine ingrandita con le gesture "*Ovest verso Est*", "*Est verso Ovest*", "*Nord verso Sud*" e "*Sud verso Nord*".

Infine è possibile ruotare le immagini selezionate, per fare ciò eseguire una movimento circolare in senso orario per ruotare l'immagine verso destra e un movimento in senso antiorario per ruotare l'immagine verso sinistra. Per chiudere il software è sufficiente un doppio Tap sull'elettrodo centrale.

CONCLUSIONI

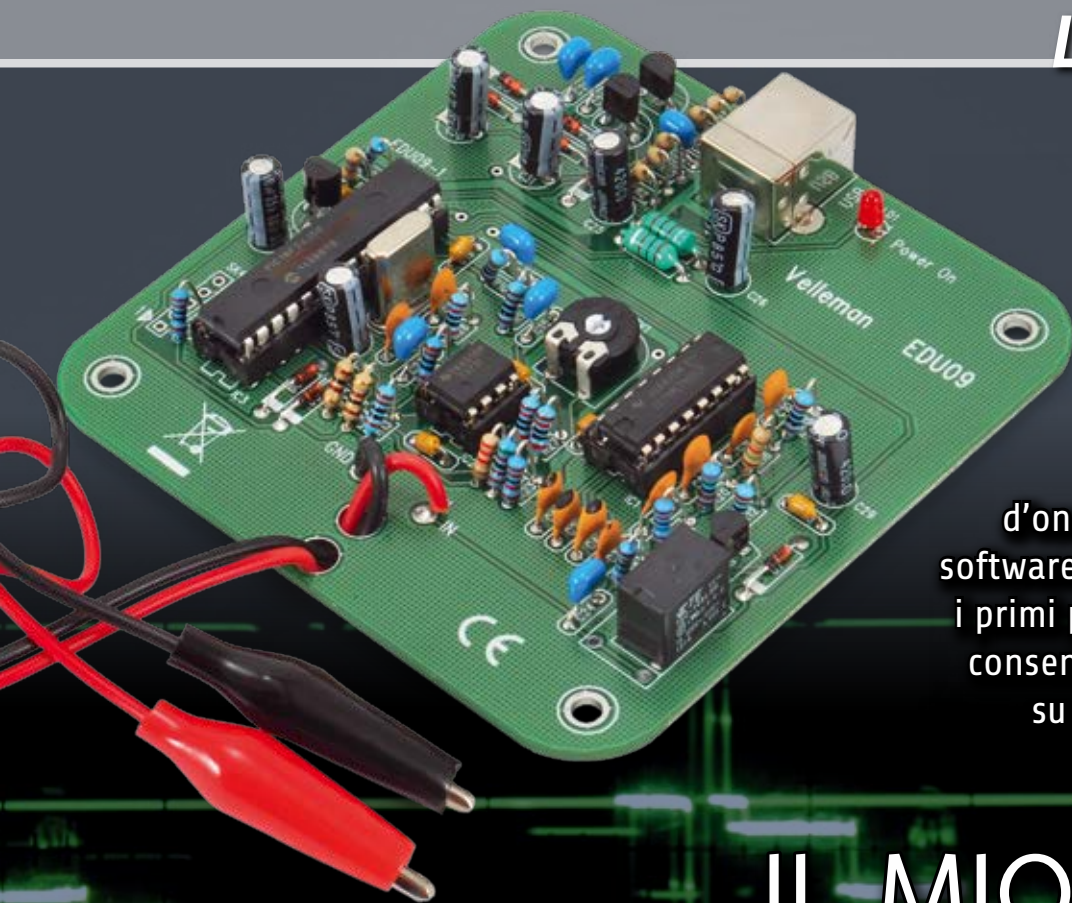
In questa terza puntata dedicata alla tecnologia GestIC abbiamo imparato a utilizzare il nuovo elettrodo con Arduino Uno Rev. 3 e Arduino Leonardo Rev. 3, vi abbiamo dato una serie di spunti ed esempi per l'utilizzo delle nostre librerie, fornendovi i mezzi per implementare nuove e interessanti funzioni da utilizzare nei vostri progetti. Nel prossimo articolo vedremo l'applicazione con Raspberry Pi.



per il MATERIALE

Il software Aurea di Microchip può essere scaricato dal sito www.microchip.com/gestic. Il sensore di prossimità (cod. MGC3130) è reperibile presso Futura Elettronica a 9,80 Euro.

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



Digitalizzatore di segnali da interfacciare al computer per vedere, memorizzare e analizzare le forme d'onda tramite un apposito software. Ideato per chi muove i primi passi in questo mondo, consente di effettuare misure su circuiti audio e digitali a bassa frequenza.

IL MIO PRIMO OSCILLOSCOPIO DIGITALE

di BORIS LANDONI

Il moderno oscilloscopio è decisamente differente da quello a CRT che abbiamo usato nei nostri banchi di laboratorio fino a una decina di anni fa; infatti il prezioso strumento per visualizzare le forme d'onda ha subito varie trasformazioni, la più rilevante delle quali è indubbiamente quella che ha visto il passaggio dall'analogico al digitale; ciò ha permesso di implementare potenti funzionalità come l'elaborazione del segnale e l'analisi delle forme d'onda. Oggi l'oscilloscopio è digitale e, nella versione stand-alone, dispone di un visualizzatore LCD; ma vanno prendendo piede anche gli strumenti da abbinare al computer, per i quali il termine "oscilloscopio" è quasi improprio: si tratta infatti di digitalizzatori di segnale,

che trasformano in dati numerici le grandezze campionate dai propri ingressi, inviando i dati corrispondenti a un software che gira su computer e che esegue, oltre alla visualizzazione in tempo reale, tutta una serie di operazioni consentite dalla grande capacità di memorizzazione dei Personal Computer. Gli oscilloscopi digitali oggi includono convertitori analogico/digitali (ADC) ad alta velocità di campionamento e di varia risoluzione (tipicamente 8 bit). Gli oscilloscopi basati su PC offrono il vantaggio di poter definire la funzionalità dello strumento da software; questo significa non solo poter compiere misure generali, ma anche, con opportune modifiche, misure specifiche. Inoltre tali strumenti possono funzionare come

CARATTERISTICHE TECNICHE

Lo strumento che vi proponiamo permette, in abbinamento all'apposito software, di svolgere le funzioni di oscilloscopio, analizzatore di spettro e registratore di transienti. Di seguito sono elencate le prestazioni nelle tre funzionalità.

OSCILLOSCOPIO

- banda passante: DC ÷ 200 kHz ± 3 dB
- impedenza d'ingresso: 100 kohm / 20 pF
- max. tensione d'ingresso: 30 V (AC + DC)
- base dei tempi: 10 µs ÷ 500 ms / divisione
- gamma d'ingresso: 100 mV ÷ 5 V / divisione
- sensibilità d'ingresso: 3 mV
- readout: True RMS, dBV, dBm, p / p, Duty cycle, frequenza
- buffer di registrazione: 1.000 campionamenti
- frequenza di campionamento: da 62,5 Hz a 1,5 MHz
- funzione storico campionamento
- funzione auto set-up
- funzione pre-trigger: gamma da 0,1 ms/div a 500 ms/div
- opzioni di persistenza: colore attenuato, variabile e infinito

ANALIZZATORE DI SPETTRO

- gamma di frequenza: da 0÷150 Hz a 75 kHz
- principio di funzionamento: FFT (Fast Fourier Transform)
- risoluzione FFT: 512 linee

REGISTRATORE DI TRANSIENTI

- base dei tempi: da 20 ms/div a 2.000 s/div
- tempo di registrazione max: 9,4 h/schermata
- salvataggio automatico dei dati
- registrazione e visualizzazione di schermate
- registrazione automatica superiore a 1 anno
- numero massimo di campionamenti al secondo: 100
- numero minimo di campionamenti al secondo: 1/20

analizzatore di spettro, frequenzimetro, ecc. Gli oscilloscopi del genere offrono diversi vantaggi rispetto a quelli tradizionali perché l'hardware è un digitalizzatore e la funzione di visualizzazione e analisi delle forme d'onda è implementata dal software che gira sul PC.

Essendo strumenti pratici ed economici rispetto a quelli standalone, abbiamo deciso di proporvi un progetto del genere: si tratta di un digitalizzatore da PC, connesso via USB, che abbinato all'apposito software permette di realizzare le funzioni dell'oscilloscopio digitale tradizionale ed altre addizionali.

Lo strumento non è certo professionale, ma permette a chi è alle prime armi o non deve misurare

frequenze troppo elevate, di compiere le sue prime misure.

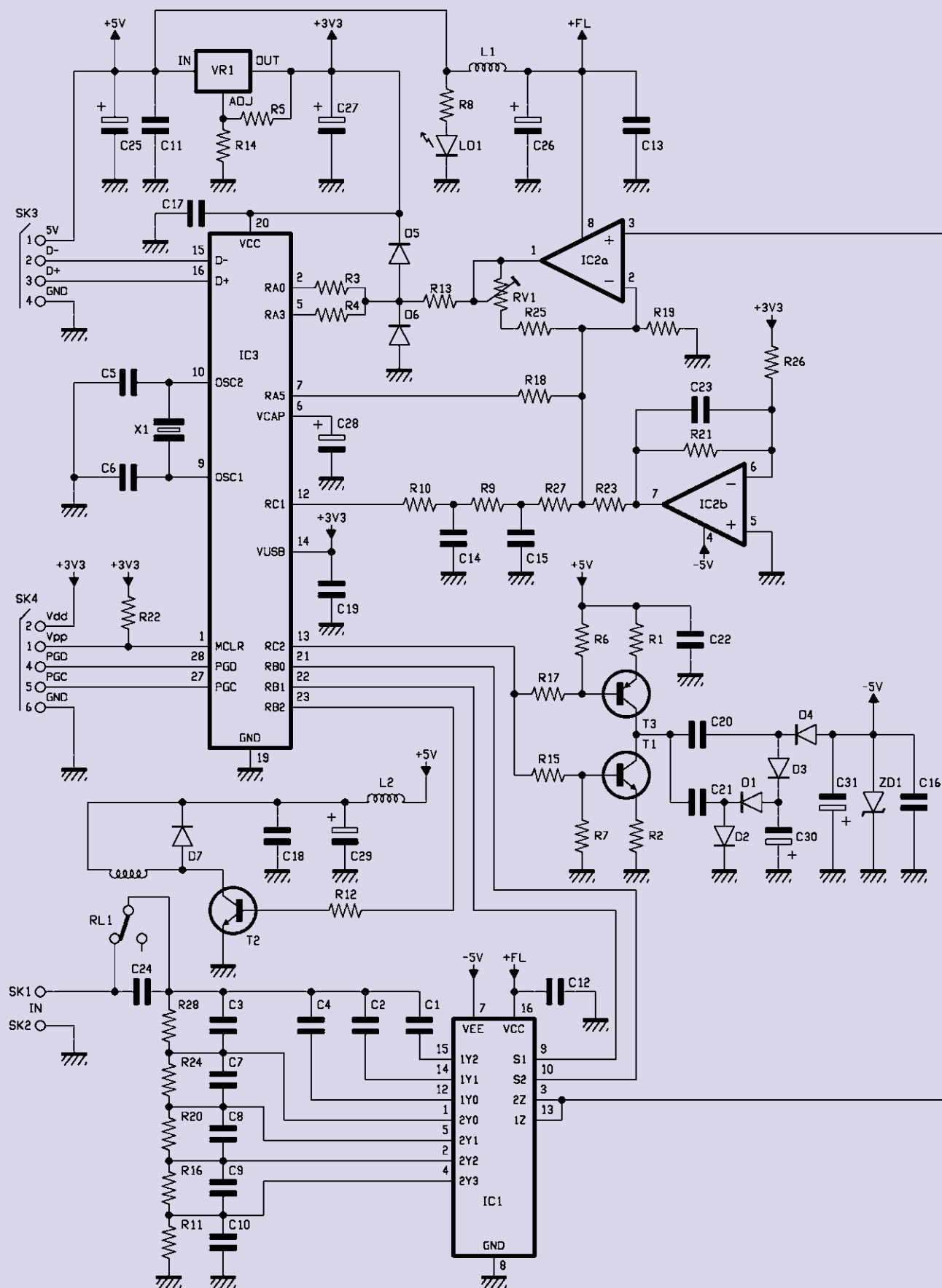
SCHEMA ELETTRICO

Il circuito è gestito da un potente microcontrollore Microchip PIC18F24J50, basato su architettura ad 8 bit e provvisto di interfaccia USB 2.0 Full Speed di tipo Device, che consente di interfacciarlo direttamente alla porta USB di qualsiasi computer; il micro integra una memoria di programma Flash da 16 kB, una RAM da 3.800 byte, 2 timer a 8 bit e 3 a 16 bit, due moduli Capture/Compare/PWM, un A/D converter da 10 bit assegnabile a 10 canali. Sul piano della connettività, oltre all'USB il microcontrollore implementa 2 UART, 2 A/E/USART, 2 link SPI e 2 bus

I²C. Il PIC18F24J50 funziona a tensione di valore compreso tra 2 e 3,6 volt ma i suoi input digitali sono 5V-tolerant, quindi possono essere pilotati con segnali TTL tradizionali; il clock è ricavato dall'oscillatore interno con l'aiuto del quarzo X1 (4 MHz) e dei condensatori C5 e C6.

L'impiego del PIC18F24J50 consente di ottenere dal digitalizzatore una frequenza di campionamento di 1,5 Ms/s, corrispondenti a 1,5 MHz.

Nel circuito, il microcontrollore acquisisce i segnali campionati dal multiplexer analogico a due canali 74HCT4052 (siglato IC1 nello schema elettrico), che nel nostro caso permette di selezionare in sequenza 6 portate attraverso il controllo degli ingressi di scansione con due segnali di clock che fornisce sui propri piedini RB0 ed RB1: ad ogni impulso, l'uscita di ciascun canale del multiplexer presenta la tensione letta dal successivo ingresso. Per il canale 1 gli input sono 1Y0, 1Y1, 1Y2 (1Y3 non è usato) quindi inizialmente sul piedino 1Z si trova il segnale prelevato da 1Y0; al primo impulso di clock su S1 passa il segnale presente su 1Y1, al secondo quello su 1Y2 ecc. Il discorso per l'altro canale è simile: all'inizio passa su 2Z il segnale presente su 2Y0, al primo impulso su S2 passa il segnale su 2Y1, al secondo quello su 2Y2 ecc. Le uscite dei due canali sono unite, cosicché di fatto il doppio multiplexer diventa unico e permette di prelevare 7 livelli di segnale ottenuti dal partitore a scala di resistenze R11/R16/R20/R24/R28 (ognuna ha in parallelo un condensatore di filtro per spegnere impulsi spurii che possono essere generati dal passaggio tra una linea e l'altra) e dai condensatori C1, C2, C4, i quali, insieme alla resistenza



interna degli switch CMOS contenuti nel 74HCT4052, formato da celle passa-alto che filtrano i segnali e impostano una certa attenuazione. La linea RB2 del microcontrollore pilota la base del transistor NPN T2, il cui collettore alimenta la bobina del relé RL1, opportunamente

proteggendo dal diodo D7 che spegne le extratensioni inverse generate dalla stessa all'interdizione; lo scambio di RL1 permette di selezionare l'accoppiamento: se è chiuso (relé a riposo, ovvero RB2 del micro a zero logico) l'ingresso dello strumento (IN) è accoppiato in continua

(DC coupling) mentre quando il relé è eccitato (RB2 a livello logico alto) l'ingresso è accoppiato in alternata (AC coupling) attraverso il condensatore C24, che a riposo sarebbe cortocircuitato dallo scambio del relé. Notate, sulla linea dei 5V, il filtro L/C composto da L2 e C18/C29,

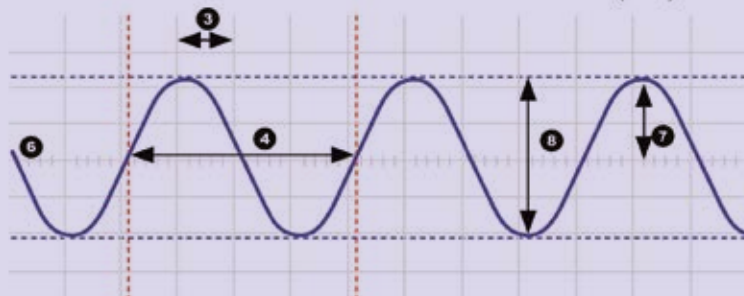
Il linguaggio dell'oscilloscopio

Negli oscilloscopi vengono adottate definizioni riguardanti le funzionalità e le visualizzazioni, che riportiamo di seguito.

- **Volt/Div:** definisce l'ampiezza (in volt) corrispondente a una divisione verticale.
- **Time/Div:** definisce il tempo corrispondente a una divisione orizzontale.
- **Division (1):** griglia visibile sullo schermo dell'oscilloscopio che aiuta a valutare l'ampiezza del segnale e il relativo periodo.
- **Period (2):** anche detto (T), è la durata di un ciclo della forma d'onda alternata ($= 1/f$).
- **Frequency (f):** numero di volte per cui il segnale si ripete in un secondo (cicli al secondo).
- **Trace (3):** è la linea che viene disegnata sullo schermo e che rappresenta il segnale d'ingresso.
- **Amplitude (4):** rappresenta l'ampiezza del segnale ed è espressa in mV o V (per i segnali ripetitivi è Vpeak).
- **Peak-to-peak (5):** differenza tra il valore della cresta positiva e quella negativa del segnale. Per i segnali sinusoidali equivale a $2 \times V_{peak}$.
- **AC coupling:** l'oscilloscopio visualizza solo componenti alternate (AC); qualsiasi componente DC (continua) applicata viene ignorata.

- **DC coupling:** l'oscilloscopio visualizza sia la componente alternata (AC) che quella continua (DC) del segnale.
- **Analog:** gli oscilloscopi analogici utilizzano il segnale in ingresso per deviare il fascio elettronico che effettua una "spazzolata" dello schermo da sinistra verso destra. L'immagine lasciata sullo schermo dal fascio elettronico rappresenta la forma d'onda del segnale applicato.
- **Digital:** gli oscilloscopi digitali eseguono una conversione analogica/digitale del segnale in ingresso ed elaborano tutti i dati acquisiti per poter rappresentare digitalmente la forma d'onda sul display.
- **Auto-setup mode:** l'oscilloscopio seleziona automaticamente un valore per Volts/Div e Time/Div in modo che sul monitor venga visualizzato correttamente uno o più periodi del segnale.
- **Clipping:** è quando la cresta superiore o quella inferiore o entrambe le creste del segnale vengono tagliate (cut-off), ad esempio, perché il segnale non può oscillare oltre il valore della tensione di alimentazione.
- **Noise:** segnali elettrici indesiderati che si sommano al segnale misurato.
- **Ripple:** variazione periodica indesiderata di una tensione continua.

- **Signal:** segnale (tensione) applicato all'ingresso dell'oscilloscopio; l'oggetto della misurazione.
- **Sine wave:** funzione matematica che rappresenta un'oscillazione ripetitiva nel tempo. La forma d'onda mostrata all'inizio di questo glossario è un'onda sinusoidale.
- **Spikes:** transienti di breve durata in un segnale.
- **Bandwidth:** (larghezza di banda) generalmente espressa in MHz. È la frequenza alla quale un'onda sinusoidale applicata sarà visualizzata con un'ampiezza pari al 70,7% della sua ampiezza originale. Gli oscilloscopi più costosi sono caratterizzati da una maggiore larghezza di banda. In linea di massima la larghezza di banda di un oscilloscopio deve essere almeno 2 volte superiore alla frequenza del segnale applicato al suo ingresso.
- **DC reference:** la misurazione DC (grandezze in continua) viene sempre effettuata rispetto al potenziale di massa, quindi è necessario definire questo livello. Se non si imposta il corretto riferimento DC, la lettura potrebbe non essere corretta. Nella maggior parte dei casi, questo livello viene fatto coincidere col centro dello schermo, ma ciò non è obbligatorio.
- **Accoppiamento d'ingresso:** sono possibili tre impostazioni: accoppiamento AC, accoppiamento DC e GND. Con accoppiamento AC, viene applicato in serie all'ingresso un condensatore. Questo blocca la componente continua del segnale lasciando passare solamente quella alternata. Con accoppiamento DC, il condensatore viene bypassato e sia la componente alternata che



la cui funzione è sopprimere i pur deboli disturbi generati dall'assorbimento di corrente o dalla sua assenza, rispettivamente quando il relé viene eccitato o viene rimesso a riposo; ciò per limitare il più possibile le interferenze che potrebbero affliggere la misura dei segnali più deboli.

quella continua del segnale vengono lasciate passare. I segnali a bassa frequenza (<20Hz) devono sempre essere visualizzati con accoppiamento DC. Con l'accoppiamento AC, il condensatore interno interferirebbe con il segnale causando una visualizzazione errata.

- **Sample rate:** generalmente espressa in campioni (s/s) o Mega campioni/secondo (Ms/s) e a volte in MHz. È la frequenza alla quale un oscilloscopio digitale acquisisce i campioni del segnale. Più elevata è la frequenza di campionamento, più l'immagine visualizzata sullo schermo è fedele alla forma d'onda applicata. Teoricamente la frequenza di campionamento deve essere il doppio della massima frequenza del segnale applicato; tuttavia, per ottenere migliori risultati con segnali non puramente sinusoidali, è preferibile che la frequenza di campionamento sia almeno 5 volte quella massima.
- **Sensitivity:** indica l'ampiezza minima del segnale d'ingresso in grado di far deviare verso l'alto o verso il basso la traccia sullo schermo, ovvero la minima ampiezza rilevabile e visualizzabile sullo schermo. Generalmente è espressa in mV.
- **Slope:** determina se il punto di trigger si trova sul fronte ascendente o discendente di un segnale.
- **Vrms:** la tensione efficace di una sorgente AC rappresenta la tensione continua necessaria per generare in una resistenza la stessa quantità di calore che produrrebbe una sorgente AC. Per un segnale sinusoidale, $V_{rms} = V_{peak} / 1,4142$.

Le uscite dei due stadi del multiplexer analogico confluiscono nell'amplificatore facente capo a IC1a, un operazionale TLV272 a basso rumore che troviamo configurato in modo non-invertente ed accoppiato in continua; l'accoppiamento in DC è necessario in quanto l'amplificatore deve trattare anche i segnali in cc che si presentano al suo ingresso quando l'IN dell'oscilloscopio è accoppiato in DC.

Il guadagno in tensione (G) dello stadio facente capo a IC2a è determinato dalla rete di retroazione formata da RV1/R25 e da R19, secondo la relazione:

$$G = (RV1+R25+R19) / R19$$

Essendo RV1 un trimmer, il guadagno può essere regolato in fase di taratura per ottenere la massima dinamica senza cimarare il segnale.

L'ingresso invertente dell'IC2a è polarizzato con la tensione di riferimento fornita dal secondo operazionale contenuto in IC2, configurato come amplificatore invertente con guadagno pari a 0,5; IC2b riceve sull'ingresso invertente 3,3 volt, che divide per due, dato che il guadagno G in questa configurazione è pari a:

$$G = -R21/R26$$

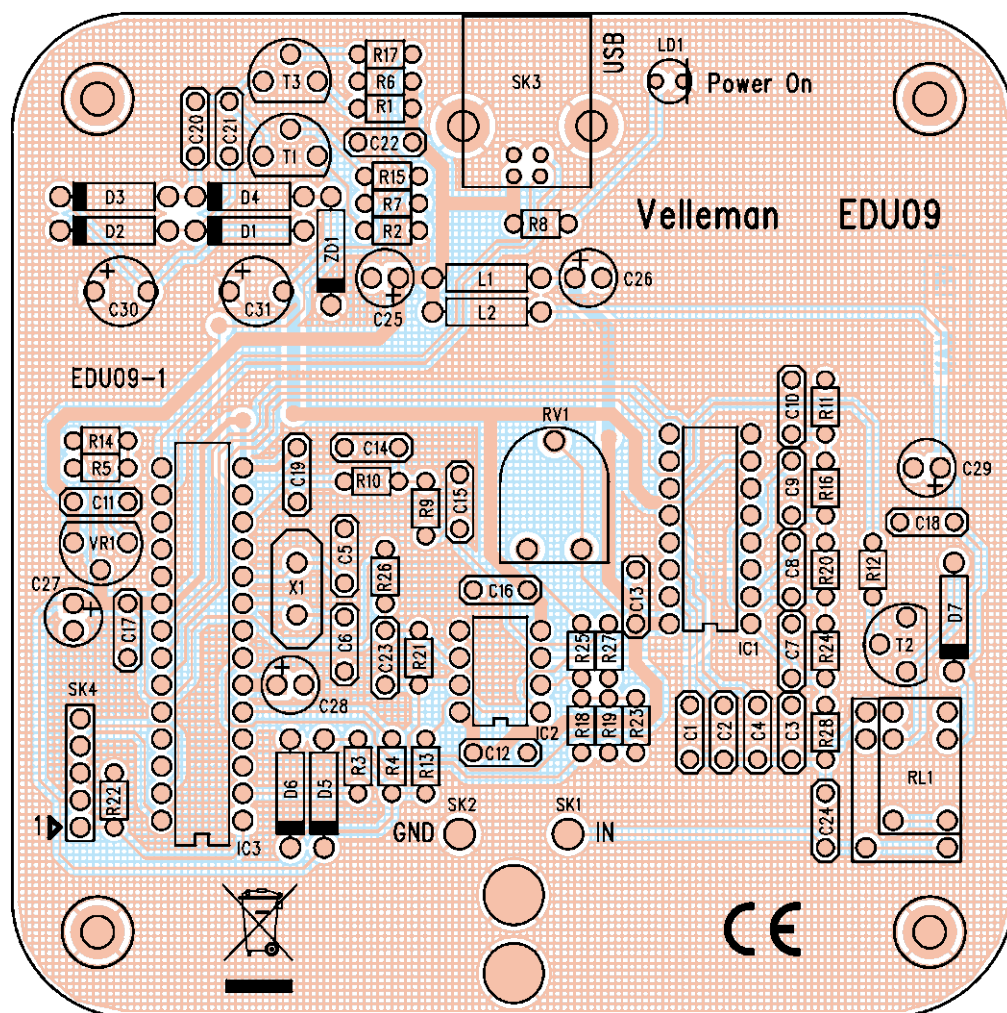
Siccome l'amplificatore è invertente, gli 1,65 volt sono negativi. Tale tensione viene ulteriormente ridotta dal partitore che R23 forma con R19, arrivando a -0,52 V, ed è applicata ad RA5 del microcontrollore, oltre che fornita come riferimento alla linea RC1 (i riferimenti servono per l'ADC) dopo essere stata opportunamente filtrata dalle due celle passabasso formate da R27/C15 e da R9/C14, il cui scopo è eliminare disturbi di vario genere pro-

venienti dall'alimentazione ed eventuale ripple. Come vedremo tra breve, infatti, il circuito preleva l'alimentazione principale dall'USB del computer e sappiamo che l'alimentazione dei PC non è proprio pulitissima, dato che proviene da alimentatori a commutazione.

L'uscita dell'IC2a fornisce all'ADC del microcontrollore (associato a due canali: RA0 ed RA3) il segnale da digitalizzare e inviare al computer sotto forma di dati via USB; per sopperire a eventuali ritardi di comunicazione, il micro permette di bufferizzare le misure fino a 1.000 campionamenti.

Procediamo con l'analisi dello schema e vediamo che siccome IC3 è alimentato a 3,3 volt e l'operazionale a 5V duali, per evitare che in caso di superamento, da parte della tensione di segnale riportata dal piedino 7 dell'IC2a sulle linee RA0 ed RA3 l'IC3 possa essere danneggiato, sono stati collocati i diodi al silicio D5 e D6, i quali bypassano RA0 ed RA3 nel caso la tensione d'uscita di IC2a superi il massimo rispettivamente positivo e negativo. Trattandosi di diodi Schottky, nel primo caso, su RA0 ed RA3 si registrerà una tensione di 0,2 V superiore a quella di alimentazione del microcontrollore, mentre nel secondo si troveranno -0,2V; sono valori più che tollerabili dagli I/O del micro.

Vediamo adesso la sezione di alimentazione, che consta di due stadi: uno indipendente basato sul regolatore VR1 ed uno governato dal microcontrollore. L'alimentazione principale viene prelevata dai contatti + (1) e - (4) del connettore USB SK3, la cui tensione di 5 volt viene filtrata dai condensatori C11 e C25, quindi applicata all'ingresso del regolatore lineare VR1 e al



Elenco Componenti:

R1, R2: 2,2 ohm
 R3, R4: 100 ohm
 R5÷R7: 680 ohm
 R8÷R13: 1 kohm
 R14: 1,1 kohm
 R15÷R16: 1,5 kohm
 R17: 2,2 kohm
 R18: 2,7 kohm
 R19: 5,1 kohm
 R20: 7,5 kohm
 R21÷R22: 10 kohm
 R23: 1 kohm
 R24: 15 kohm
 R25, R26: 20 kohm
 R27: 20 kohm
 R28: 75 kohm
 RV1: Trimmer 4,7 kohm MO
 C1: 2,2 pF ceramico
 C2: 6,8 pF ceramico
 C3: 10 pF ceramico
 C4: 15 pF ceramico
 C5, C6: 27 pF ceramico
 C7: 47 pF ceramico
 C8: 100 pF ceramico
 C9: 470 pF ceramico
 C10: 680 pF ceramico
 C11÷C13: 100 nF ceramico
 C14, C15: 1 µF 63 VL elettrolitico

piedino 8 del doppio operativo U2, tramite il filtro L1/C13/C26, che insieme a C11 e C25 forma una cella a pi-greca usata per sopprimere i disturbi in arrivo dall'USB del computer e

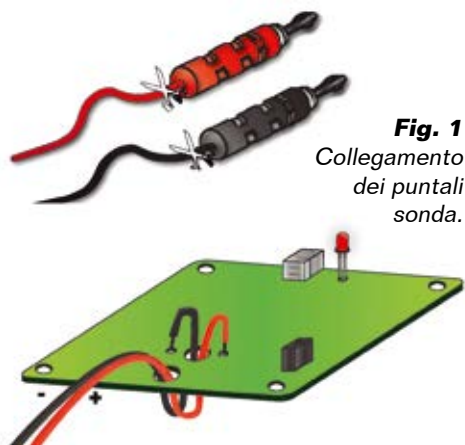


Fig. 1
Collegamento dei puntali sonda.

a non propagare a quest'ultimo eventuali interferenze presenti nella nostra scheda. Quando il circuito è collegato al PC e questo è acceso, o comunque fornisce alimentazione all'USB, il LED LD1 si illumina indicando che la scheda è sotto tensione. Il regolatore VR1 è un LM317 montato nella tipica configurazione, che eroga tra il contatto OUT e la massa una tensione stabilizzata il cui valore dipende dal rapporto tra le resistenze R14 ed R5: nel nostro caso, si tratta di 3,3 volt. La formula che fornisce la tensione d'uscita dell'integrato in funzione della resistenza di retroazione R5 e di quella di massa R14 è:

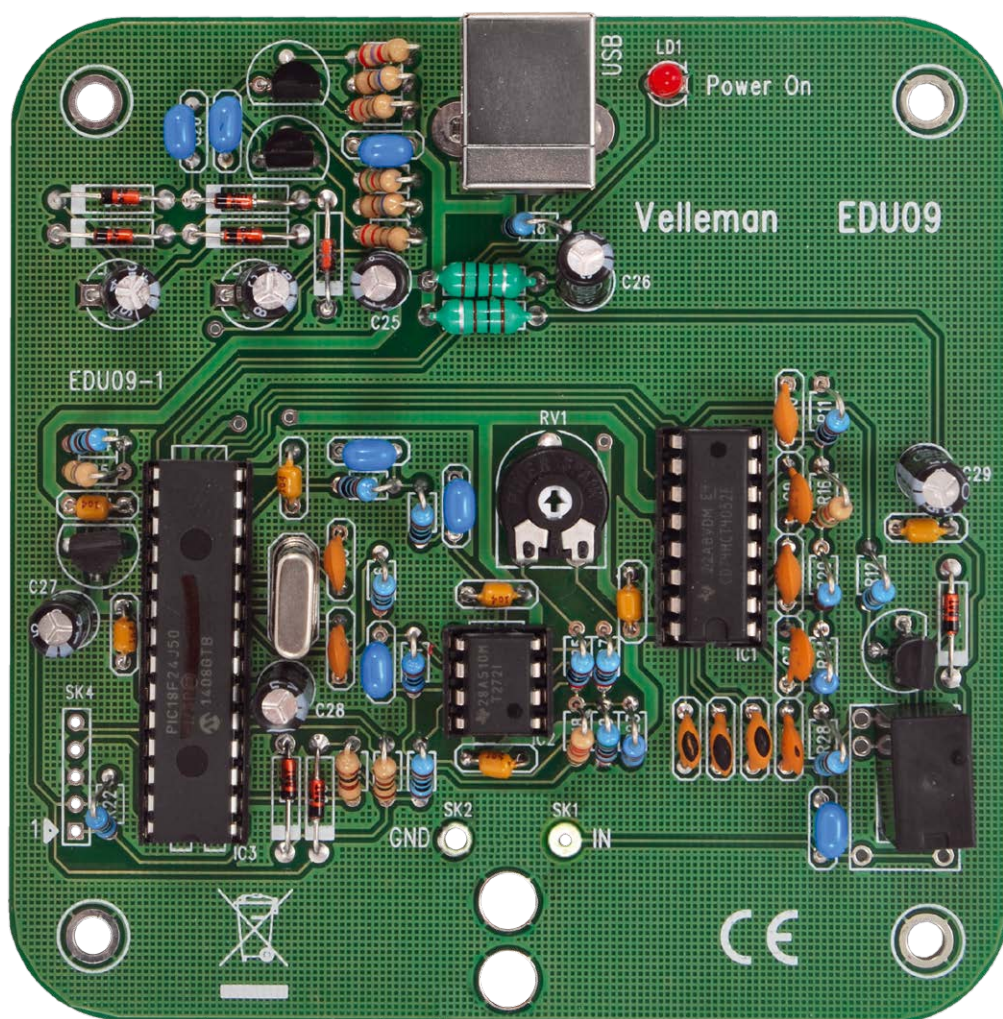
$$V_{out} = 1,25V (1 + R14/R5) + I_{adj} \times R14$$

dove 1,25V è il potenziale tra il piedino ADJ e la massa, mentre I_{adj} è la corrente che scorre da ADJ verso massa, che vale tipicamente 100 microampere. Essendo R5 pari a 680 ohm e R14 da 1,1 kohm, otteniamo 3,38 volt. L'alimentazione negativa per gli operazionali viene ottenuta mediante un invertitore di tensione costruito dall'hardware facente capo ai transistor T1 e T3, dai diodi D1, D2, D3, D4, DZ1, nonché dai condensatori ad essi collegati; il tutto viene pilotato dalla linea RC2 del microcontrollore, che per impostazione

C16÷C19: 100 nF ceramico
 C20÷C24: 1 μ F 63 VL
 elettrolitico
 C25÷C29: 10 μ F 63 VL
 elettrolitico
 C30: 100 μ F 16 VL
 elettrolitico
 C31: 100 μ F 16 VL
 elettrolitico
 D1÷D7: BAT85
 ZD1: Zener 5v1 400mW
 T1÷T3: BC327
 L1, L2: Induttanza 100 μ H
 X1: Quarzo 4 MHz
 VR1: LM317LZ
 RL1: Relè 5V (TSC-106D3H)
 IC1: CD47HCT4052E
 IC2: TLV272IP
 IC3: PIC18F24J501-I/SP
 (VKEDU09)
 LD1: LED 3 mm rosso

Varie:

- Connettore USB-B
- Zoccolo 4+4
- Zoccolo 8+8
- Zoccolo 14+14
- Circuito stampato PEDU09



del firmware, genera continuamente un segnale rettangolare di frequenza costante a livello 0/3,3 volt che determina la commutazione dei transistor dell'invertitore di tensione. Il funzionamento dello stadio è il seguente: quando RC2 si trova a livello logico basso, T1 è interdetto mentre T3 è in saturazione, cosicché con il proprio collettore alimenta i condensatori C20 e C21; il primo si carica attraverso D3, D2, D1 (gli ultimi due bypassano l'elettrolitico C30) e il secondo attraverso D2. I diodi sono tutti Schottky, quindi hanno una caduta quasi irrilevante (tipicamente 0,2V ciascuno) rispetto alla tensione di carica dei condensatori.

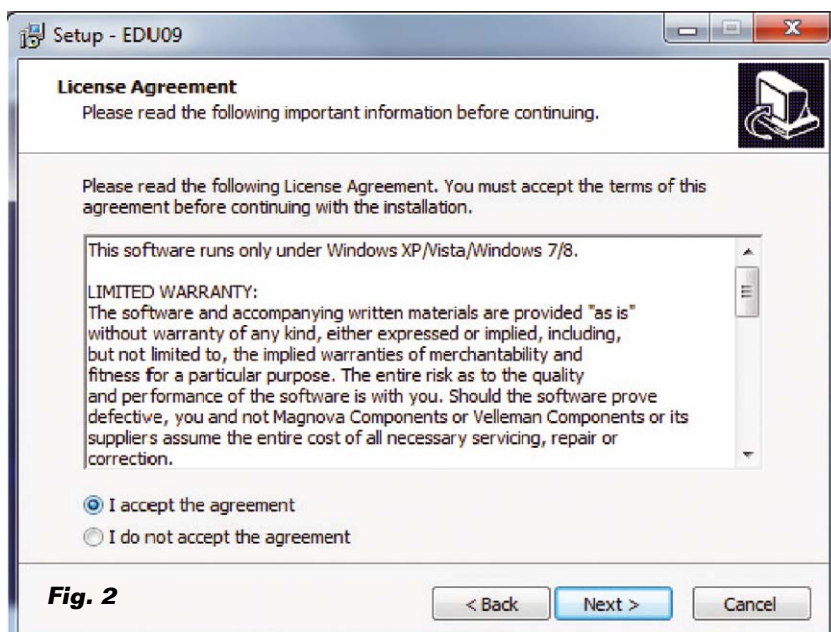


Fig. 2

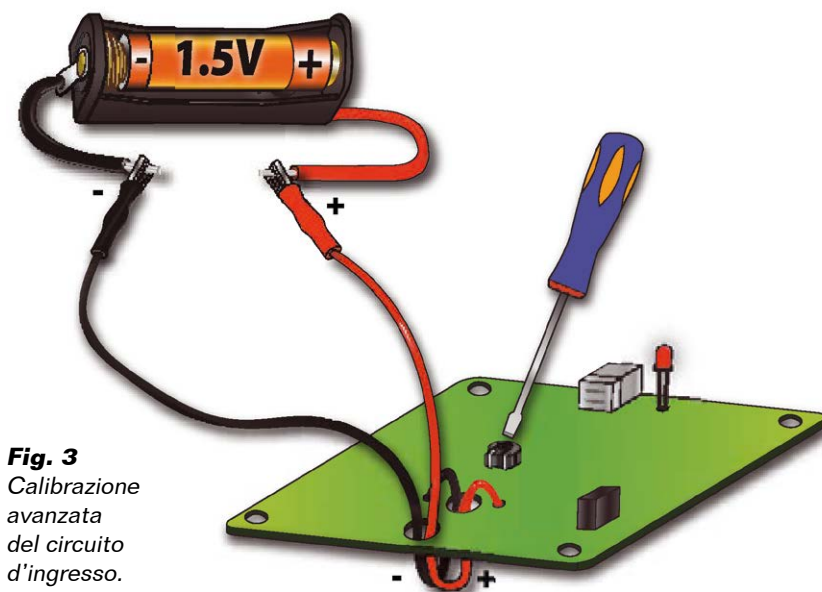


Fig. 3
Calibrazione
avanzata
del circuito
d'ingresso.

Quando il segnale rettangolare assume il livello logico alto, la situazione si inverte e T3 va in interdizione (grazie al resistore di pull-up R6, che ne mantiene la base a circa 5 volt) mentre T1 va in conduzione, scaricando C21 e C20: il primo fornisce -tramite D1 e T1- la propria energia a C30, che perciò si carica a 5 volt negativi rispetto a massa, mentre il secondo cede la propria carica -tramite D4- a C16 e C31, che si caricano a 5 volt negativi, eventualmente limitati mediante il diodo Zener ZD1. Nel successivo ciclo C30 cede la propria energia

a C31 e C16, caricando questi condensatori sempre a 5 volt.

REALIZZAZIONE PRATICA

Bene, questo è quanto, almeno per la descrizione dell'hardware; prima di passare al software da PC che permette di acquisire e visualizzare le misure, vediamo i dettagli costruttivi della scheda, che richiede per prima cosa la realizzazione del circuito stampato sul quale prenderanno posto tutti i componenti. Il c.s. è a doppia faccia e va preparato a partire dalle tracce lato

www.elettronica.in.it; dopo averlo inciso e forato, disponetevi le resistenze e gli zoccoli per gli integrati, quindi le induttanze, i diodi al silicio, il trimmer e i condensatori non polarizzati. Procedete con i transistor, il quarzo, gli elettrolitici, il connettore USB, il regolatore LM317 e il relé, quindi sistemate il LED. Se montate il kit, inserite i distanziali in plastica per la piastra di copertura e tenete LD1 ad un'altezza che permetta di farlo spuntare dall'apposito foro.

Per l'orientamento dei componenti polarizzati fate riferimento al disegno di montaggio che trovate due pagine indietro.

Il circuito va completato collegandogli i puntali, che possono essere tranquillamente quelli di un multimetro digitale (cavetti terminanti con coccodrilli), vista la banda passante relativamente bassa dello strumento; procurati i puntali, tagliate gli spinotti (banane) dai lati opposti dei cavetti e spelate i fili, quindi introduceteli nel foro grande più vicino al foro della basetta e fateli rientrare da quello attiguo, infilandoli -poi- ognuno nella rispettiva piazzola IN SK1 (il rosso va nella piazzola collegata a C24/RL1 e il nero a quella di massa). Per non sbagliare fate riferimento alla Fig. 1. In alternativa ai puntali potete collegare, con del cavetto coassiale, i contatti IN a un connettore BNC femmina da pannello (da fissare a una parete del contenitore del vostro strumento) come quelli degli oscilloscopi tradizionali, ricordando che la calza-schermo deve unire la massa del circuito e l'involucro del BNC, mentre il conduttore centrale deve collegare insieme il contatto centrale del connettore e l'altra piazzola IN.

Una volta realizzato il circuito e trovato l'eventuale conteni-

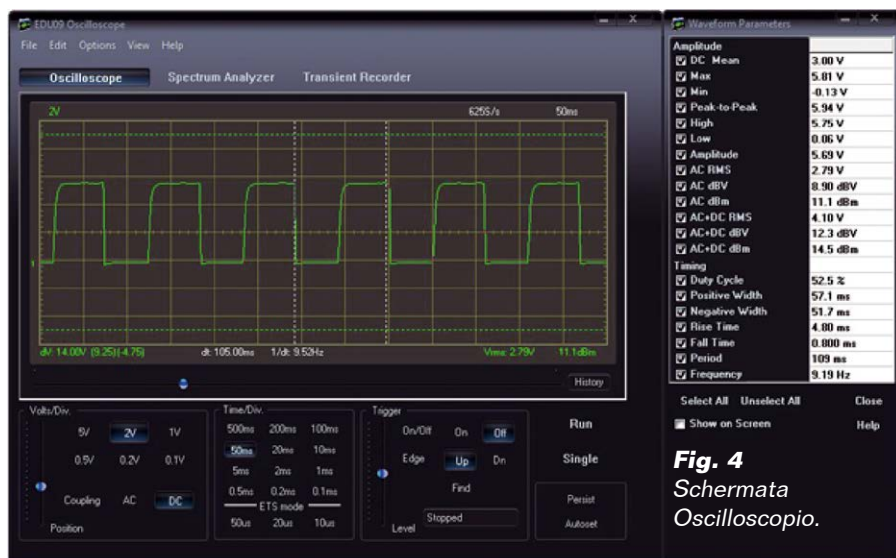


Fig. 4
Schermata
Oscilloscopio.

tore (questo vale se scegliete di non usare il kit previsto) potete provvedere all'installazione, collegandolo alla presa USB di un PC: infatti il digitalizzatore fa tutto con l'USB, dal quale trae l'alimentazione e invia i dati sulle acquisizioni.

Per gestire la scheda serve un apposito software (per ambiente Windows dall'XP in poi) scaricabile dalla pagina web www.vellemanprojects.eu/support/downloads/?code=EDU09. Una volta scaricato il relativo archivio compresso, lo si scompatta e si conferma l'avvio dell'installazione (Fig. 2). Al termine dell'installazione, l'installer propone la schermata corrispondente, nella quale dovete fare clic su Finish per completare la procedura e tornare al sistema operativo. La scheda dell'oscilloscopio non richiede l'installazione di alcun driver aggiuntivo, perché la periferica USB Microchip per impostazione predefinita viene riconosciuta come periferica HID; è anche possibile personalizzarla acquistando e programmando un driver specifico da Microchip, ma in questo caso serve a poco. Dopo l'installazione collegate il cavo USB alla scheda e al PC, quindi verificate che il LED rosso si accenda e attendete che avvenga il rilevamento e che la procedura corrispondente sia completata, quindi ruotate il cursore del trimmer RV1 nella posizione centrale. Adesso avviate il software EDU09.EXE precedentemente installato; se l'unità è stata collegata per la prima volta, la procedura di calibrazione si avvierà automaticamente. Se non dovesse avviarsi in automatico, nel menu Options selezionate "Calibrate" e poi cliccate su OK per avviare la calibrazione e attendete il completamento dell'operazione.



Fig. 5
Schermata analisi di spettro.



Fig. 6 - Schermata del registratore di transienti.

È anche possibile effettuare una calibrazione avanzata, vale a dire la messa a punto del circuito amplificatore d'ingresso; allo scopo è richiesta una batteria da 1,5V da collegare come mostrato nella Fig. 3. Questa calibrazione non è indispensabile; la si può eseguire quando si desidera ottenere maggior precisione nelle proprie misurazioni. Per effettuarla procedere così:

- selezionare "Expert Settings"

dal menu "Options";

- selezionare "Waveform Parameters..." dal menu "View";
- nella finestra "Waveform Parameters" spuntare "DC Mean";
- misurare la tensione della batteria con un multimetro e prenderne nota;
- collegare la batteria all'ingresso dell'oscilloscopio;
- impostare "0,5V" nella casella Volts/Div e quindi fare clic sul pulsante "Run";

Scegliere un oscilloscopio

Quello proposto in queste pagine è uno strumento basilare a scopo più che altro didattico; quando avrete preso la mano con l'oscilloscopio e vi servirà uno strumento professionale, vorrete orientarvi verso un prodotto più prestante, ma scegliere tra la vasta offerta commerciale non è semplice. Per aiutarvi nel compito, elenchiamo le principali caratteristiche da valutare.

LARGHEZZA DI BANDA

Definisce il campo di frequenze del segnale che può essere applicato al front-end analogico e che può essere campionato. La larghezza di banda viene definita come il campo di frequenze entro cui l'attenuazione non scende oltre i 3 dB, ovvero dove il segnale si mantiene tra il 100% e il 70,7%. La larghezza di banda è tipicamente definita da una frequenza inferiore ed una superiore, che sono, rispettivamente, la minima e la massima alla quale si verifica l'attenuazione di 3 dB. Di norma, per gli oscilloscopi la larghezza di banda corrisponde alla frequenza superiore, dato che tali strumenti partono dalla continua (DC). In linea generale la larghezza di banda dello strumento dev'essere almeno il doppio della frequenza del segnale da misurare. Siccome gli oscilloscopi digitali vengono usati anche per misurare il tempo di salita di segnali come gli impulsi oppure altri segnali con fronti ripidi, per acquisire la forma d'onda reale di segnali del genere o che non siano una pura sinusoidale, è necessario disporre di un'elevata larghezza di banda, in quanto i segnali diversi dal sinusoidale sono composti da più forme d'onda elementari. Ciò è dimostrato dall'Analisi di Fourier, secondo la quale un'onda quadra è composta da una sinusoidale di pari frequenza (detta "fondamentale") e da infinite armoniche, ognuna delle quali è una sinusoidale di frequenza multipla e ampiezza pari al rapporto tra l'ampiezza della quadra e l'ordine di armonica. Per esempio, un'onda quadra a 10 MHz è composta da una sinusoidale a 10 MHz, una seconda sinusoidale

(detta "seconda armonica") a 20 MHz con ampiezza pari a metà di quella della fondamentale, una terza di frequenza pari a 30 MHz e ampiezza pari a 1/3 e via di seguito. Quindi la forma d'onda visualizzata corrisponde a quella reale tanto più ampia è la larghezza di banda. Come regola empirica, bisogna utilizzare la seguente formula (dove RT è il tempo di salita) per calcolare la larghezza di banda (B) del segnale in base al suo tempo di salita (definito come il tempo necessario per la transizione dal 10 al 90 % di ampiezza del segnale):

$$RT = 0,35 / B$$

Teoricamente, per avere una rappresentazione reale degli impulsi bisognerebbe avere una larghezza di banda da 3 a 5 volte la frequenza del segnale. In altre parole, il digitalizzatore dovrebbe avere un tempo di salita da 1/5 a 1/3 di quello del segnale, almeno se si desidera effettuare un'acquisizione affetta dal minimo errore.

FREQUENZA DI CAMPIONAMENTO

Oltre alla larghezza di banda, nella scelta di un oscilloscopio o un digitalizzatore per oscilloscopio da PC bisogna considerare il sample rate, ossia la frequenza di campionamento dell'ADC (espressa in MS/s); infatti un'ampia larghezza di banda può non essere sufficiente se la frequenza di campionamento è troppo limitata. Mentre la larghezza di banda descrive la frequenza più alta dell'onda sinusoidale che può essere digitalizzata con attenuazione di -3dB, la frequenza di campionamento è la velocità con cui è scandita l'acquisizione di campioni da parte del convertitore analogico/digitale (ADC) dello strumento per digitalizzare il segnale in arrivo. Va ricordato che la frequenza di campionamento e la larghezza di banda non sono direttamente correlate, tuttavia c'è una regola per conoscere la relazione ottimale tra queste due caratteristiche: la frequenza di campionamento in tempo reale del digitalizzatore dev'essere da 3 a 4 volte la larghezza di banda del dispositivo.

MODALITÀ DI CAMPIONAMENTO

Ci sono due principali modalità di campionamento: quella in tempo reale (RTS=Real Time Sampling) e il campionamento in tempo equivalente (ETS=Equivalent Time Sampling). La frequenza di campionamento in tempo reale descrive la frequenza di clock dell'ADC e indica la velocità massima di un segnale in ingresso che può essere acquisito in un'acquisizione single-shot (a singolo impulso).

Invece il campionamento in tempo equivalente è un metodo di ricostruzione di un segnale basato su una serie di forme d'onda triggerate, ciascuna delle quali è acquisita in modalità single-shot. Il vantaggio dell'ETS è che offre un'elevata frequenza di campionamento effettiva; il rovescio della medaglia è che richiede più tempo ed è applicabile solo a segnali ripetitivi, quindi non a singoli impulsi. La ETS non aumenta la larghezza di banda analogica del digitalizzatore, ma è utile solo quando è necessario ricostruire il segnale a una frequenza di campionamento più elevata.

RISOLUZIONE E GAMMA DINAMICA

Questi due parametri dipendono dall'A/D converter: il numero di bit restituiti dalla conversione A/D rappresenta la risoluzione del digitalizzatore; per un dato campo di ampiezze del segnale d'ingresso, il numero di possibili livelli discreti utilizzati per rappresentare il segnale digitale è 2b, dove "b" è la risoluzione del digitalizzatore. L'intervallo di ampiezza del segnale in ingresso è diviso in 2b passi e la più piccola tensione rilevabile con il digitalizzatore corrisponde all'intervallo di input diviso per 2b. Ad esempio, un digitalizzatore a 8 bit divide un'ampiezza in ingresso da 10 Vpp in 28=256 livelli, da 39 mV ciascuno, mentre un digitalizzatore a 24 bit divide lo stesso campo di ingresso di 10 Vpp in 224=16.777.216 livelli da 596 nV (la risoluzione è 65.536 volte più che nel caso ad 8-bit).

Di norma è meglio usare un digitalizzatore ad alta risoluzione, in quanto permette

- regolare il trimmer RV1 finché il valore "DC Mean" visualizzato nella finestra "Waveform Parameters" non corrisponde al valore della tensione della pila misurato in precedenza.

Adesso l'oscilloscopio digitale è pronto all'utilizzo e potete fare con esso le vostre misure. Per prenderci un po' la mano suggeriamo di realizzare un circuito dimostrativo (Tutor Kit per

Oscilloscopio) che, alimentato a tensione alternata, fornisce ben 6 segnali campione con cui potete fare le prime prove di visualizzazione e misura. Di questo circuito parleremo in un apposito

di misurare con accuratezza segnali anche di piccolissima ampiezza; tale misura potrebbe anche essere effettuata con uno strumento a risoluzione più bassa, ma in tal caso l'ampiezza del segnale campionabile sarebbe ridotta, ovvero, si potrebbe campionare solo segnali a ridotta gamma dinamica. Se l'ADC presenta un'ampia gamma dinamica e bassa risoluzione, con lo strumento è possibile misurare segnali di grande ampiezza, ma il segnale debole sarebbe confuso con il rumore; di riflesso, limitando la gamma dinamica bisogna cedere i segnali di grande ampiezza e la misura sarebbe falsata e non valida. Quindi, per applicazioni che implicano segnali molto dinamici (con grande differenza tra la massima e la minima ampiezza), è necessario uno strumento ad alta risoluzione che abbia una vasta gamma dinamica.

TRIGGERING

Tipicamente, oscilloscopi e digitalizzatori acquisiscono il segnale al verificarsi di certi eventi; la capacità di implementare varie funzionalità di trigger permette di isolare gli eventi e catturare il segnale prima e dopo di essi.

Molti digitalizzatori includono modalità di trigger analogico, digitale, sui fronti (edge) e via software; altre opzioni di trigger includono finestre, isteresi e trigger video.

I digitalizzatori degli oscilloscopi professionali dispongono di circuiti di trigger capaci di agganciare in tempi brevissimi tempi (fast rearm trigger), il che permette una modalità di acquisizione multi-record, dove il digitalizzatore cattura il numero di punti specificato, ognuno su un dato trigger, e in modo rapido si riarma e attende il trigger successivo. Un riarmo veloce assicura che il digitalizzatore non si lasci sfuggire l'evento o il trigger. La modalità multi-record permette di catturare e memorizzare solo i dati necessari, ottimizzando così l'uso della memoria integrata, oltre a limitare l'attività del bus del PC.

MEMORIA ON-BOARD

Negli strumenti da PC, la velocità di campionamento è limitata da quella del bus di comunicazione utilizzato.

Se il bus di interfaccia non può sostenere un trasferimento continuo di dati alla frequenza di campionamento di acquisizione, la memoria integrata nello strumento memorizza i dati per un certo tempo, consentendo di acquisire i segnali alla massima velocità, poi li manda al PC per l'elaborazione.

Una memoria capiente aumenta il tempo a disposizione per l'acquisizione. Inoltre, disponendo di molta memoria nello strumento, è possibile creare un buffer che sopprimerà, per un certo tempo, all'incapacità di far arrivare al computer in tempo reale i dati del campionamento.

NUMERO DI CANALI

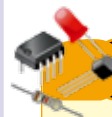
La maggior parte degli oscilloscopi ha da due a quattro canali, campionati simultaneamente ad una certa velocità. Ai fini della valutazione dello strumento è importante sapere come la frequenza di campionamento venga influenzata dal numero di canali utilizzati simultaneamente, in quanto in alcuni casi, quando si utilizzano tutti i canali del digitalizzatore, la frequenza di campionamento cala. Ciò, a causa di una tecnica comunemente usata e chiamata campionamento a tempo-interlacciato, che intercala più canali per ottenere una frequenza di campionamento più elevata. Se il digitalizzatore dell'oscilloscopio utilizza questo metodo e nella misura si impegnano tutti i canali, lo strumento non può essere in grado di acquisire alla velocità massima dichiarata con l'uso di un solo canale alla volta.

Il numero di canali richiesti dipende interamente dalla misura che bisogna effettuare; quando i tradizionali due o quattro canali non bastano, l'unica soluzione è orientarsi verso strumenti modulari espandibili al bisogno, ma si tratta di soluzioni molto costose e che di norma non vengono richieste nella gran parte degli ambiti.

Le schermate campione sono illustrate rispettivamente in **Fig. 4, Fig. 5, Fig. 6.**

Nell'articolo del generatore di 6 segnali vedremo anche le impostazioni dello strumento per effettuare la misura e la visualizzazione di segnali campione; per ora sappiate che l'impostazione predefinita (ovvero quella dopo l'installazione del software) prevede la visualizzazione della sola funzione oscilloscopio (nella finestra di lavoro è presente la sola tab "Oscilloscope") mentre aprendo il menu "Options" e cliccando su "Expert Settings", si visualizzano le tre funzioni e appaiono anche le tab Spectrum Analyzer e Transient Recorder. Per tornare alla visualizzazione predefinita riaprire il menu Options e spuntare Basic Settings. Il pannello dell'oscilloscopio riporta i comandi canonici, ossia base dei tempi (Time/Div), ampiezza (Volts/Div) e modalità di trigger. L'acquisizione si avvia con il pulsante Run.

I primi due comandi del menu File permettono di aprire acquisizioni salvate in precedenza. ■



per il MATERIALE

Questo progetto viene prodotto in scatola di montaggio dalla società Velleman e distribuito in Italia da Futura Elettronica. Il kit è completo di tutti i componenti, minuterie comprese. Il microcontrollore è già programmato, la basetta è forata e serigrafata. È disponibile al prezzo di 43,00 Euro IVA compresa (cod. EDU09).

articolo nel prossimo fascicolo di Elettronica In.

IL SOFTWARE

Il programma da PC che permette di visualizzare le forme d'onda

e analizzare i segnali prevede tre funzionalità:

- oscilloscopio;
- analizzatore di spettro;
- registratore di transienti (d'impulsi).



Che fare se abbiamo già sviluppato un'applicazione su una scheda Arduino e vogliamo dotarla di connettività internet? Lo YÚN shield è la risposta.

YÚN SHIELD CONNETTIVITÀ INTERNET PER TUTTE LE BOARD ARDUINO

..... di MARCO MAGAGNIN

Vi abbiamo presentato recentemente (sul n° 192) il microcomputer Linino ONE, la rivisitazione della società Dog Hunter della scheda Arduino YÚN. La soluzione Linino ONE consiste in una miniaturizzazione e modularizzazione della scheda Arduino YÚN, che utilizza gli stessi processori, gli stessi componenti di base ed offre le stesse funzionalità. Lo shield YÚN che vi presentiamo in questo articolo offre una soluzione ancora più flessibile ed intrigante. Permette di dotare delle funzionalità di connettività internet, proprie di Arduino YÚN, un'intera famiglia di schede della serie Arduino. Lo shield YÚN ospita il sistema operativo OpenWrt, un sistema operativo GNU/Linux appositamente progettato per la connettività internet. È lo stesso di Arduino YÚN

ed è comunemente adottato dagli Access Point con software Open Source. Bene, quale vantaggio ci offre lo shield YÚN rispetto all'Arduino YÚN originale? Semplice, lo shield si adatta a diversi modelli di Arduino, dando la possibilità di interagire con il mondo internet ad applicazioni già realizzate, rendendole adatte alle nuove visioni di interoperabilità e remotizzazione delle applicazioni. Ovviamente è possibile utilizzarlo in applicazioni nuove per le quali si vuole adottare un modulo diverso da Arduino Leonardo. Tra le schede Arduino, compatibili con lo shield YÚN, vediamo Arduino Uno, Arduino Duemilanove / Diecimila, Arduino Mega e, naturalmente, Arduino Leonardo, oltre a diverse altre. Altra piacevole sorpresa è la capacità dello shield di adattarsi sia

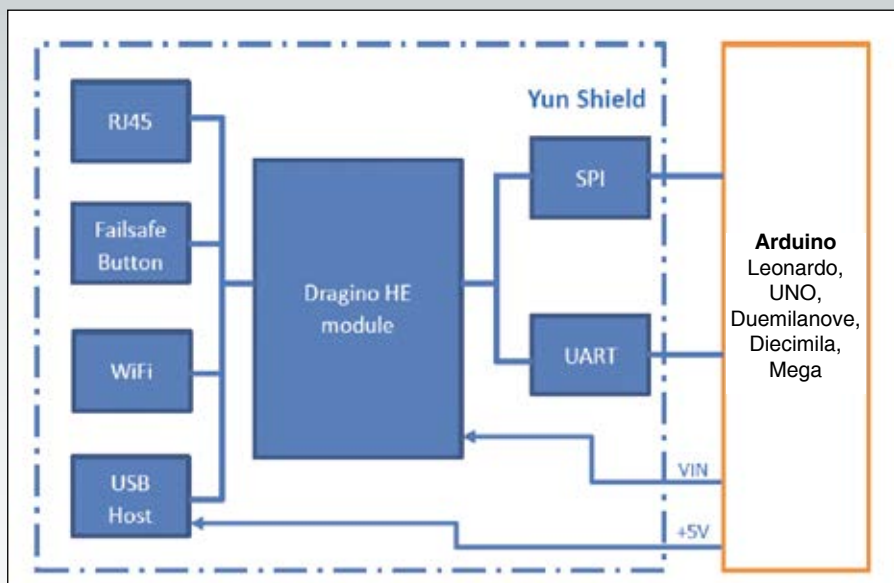


Fig. 1

il layout fisico dello shield YÚN. I connettori RJ45, WiFi, USB Host e Failsafe sono connesse direttamente al modulo Dragino HE. Il modulo Dragino HE, per comunicare con le schede Arduino, utilizza le porte SPI e UART. Lo shield Yun è compatibile con i moduli Arduino alimentati a sia a 3,3V che a 5V. Per adattare i livelli di tensione di funzionamento delle porte dello shield si configurano opportunamente i pin del jumper SV1. L'interfaccia SPI viene utilizzata unicamente durante il caricamento degli sketch dall'IDE di Arduino. Una volta terminato il caricamento dello sketch l'interfaccia SPI viene rilasciata e può essere utilizzata per collegarsi ad altre periferiche. L'interfaccia UART viene utilizzata in permanenza dalla libreria Bridge. Come per Arduino YÚN anche nel caso dello shield YÚN la libreria Bridge è il cuore del collegamento tra il processore Atheros AR9331, che ospita il sistema operativo GNU/Linux ed il microcontrollore del modulo Arduino adottato, pertanto l'interfaccia UART non può essere utilizzata da altre appli-

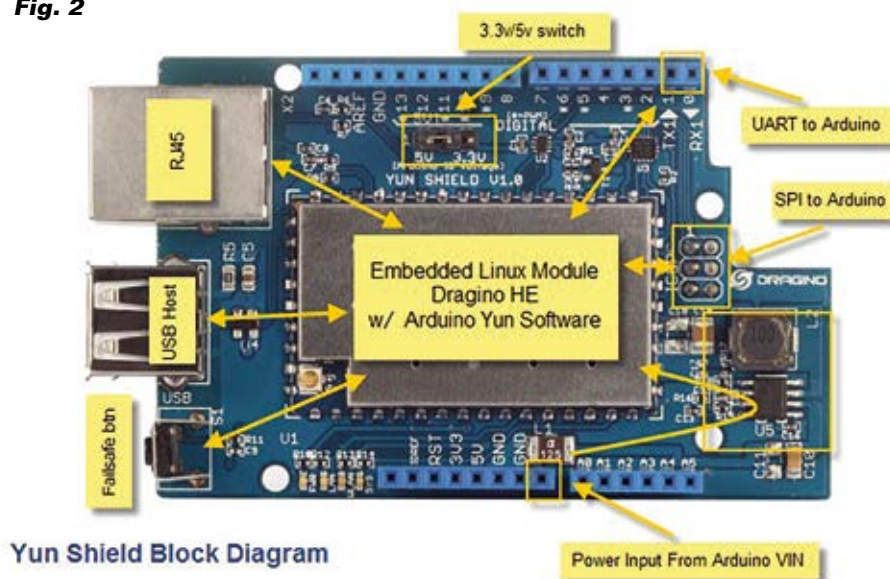
a moduli alimentati a 5V che a 3,3V. Tra le altre particolarità, un'antenna WiFi, già inclusa nella confezione, permette di instaurare connessioni WiFi particolarmente stabili nelle condizioni più disparate. Lo shield YÚN è un prodotto della società Dragino Technology, specializzata in prodotti Open Source finalizzati alle soluzioni IoT (Internet of Thing, o Internet delle Cose, che dir si voglia). Un'intera linea di prodotti che si basa su architetture che riprendono quella di Arduino YÚN, tra le quali quella che vi presentiamo.

ARCHITETTURA DEL SISTEMA

Nello schema a blocchi di Fig. 1 si vede come lo shield YÚN si collega alle schede che lo ospitano. Lo shield YÚN preleva l'alimentazione da queste schede dal pin VIN. Utilizza la connessione UART per comunicare con il modulo che lo ospita, grazie alla libreria Bridge, e la connessione SPI per la programmazione del modulo Arduino, con l'IDE collegato via TCP/IP. Il "cuore" dello shield YÚN è il modulo Dragino HE, a sua volta un compatto "microcomputer" basato sul processore Atheros AR9331, ampiamente utilizzato nelle soluzioni di connettività WiFi (anche in Arduino YÚN).

Possiede diverse interfacce come Ethernet, USB, UART, e diverse periferiche GPIO. Supporta gli standard WiFi 802.11b/g/n a 2,4Ghz, fino a 150Mb. Una curiosità, HÉ significa "core" in Cinese, che in Italiano si rende come "centro", "cuore", "nocciolo", a significare che il modulo Dragino HE si propone come centrale nelle soluzioni IoT in tecnologia WiFi. Ovviamente il sistema operativo che anima il modulo è l'Open Source OpenWrt, distribuzione open di GNU/Linux particolarmente dedicata alla connettività WiFi. Nella Fig. 2 è visibile

Fig. 2



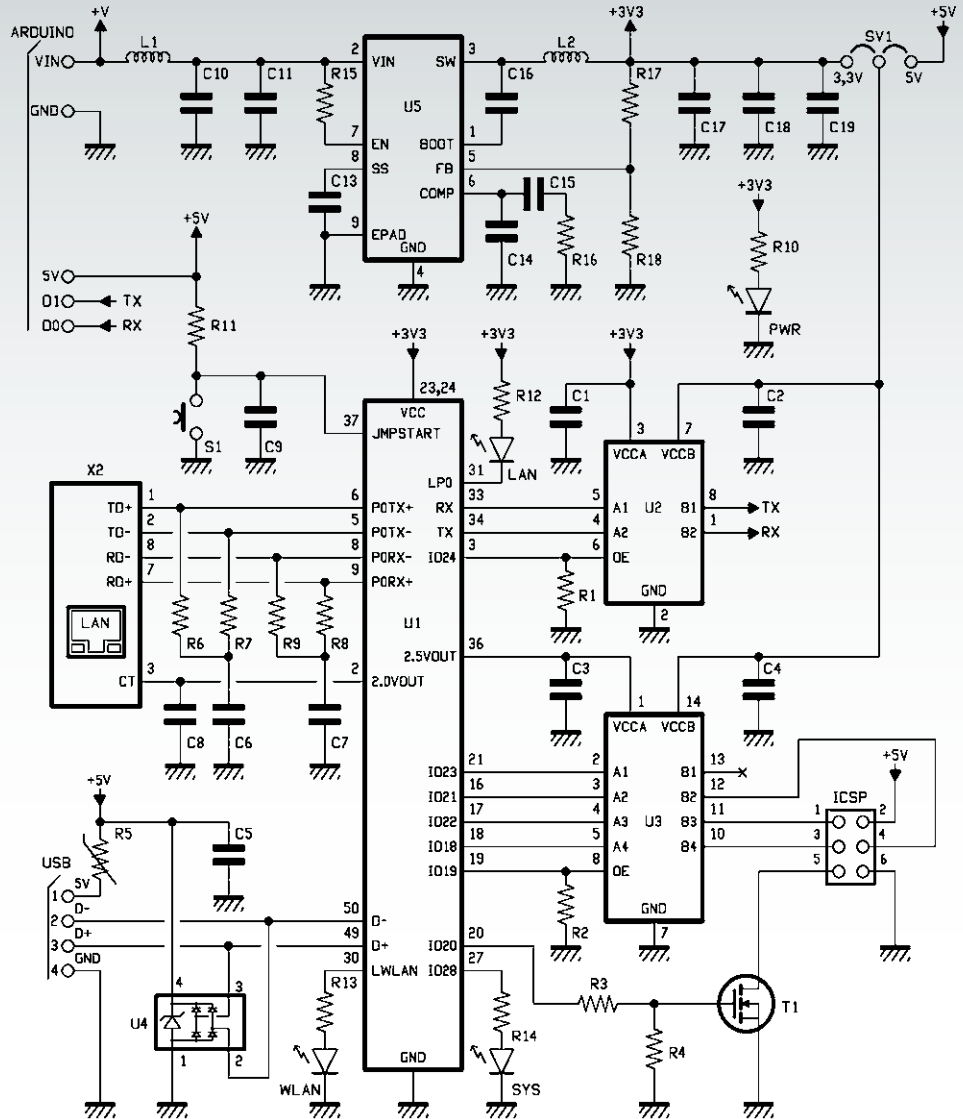
Elenco Componenti:

C1÷C4, C6÷C9, C13:
100 nF ceramico (0402)
C5, C12: 10 μ F 10 VL ceramico (0805)
C10, C11: 10 μ F 25 VL ceramico (1206)
C14, C17: -
C15: 3,9 nF ceramico (0402)
C16: 10 nF ceramico (0402)
C18, C19: 22 μ F ceramico (1210)
R1, R2, R18: 10 kohm (0402)
R3, R4: 47 kohm (0402)
R5: PTC (0805)
R6÷R9: 49,9 ohm (0402)
R10, R12: 330 ohm (0402)
R11, R15: 100 kohm (0402)
R13, R14: 270 ohm (0402)
R16: 6,81 kohm (0402)
R17: 26,1 kohm (0402)
S1: Microswitch 90°
T1: AP2302GN-HF (SOT23)
U1: Dragino HE
U2: NTB0102GT (SOT833)
U3: NTB0104 (QFN14)
U4: YSSR05 (SOT143)
U5: RT8250 (SOP8)
L1: Filtro soppressore 500mA
L2: Induttanza 10 μ H
LAN, PWR, SYS, WLAN:
LED giallo (0603)

Varie:

- Connettore TJ45 (RJ111BE)
- Connettore USB-A da CS (PN87520)
- Strip maschio 3 vie
- Jumper
- Strip maschio/femmina 6 vie (1 p.z)
- Strip maschio/femmina 8 vie (2 p.z)
- Strip maschio/femmina 10 vie (1 p.z)
- Strip maschio 2x3 vie (1 p.z)

[schema ELETTRICO]



cazioni. Lo schema di funzionamento del “ponte” tra la sezione GNU/Linux e processore del modulo Arduino è ampiamente descritto nel libro “Alla scoperta di Arduino YÚN”. Esattamente lo stesso meccanismo permette di far comunicare il processore dello shield YÚN ed il modulo Arduino di volta in volta utilizzato.

SPECIFICHE

- CPU: ATHEROS AR9331 processore MIPS 24K a 400MHz;
- Memoria RAM : 64MB;
- Memoria Flash : 16MB;

- Sistema Operativo: Open Source OpenWrt personalizzato;
- Alimentazione: 5V dal pin Arduino VIN;
- Un connettore Ethernet RJ45 10M/100M;
- WiFi: 2.4Ghz, 802.11 b/g/n, 150M
- Connettore I-PEX per collegare l'antenna interna inclusa nella confezione;
- Un connettore USB 2.0 host, utilizzabile per collegare una chiavetta USB od un dongle 3G;
- Un pulsante multifunzione

(indicato come Failsafe);

- Compatibile con i livelli di tensione UART ed SPI a 3.3V e 5V con ponticello di commutazione.

LED

Sullo shield sono presenti quattro LED con le seguenti funzioni:

- PWR: LED Power LED. Si accende quando lo shield è alimentato;
- LAN: Quando è acceso o lampeggia indica la presenza di una connessione di rete;
- WLAN: Indica la presenza di una connessione WiFi;

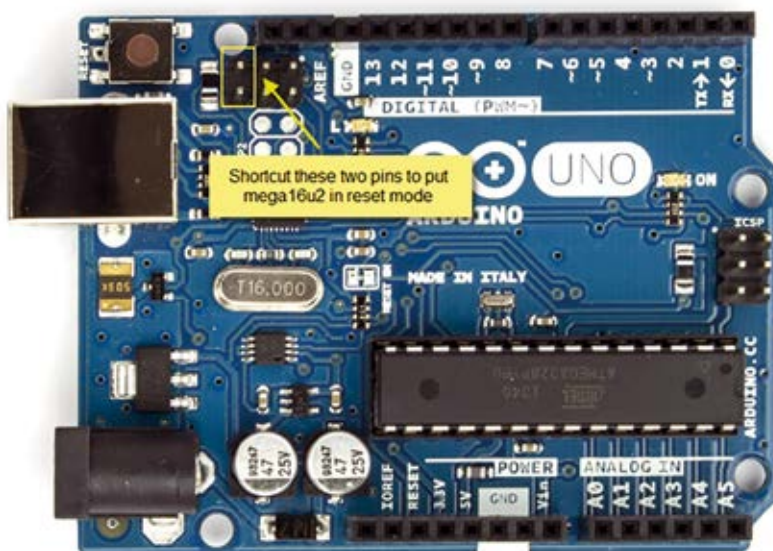


Fig. 3

- **SYS:** Indica la presenza di un'unità di massa USB. Quando è acceso indica che le cartelle presenti sulla memoria di massa USB sono montate sulla cartella /mnt, rispettivamente /mnt/sd e /www/sd.

PULSANTE MULTIFUNZIONE

Sullo shield YÚN è presente un solo pulsante indicato come "Failsafe" sul quale sono concentrate tutte le operazioni, di reset e restore, che possono essere necessarie nella gestione dello shield. Quando lo shield è acceso, dopo che il processo di boot è terminato, una pressione del pulsante Failsafe, di poco superiore ai 5 secondi, permette di riportare il chip WiFi alla configurazione di fabbrica. Una pressione per oltre 30 secondi riporta l'intero shield alla configurazione di fabbrica. L'ulteriore funzionalità del pulsante Failsafe permette di iniziare la procedura di salvataggio del modulo, da un upgrade di sistema terminato male. Il processo è assai complesso e troppo lungo da descrivere in un articolo. La raccomandazione che possiamo fare è di eseguire le operazioni di upgrade del firmware con molta attenzione per evitare di trovarsi con lo shield inutilizzabile. In

ogni caso le istruzioni di "salvataggio" si trovano all'indirizzo http://www.geeetech.com/wiki/index.php/YUN_SHIELD.

MODALITÀ DI ALIMENTAZIONE

Una questione importante nell'utilizzo dello shield YÚN è legata alle modalità di alimentazione. Lo shield riceve l'alimentazione dai pin GND e VIN. La modalità consigliata è di utilizzare questi pin. Il modulo Dragino HE, che costituisce il cuore dello shield Arduino, richiede circa 200mA di corrente a pieno carico, per questo è bene alimentarlo prelevando la tensione dai pin VIN e GND del modulo Arduino in modo da evitare di surriscaldare il modu-

lo di alimentazione LDO (Low Drop Out) a 5V della scheda. In questa configurazione, il modulo Arduino, a sua volta, deve essere alimentato dal connettore DC e non dalla porta USB. Fate attenzione ad utilizzare un alimentatore esterno che eroghi più di 7V.

UTILIZZO DELLO SHIELD YÚN CON I MODULI ARDUINO

Per sfruttare le opportunità offerte dallo shield YÚN è necessario un lavoro preparatorio sia per le schede che vogliamo collegare che per adattare l'IDE di Arduino a funzionare con le diverse combinazioni di moduli. Lo shield YÚN utilizza la porta SPI per caricare gli sketch e la porta seriale UART per comunicare, mediante la libreria Bridge, con il microcontrollore su cui è alloggiato. Per approfondimenti sul funzionamento della libreria Bridge rimandiamo, ancora una volta, al libro citato. Per poter stabilire il ponte di comunicazione tra lo shield YÚN ed il modulo Arduino è necessario controllare i seguenti punti:

- Il modulo Arduino deve essere alimentato dal connettore DC e non dal connettore USB;
- Il jumper dei livelli di tensione SPI e UART deve essere impostato coerentemente al modulo utilizzato;

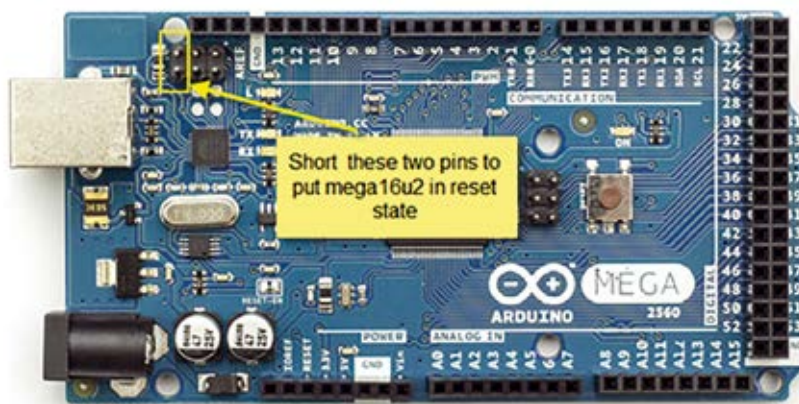


Fig. 4

- Il tipo di accoppiamento shield/modulo Arduino deve essere correttamente configurato nelle impostazioni dell'IDE. Nel proseguo dell'articolo descriviamo come modificare l'IDE di Arduino per adattarlo a riconoscere lo shield YÚN.
- Le porte SPI e UART non devono essere utilizzate da altri programmi;
- Assicuratevi che nello strumento di configurazione, nel menu "Sensor>>PowerUART" sia impostata la modalità "Linux Console/Arduino Bridge".

CONNETTERE LO SHIELD A LEONARDO

Iniziamo dalla descrizione di come connettere lo shield YÚN ad Arduino Leonardo. Si tratta, ovviamente, della combinazione più semplice, in quanto rispetta l'architettura di Arduino YÚN. Pertanto per associare allo shield YÚN il modulo Leonardo è sufficiente collocarli uno sopra l'altro. Fate attenzione che il jumper di controllo della tensione dei livelli sia posizionato sui 5V e date tensione. Per programmare il modulo Leonardo, nell'IDE di Arduino selezionate la board "Arduino Leonardo - Dragino YÚN".

CONNETTERE LO SHIELD AD ARDUINO UNO

Nel modulo Arduino UNO la connessione seriale UART tra i microcontrollori mega328P e mega16u2 interferisce con le funzionalità della libreria Bridge per la comunicazione con lo shield YÚN. È necessario disattivare il collegamento UART configurando il microcontrollore mega16u2 in modalità "reset". In pratica si tratta di cortocircuitare i pin indicati in Fig. 3 con il ponticello giallo incluso nella confezione dello shield YÚN. Dopo di che è

possibile "impacchettare" le due schede ed alimentarle utilizzando il plug DC del modulo Arduino UNO. Tenete presente che in questa configurazione non è più utilizzabile il collegamento al PC via USB di Arduino UNO. Per la programmazione ed il debug è necessario utilizzare la connessione WiFi scegliendo nell'IDE la board "Arduino Uno - Dragino YÚN".

CONNETTERE LO SHIELD AD ARDUINO MEGA2560

Anche nel modulo Arduino Mega2560 la connessione seriale UART tra i microcontrollori mega2560 e mega16u2 interferisce con le funzionalità della libreria Bridge. È necessario disattivare il collegamento UART configurando il microcontrollore mega16u2 in modalità "reset". Cortocircuitare i pin indicati in Fig. 4 con il ponticello giallo incluso nella confezione dello shield YÚN e alimentare le schede utilizzando il plug DC del modulo Arduino Mega2560. In questa configurazione non è più utilizzabile il collegamento al PC via USB di Arduino Mega. Per la programmazione ed il debug è necessario utilizzare la connessione WiFi scegliendo nell'IDE la board "Arduino Mega 2560 - Dragino YÚN".

CONFIGURAZIONE DELLO SHIELD YÚN

Lo shield YÚN ha a bordo sia il modulo WiFi che l'interfaccia LAN Ethernet, che possono essere utilizzate sia per connettersi ad internet che per l'amministrazione del sistema. Lo shield YÚN può funzionare in modo autonomo, basta alimentarlo e si trasforma in un access point di rete. Per la prima prova abbiamo scelto di montarlo su un modulo Arduino Uno, abbiamo collegato

la piccola antenna inclusa nella confezione, al connettore predisposto in un "angolo" del modulo Dragino HE, abbiamo impostato il jumper di configurazione della tensione di alimentazione su 5V e abbiamo alimentato il tutto dal connettore a barilotto di Arduino Uno. La tensione di alimentazione raccomandata è di 7,5V. Al momento del boot lo shield YÚN, come configurazione predefinita, si presenta come un Access Point con SSID di nome "Dragino-xxxxxx", come visibile in Fig. 5, dove le "x" stanno per il MAC Address dello shield. Per collegarsi la prima volta allo shield YÚN, il metodo è del tutto simile a quello richiesto per Arduino YÚN. Si utilizza un PC dotato di accesso WiFi, ci si collega all'SSID dello shield, senza password, poi si apre un browser e si digita l'indirizzo 192.168.240.1. Ovviamente la connessione del PC deve essere configurata in modo da acquisire un indirizzo IP, dal server DHCP dello shield. A questo punto si



Fig. 5

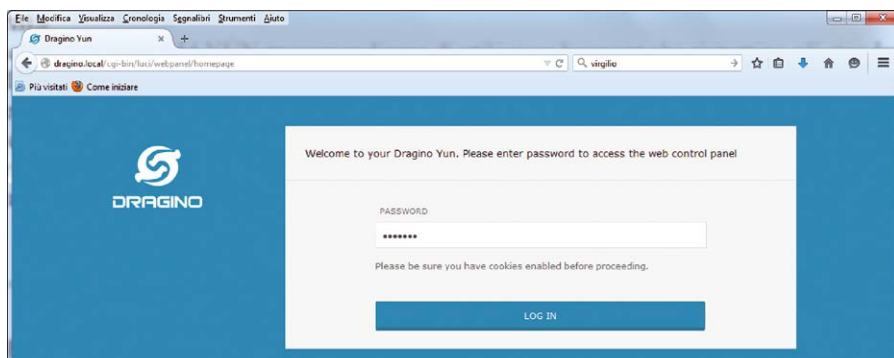


Fig. 6

apre la pagina di accesso allo strumento di configurazione LuCI. Vi è anche la possibilità di collegarsi allo shield via connessione SSH utilizzando strumenti come Putty, WinSCP o MobaXterm, tutti strumenti il cui utilizzo è stato ampiamente descritto nei numeri precedenti della rivista. Per accedere allo shield YÚN raccomandiamo di utilizzare browser che rispettino gli standard HTML5 come Firefox e Chrome. Per accedere allo shield possiamo digitare nel

campo URL il nome host dello shield "dragino.local", se non funziona, a causa della nostra configurazione di rete, digitiamo l'indirizzo IP dello shield. Se siamo collegati allo shield YÚN utilizzando l'Access Point predefinito, digitiamo l'indirizzo IP dell'Access Point stesso, ovvero 192.168.240.1. Se, invece, siamo collegati allo shield YÚN via rete cablata, è necessario conoscere l'indirizzo IP che viene assegnato dal DHCP del router allo shield. In mancanza, nel sistema è pre-

configurato un indirizzo di Fall-back, e precisamente l'indirizzo IP 172.31.255.253 con maschera di rete 255.255.255.252 (Attenzione). In questo modo è possibile collegarsi allo shield con un cavo di rete, impostando sul proprio PC l'indirizzo IP 172.31.255.253 sempre con maschera di rete 255.255.255.252.

In qualsiasi modo ci colleghiamo, si aprirà la pagina di accesso dello strumento di configurazione, visibile in **Fig. 6**.

Si accede allo strumento di configurazione digitando la password "dragino", che è la password dell'unico utente predefinito, "root", come d'uso per i router e gli Access Point. Otteniamo la pagina visibile in **Fig. 7** che ci mostra la configurazione delle interfacce di rete attive. Già questa pagina ci dice che le interfacce di

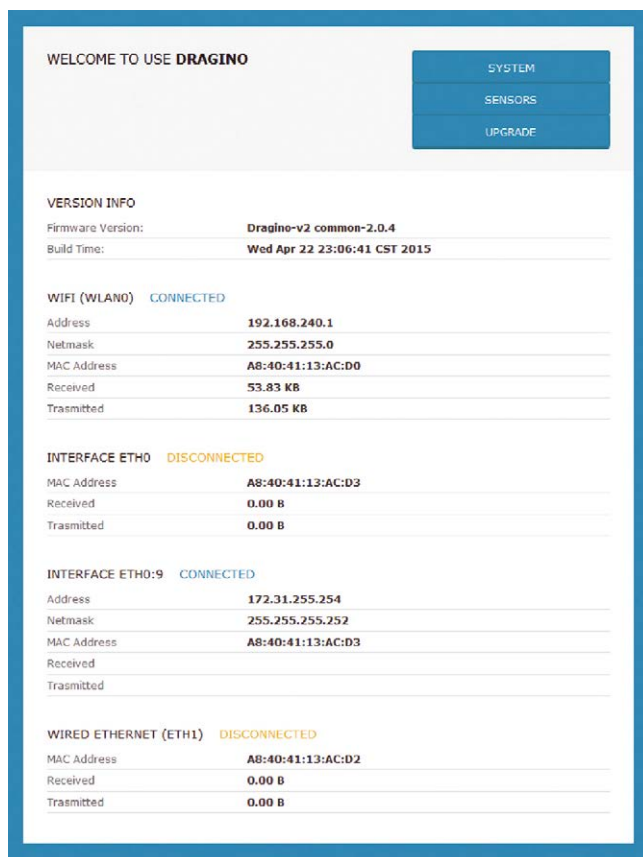


Fig. 7

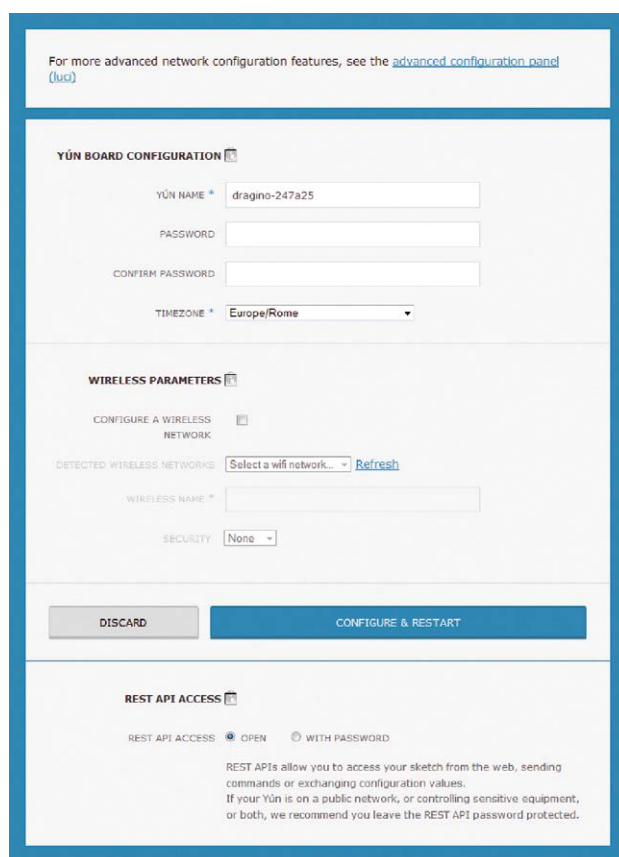


Fig. 8

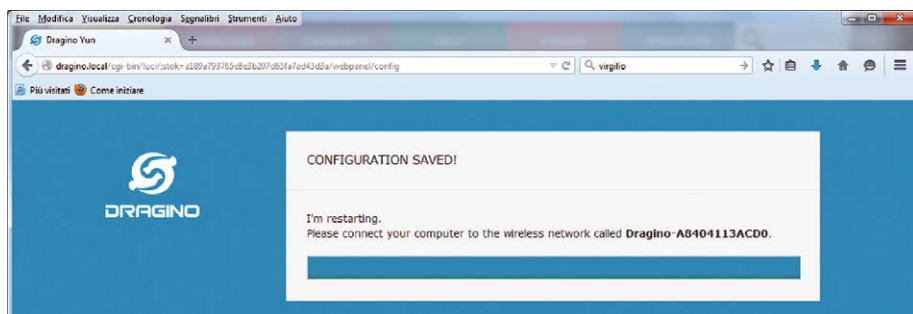


Fig. 9

pagina un po' verso il basso troviamo la sezione che ci permette di configurare l'interfaccia WiFi della scheda nelle diverse modalità (Access Point, Client, Ad Hoc, ecc). Le altre sezioni di configurazione ci permettono di impostare le regole di sicurezza accessi (Firewall), le regole di Routing, il DHCP ed il DNS (**Fig. 11**).

rete sono preconfigurate ed un web server è attivo con un'applicazione già installata che permette di configurarne il funzionamento lato GNU/Linux.

Sempre nella pagina di **Fig. 7** vediamo qual'è l'eventuale indirizzo IP assegnato alla scheda lato WAN, se abbiamo collegato un cavo di rete, e la configurazione della sezione WiFi, già attiva e preconfigurata come Access Point con il proprio IP statico. In alto a destra nella pagina troviamo i pulsanti che ci permettono di accedere alle tre principali sezioni di configurazione del sistema: "SYSTEM", "SENSORS" e "UPGRADE". Per il momento clicchiamo su "SYSTEM", che ci aprirà la sezione di configurazione di sistema visibile in **Fig. 8**, analoga a quelle di Arduino YÚN e Linino. In questa pagina configuriamo la Time Zone corretta (Europe/Rome), e selezioniamo il check box "Open" nella sezione "Rest API Access". Confermiamo cliccando sul grosso pulsante "CONFIGURE & RESTART" e aspettiamo che abbia termine la riconfigurazione del sistema, che dura alcuni minuti, durante la quale abbiate pazienza e non toccate nulla (**Fig. 9**). Ci ricollegiamo allo shield, clicchiamo sul link "advanced configuration panel" ed entriamo nell'ambiente di amministrazione vero e proprio. Dalle pagine di menù, in alto, possiamo accedere alle diverse sezioni, sintetizzate in **Fig. 10**.

Se, per curiosità, andiamo alla sezione "Network" e "Wifi", all'in-

terno della pagina clicchiamo sul pulsante "Edit" e scorriamo la

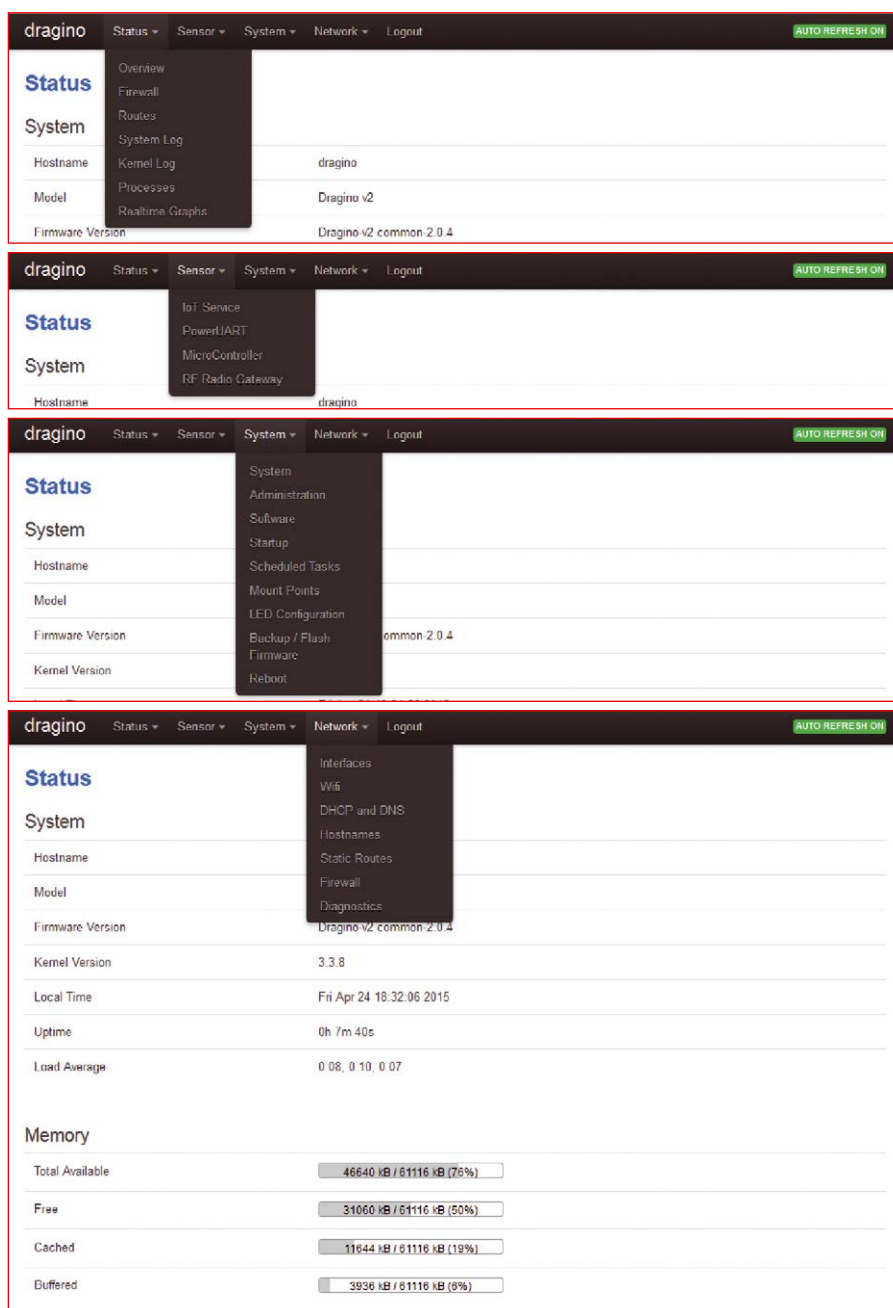


Fig. 10

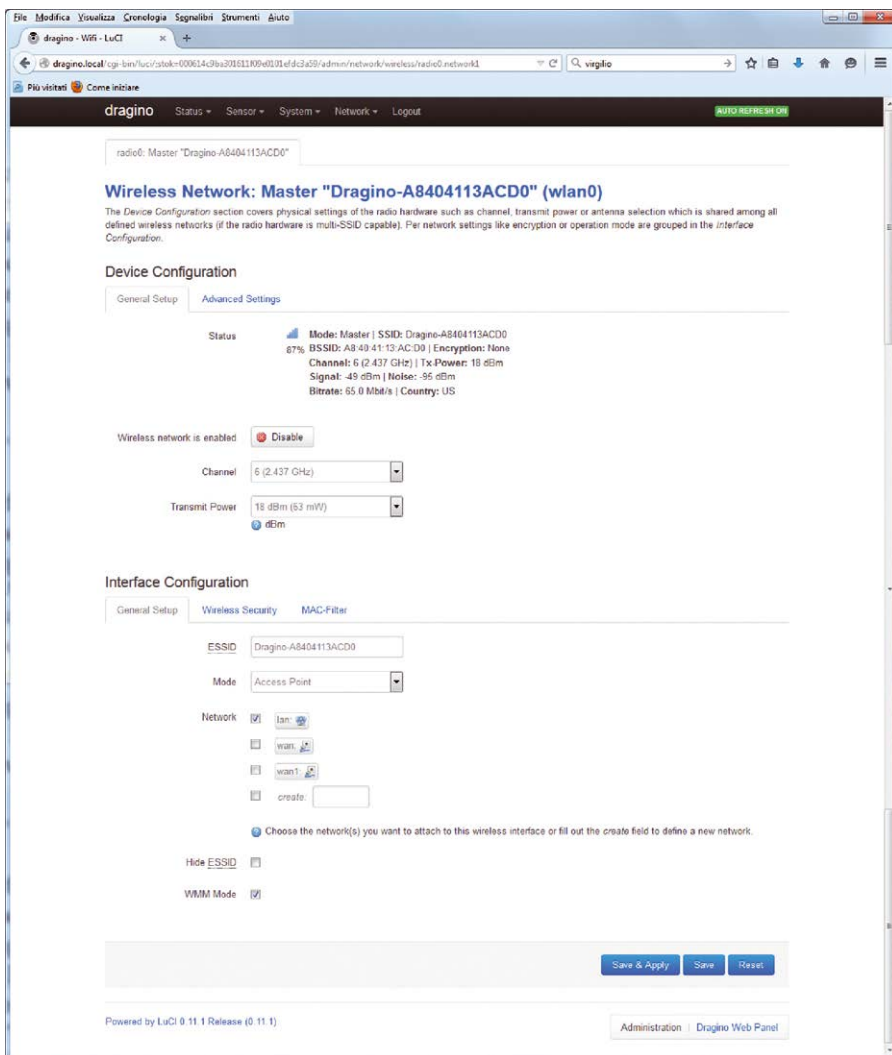


Fig. 11

UPGRADE FIRMWARE

Anticipiamo la descrizione di questa sezione in quanto è opportuno aggiornare il firmware dello shield all'ultima release disponibile. Il motivo è dato principalmente dalle modifiche apportate alla sezione "SENSORS", che ha subito variazioni di progetto nel corso dei rilasci. Pensata inizialmente per facilitare il collegamento a servizi di raccolta ed elaborazioni dati da sensori come Xively, è stato poi modificato per configurare anche il tipo di modulo Arduino sul quale lo shield viene alloggiato. Gli aggiornamenti alle diverse versioni di firmware sono visibili all'indirizzo:

http://wiki.dragino.com/index.php?title=Yun_Firmware_Change_Log

Le diverse release di firmware da scaricare si trovano all'indirizzo:

<http://www.dragino.com/downloads/index.php?dir=motherboards/ms14/Firmware/Yun/>

Prima di continuare vi raccomandiamo di eseguire il processo di aggiornamento del firmware con molta attenzione, con calma e controllando più volte ogni operazione, prima e dopo averla

In definitiva, ci troviamo di fronte ad un vero e proprio dispositivo di connettività che ci permetterà di risolvere una vasta gamma di applicazioni, di esigenze di elaborazione e di configurazione di rete. Un'ultima curiosità, proviamo a connetterci alla scheda utilizzando i client SSH Putty e WinSCP (per chi volesse saperne di più su tutto quanto detto finora, incluso la maggior parte delle pagine di configurazione, rimandiamo agli articoli introduttivi su Raspberry Pi ed al libro "Il mio primo Linux embedded"). Ci connettiamo con l'indirizzo IP dello shield YUN, indicando l'utente "root" e password "dragino". Otteniamo i risultati visibili in Fig. 12 e Fig. 13, che ci dimostrano che anche il

server SSH è attivo e funzionante "out of the box".

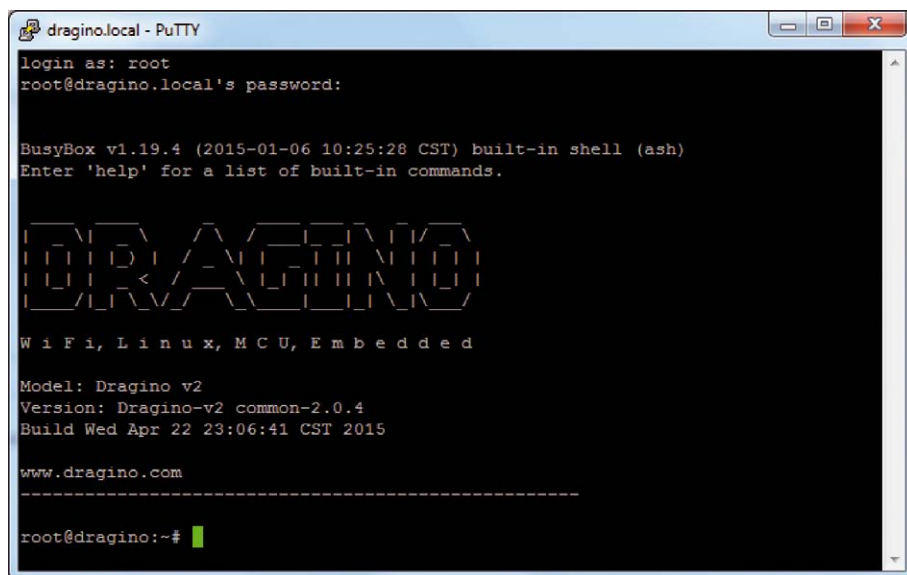


Fig. 12

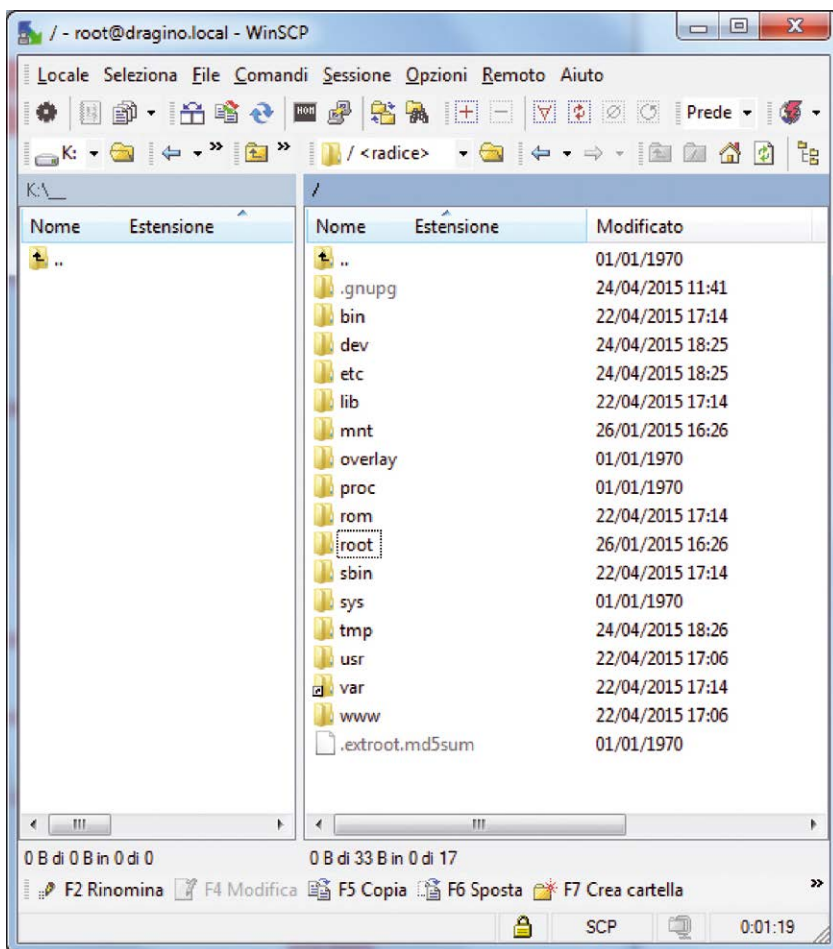


Fig. 13

mandazione, togliete il segno di spunta all'opzione "Keep Setting" (Fig. 14). Questa operazione ha lo svantaggio di azzerare le eventuali configurazioni di sistema preesistenti, ma garantisce dal creare disallineamenti tra i file di configurazioni preesistenti e quelli della nuova versione. Questa è una delle condizioni che potrebbero "bloccare" lo shield dopo un aggiornamento firmware. Col pulsante "Sfoglia" selezionate il file che avete appena scaricato. Tutto a posto? Ricontrollate. Poi cliccate sul pulsante "Upload Firmware" (Fig. 15).

Ora lasciate in pace il sistema finché non è terminata l'operazione di aggiornamento, che richiede diversi lunghi minuti (Fig. 16). Senza toccare nulla osservate i LED gialli sullo shield.

Lampeggiano in vario modo. Prima di provare a ricollegarvi aspettate che i LED si siano riacciati in modo stabile. Non necessariamente tutti e tre, se non è presente una connessione LAN via cavo, il relativo LED rimane spento. In particolare curate quello che indica l'attività del chip WiFi, che è l'ultimo a riattivarsi.

eseguita. Prendetevi tutto il tempo necessario, l'operazione è abbastanza lunga e verificate di utilizzare una connessione di rete stabile ed una fonte di alimentazione per lo shield adatta ed affidabile.

Come prima operazione è necessario scaricare il firmware da trasferire sullo shield. Andate all'indirizzo di "Download", cliccate sulla cartella "Newest Firmware" ed infine sulla cartella "common-build-xxxxxxx". Attenzione, a seconda delle release queste indicazioni potrebbero non essere completamente esatte. Nella cartella che si apre vediamo alcune immagini di firmware. Quella che dovete scaricare ed utilizzare è quella che contiene nel nome il termine "sysupgrade" e suffisso .bin. Scaricatela e salvatela in una cartella del PC. Al termine andate alla pagina di configurazione

dello shield e cliccate sul pulsante "UPGRADE". In questa pagina è possibile scegliere il firmware da utilizzare, per poi procedere all'aggiornamento. Una racco-

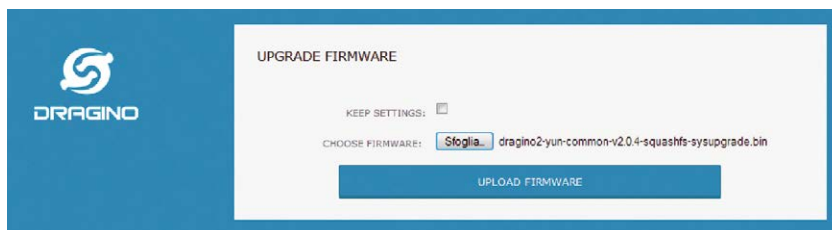


Fig. 14

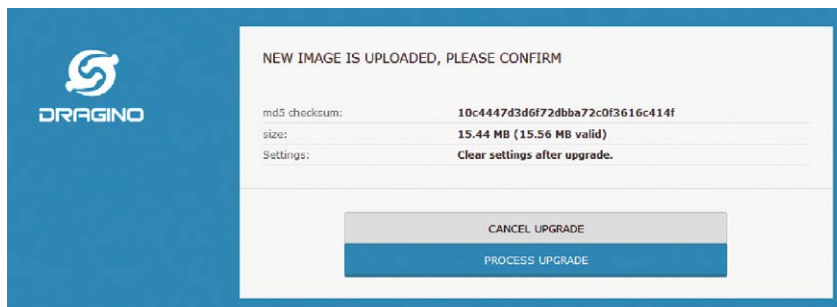


Fig. 15

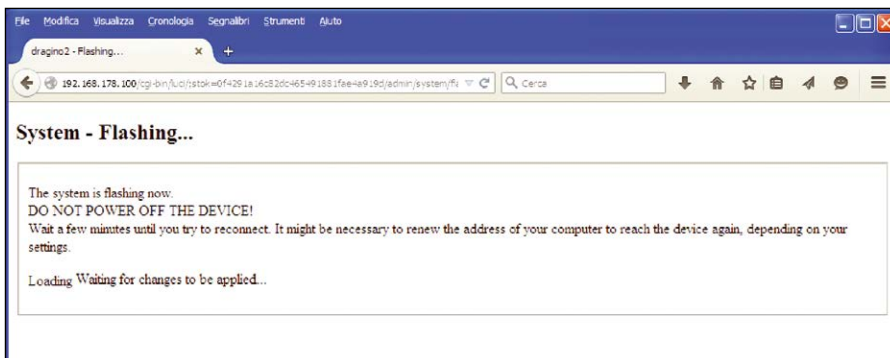


Fig. 16

firmware stanno a significare che ci troviamo di fronte ad un prodotto in crescita, che vuole trovare la sua collocazione nel mondo dell'Internet delle Cose. Al momento non possiamo che giudicare in modo positivo questa condizione.

Terminato? Tutto a posto? Potete ricollegarvi allo shield.

SENSORS

La sezione "Sensors" presenta funzionalità differenti a seconda della release del firmware installato. Se avete installato l'ultima release, aprendo la pagina troverete la possibilità di configurare il modulo Arduino sul quale è stato alloggiato lo shield che, in genere, viene riconosciuto automaticamente. Diversamente da quanto accade per Arduino YÚN e per Linino, questa funzionalità è necessaria per configurare correttamente le impostazioni da utilizzare durante l'upload degli sketch come il tipo di MCU, di bootloader e le impostazioni dei fuse.

A questo proposito una funzionalità simpatica presente nella pagina permette di caricare gli sketch direttamente a partire dal file .hex, frutto della compilazione. Questa operazione permette di evitare ogni volta la compilazione degli sketch, ma soprattutto permette di non dover necessariamente utilizzare l'IDE di Arduino per il solo upload degli sketch (Fig. 17). Una funzione presente nel menu "SYSTEM">>advanced configuration panel>>Sensor>>Microcontroller" permette di automatizzare l'aggiornamento alle nuove versioni degli sketch ad ogni boot (Fig. 18). La presenza di funzioni simili in sezioni differenti dello strumento di configurazione e le frequenti release di aggiornamento del

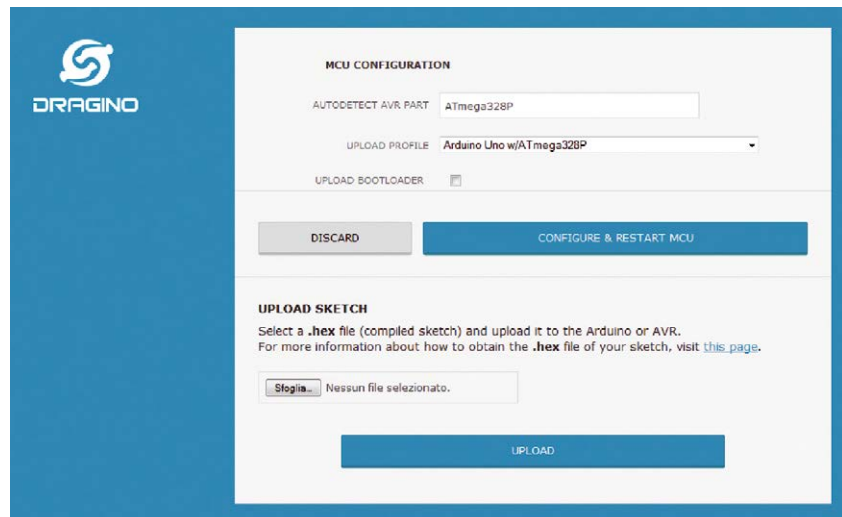


Fig. 17

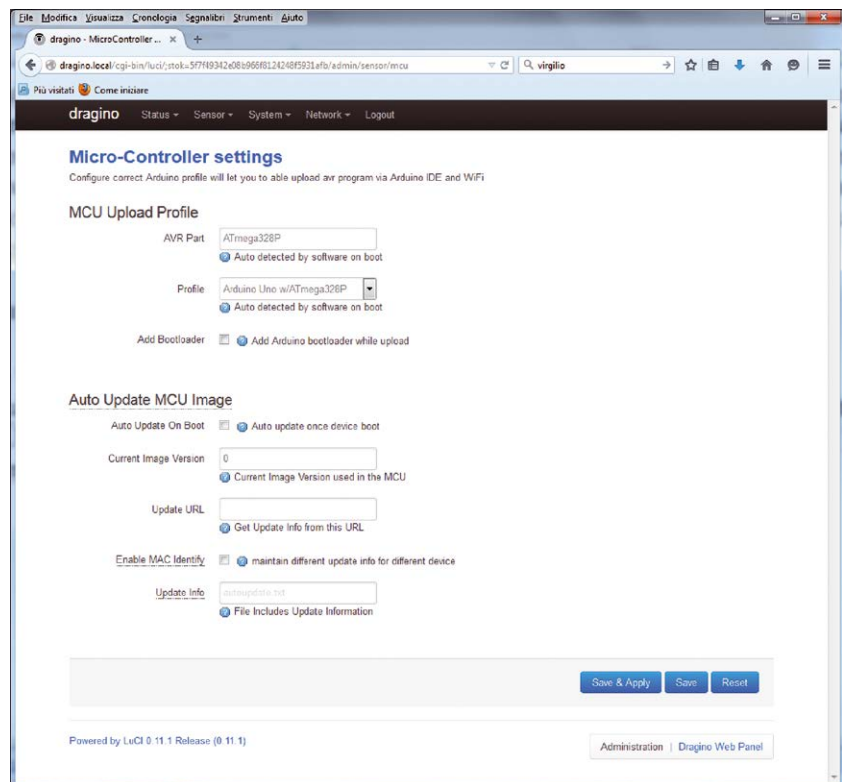


Fig. 18

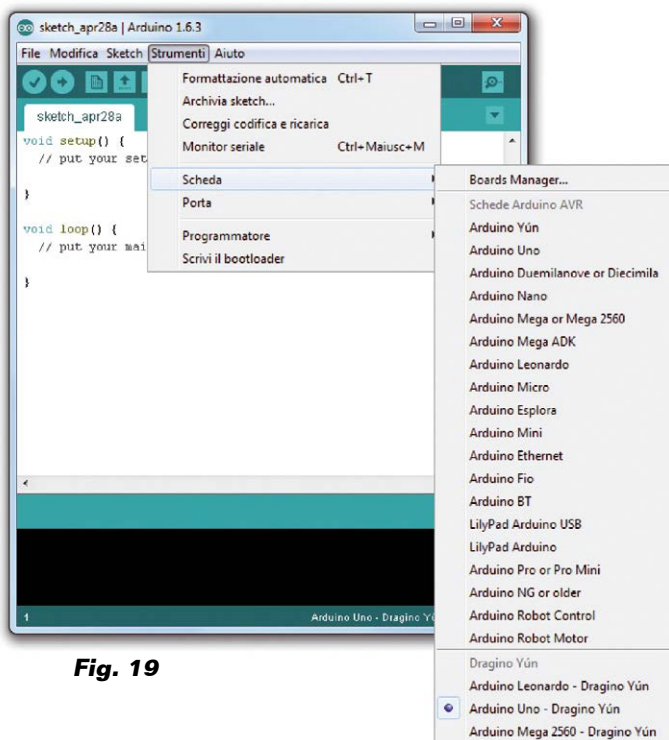


Fig. 19

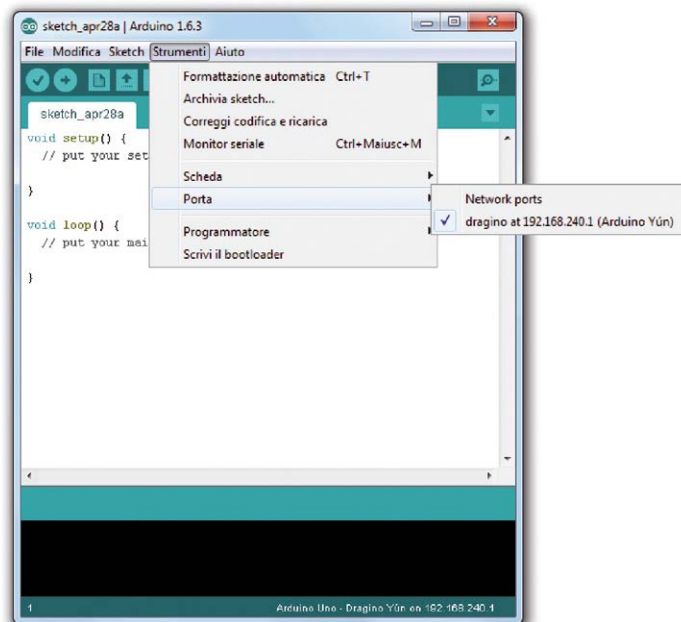


Fig. 20

PROGRAMMAZIONE

Lo shield YÚN, o meglio la combinazione shield più modulo Arduino si programma utilizzando l'IDE di Arduino. Prima, però, è necessario personalizzare l'IDE

per permettere di riconoscere e programmare le diverse combinazioni di shield e moduli. Abbiamo già anticipato che possiamo scaricare l'IDE di sviluppo per Arduino YÚN dal sito di Arduino. Per

le nostre prove abbiamo deciso di utilizzare l'ultima versione, al momento disponibile, dell'IDE di Arduino, la 1.6.3, che si può scaricare dall'indirizzo <http://www.arduino.cc/en/main/Software>.

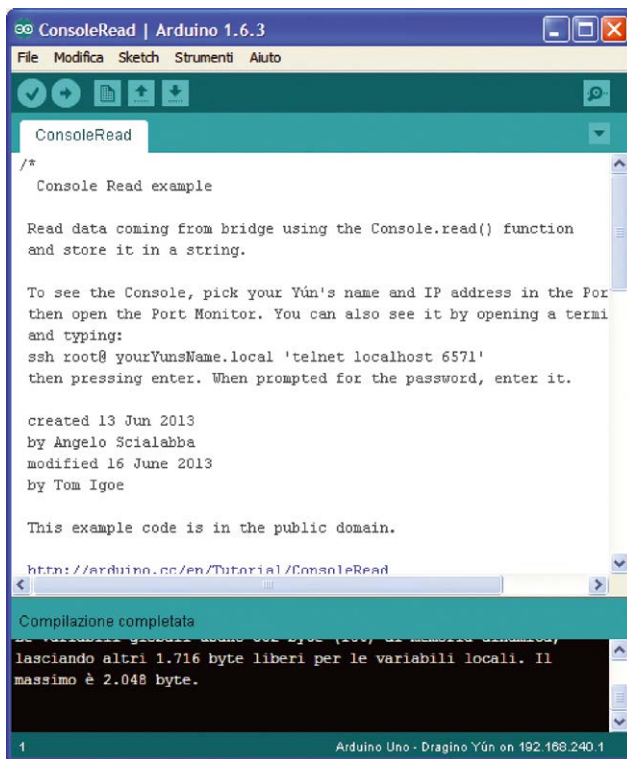


Fig. 21

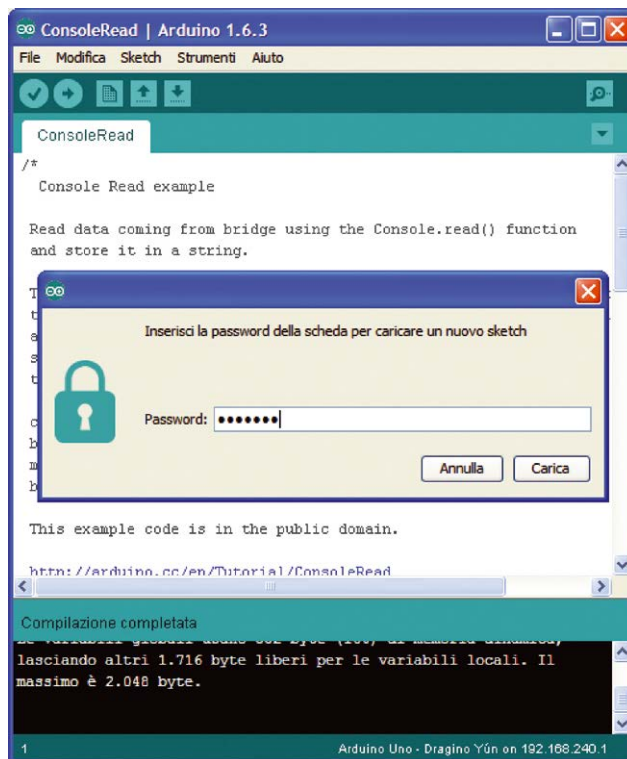


Fig. 22

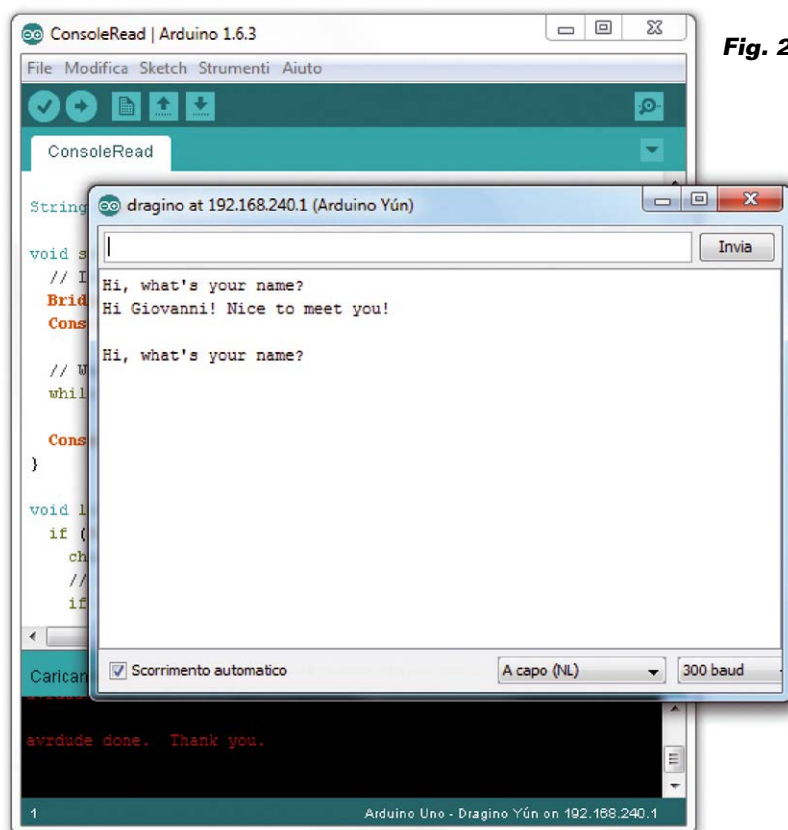


Fig. 23

Questa release è particolarmente adatta ad essere personalizzata con componenti di terze parti, facilitando così l'utilizzo con schede e moduli "compatibili". Noi abbiamo scelto la versione che non richiede installazione, caratterizzata dal suffisso .zip. Basta solo scompattarla in una cartella dedicata. Creiamo un collegamento del programma "arduino.exe" sul desktop. Per personalizzare l'IDE di Arduino dobbiamo scaricare i software necessari dal repository di GitHub dall'indirizzo <https://github.com/dragino/modules>, dove è possibile scaricare l'intera libreria in formato compresso (.zip) per poi scompattarla in una cartella di comodo. All'interno della cartella bisogna ricercare la sottocartella "modules-master\hardware\YunShield\sketchbook\hardware\YunShield" e copiarla nell'IDE di Arduino sotto la cartella "Arduino_1_6_3\arduino-1.6.3\hardware". Collegiamo il PC all'insieme shield più modulo mediante la con-

nessione WiFi. Apriamo l'IDE di Arduino. Nella voce di menu "Strumenti>>Scheda" scegliamo la configurazione, nel nostro caso "Arduino Uno - Dragino Yún" (Fig. 19). Poi, nella voce di menu "Strumenti>>Porta", selezioniamo la connessione via TCP/IP (Fig. 20). Quella che riporta l'indirizzo IP dello shield. Se non vediamo la connessione tra shield e PC, apriamo una connessione SSH con PuTTY e diamo il comando:

```
/etc/init.d/avahi-daemon restart
```

che forza OpenWrt a inviare in rete la disponibilità del servizio di connessione. Se lo shield non viene individuato al primo comando ripetete l'operazione. Potrebbe capitare che, nonostante tutto, non si veda ancora la scheda. In questo caso sapere che la connessione avviene utilizzando la porta 5353 UDP della connessione TCP/IP ci può aiutare. La causa del malfunzionamento può essere

un firewall, per esempio quello dell'antivirus, oppure del router, che blocca l'utilizzo della porta, oppure una configurazione degli indirizzi di rete che non permette al PC di vedere la scheda. Per chi volesse approfondire ulteriormente questo argomento rimandiamo sempre al libro citato in precedenza. A questo punto, per provare che il tutto funzioni correttamente, possiamo provare a caricare uno sketch. Abbiamo scelto lo sketch "Console Read", che ci permette di provare il "ponte" tra il modulo Arduino e lo shield Yún. Lo sketch è raggiungibile dal menu File>>Esempi>>Bridge>>Console Read. Compiliamo e carichiamo lo sketch nel solito modo. Alla richiesta di password per l'Upload rispondiamo con "dragino" (Fig. 21 e Fig. 22). Apriamo il monitor seriale dell'IDE di Arduino dal menu "Strumenti>>Monitor Seriale". Impostiamo la velocità di comunicazione a 115200 baud. Apparirà la semplice applicazione interattiva visibile in Fig. 23.

UN ESEMPIO: MQTT

Approfittiamo della presentazione dello shield Yún e della sua vocazione a semplificare le soluzioni orientate all'IoT, per introdurre un esempio di configurazione ed utilizzo del protocollo MQTT, che si sta sempre più diffondendo nella comunicazione tra sensori e server di servizio dedicati proprio all'IoT. Il protocollo MQTT (Message Queue Telemetry Transport) è stato sviluppato nel 1999 da IBM e Arcom, ora Cirrus Link, per la comunicazione dei dati di telemetria per il controllo di migliaia di chilometri di oleodotti e poi in campo medico per comunicare con pacemaker e altri dispositivi medici. Data la sua semplicità e la possibilità di essere implementato su sistemi embedded dalle risorse ridotte si

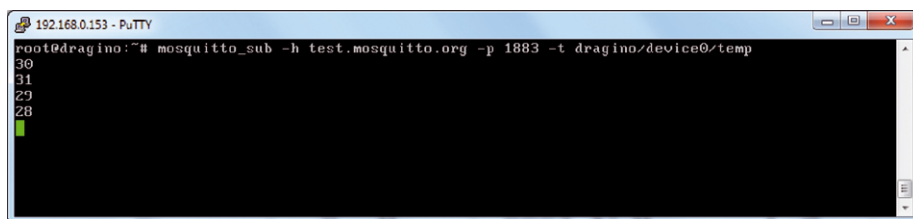


Fig. 24

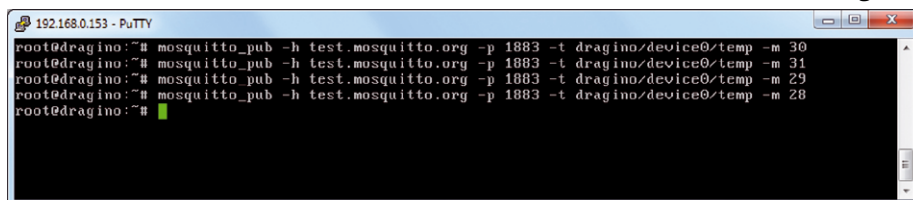


Fig. 25

sta imponendo come protocollo di riferimento per ciò che viene definito come IoT. Il consorzio OASIS ha ora costituito un nuovo comitato tecnico che consentirà ai prodotti che comunicano via MQTT di lavorare con più tipi di reti, dispositivi e applicazioni software. Tra le aziende che sono coinvolte nel progetto figurano Cisco, IBM, Red Hat, Software AG e Tibco. MQTT è un protocollo che implementa un broker di messaggistica ai quali i diversi utilizzatori si iscrivono come client di pubblicazione e/o di sottoscrizione. Caratteristica peculiare è che viene supportata la comunicazione molti a molti. Utilizzeremo questo protocollo nei prossimi articoli per proporre delle soluzioni, anche se di livello amatoriale, che vanno nella direzione dell'IoT. Per ora vi descriviamo il funzionamento di base del protocollo MQTT utilizzando proprio lo shield YÚN. Dobbiamo assicurarci che lo shield YÚN sia connesso ad internet. Apriamo una connessione remota con SSH allo shield utilizzando lo strumento PuTTY e digitiamo i comandi necessari ad installare il client MQTT, necessario a comunicare con il broker di test.

```
opkg update
opkg install libcacas
opkg install libmosquitto
opkg install mosquitto-client
```

Per la nostra prova utilizziamo il server MQTT pubblico "test.mosquitto.org", un server broker MQTT pubblicamente utilizzabile per i test di comunicazione. Gli utenti possono pubblicare e sottoscrivere "code" di messaggi su questo server per eseguire i propri test. Per eseguire le prove con messaggi "sicuri" è necessario scaricare il file contenente il certificato di autenticazione della connessione mosquitto.org.crt. Il servizio di test ascolta sulle porte 1883, 8883, 8884 e 8885. La prima è libera, le altre richiedono il certificato di autenticazione per stabilire la connessione. Per eseguire la nostra prova apriamo due finestre SSH con PuTTY. Nella prima finestra ci sottoscriviamo ad una coda di messaggi con il comando:

```
mosquitto_sub -h test.mosquitto.org -p 1883 -t dragino/device0/temp
```

Questo comando apre una connessione al broker e si mette in ascolto della coda di nome "temperature" sulla porta 1883 (Fig. 24). Nella seconda finestra digitiamo il comando che ci permette di pubblicare dati, nel nostro caso valori di temperatura, sulla stessa coda (Fig. 25):

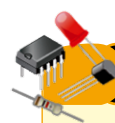
```
mosquitto_pub -h test.mosquitto.org -p 1883 -t dragino/device0/temp -m 31
```

Come inviamo il comando vediamo il risultato nella prima finestra. Divertitevi a sperimentare con diverse code e molteplici finestre di sottoscrizione e pubblicazione.

COME IMPOSTARE LE CARTELLE SULLA SD CARD

Concludiamo questo articolo con il suggerimento su come impostare l'equivalente delle cartelle e sottocartelle sulla SD Card per l'utilizzo di pagine web personalizzate come è possibile fare con Arduino YÚN.

Avrete notato che lo shield YÚN non dispone di un alloggiamento per una SD Card. Per ottenere lo stesso risultato utilizziamo una memoria di massa USB, la cosiddetta "chiavetta" USB, sulla quale creiamo le cartelle arduino e arduino/www. Una volta inserita la "chiavetta" nel connettore USB dello shield il sistema riconoscerà le cartelle e le monterà automaticamente sotto la cartella mnt/sd, esattamente come succede con Arduino YÚN. ■



per il MATERIALE

Tutto il materiale utilizzato in questo progetto è disponibile presso Futura Elettronica. Lo YUN Shield per Arduino UNO e Mega (cod. YUNSHIELD) viene fornito montato e collaudato al prezzo di Euro 45,00; la scheda Arduino Uno Rev3 (cod. ARDUINOUNOREV3) costa 24,50 Euro; la board Arduino Mega2560 Rev3 (cod. ARDUINOMEGA2560REV3) è disponibile al prezzo di 51,00 Euro. 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>



Montiamo un rasaerba con motore a scoppio a bordo di una piattaforma motorizzata comandabile a distanza tramite il joypad della PS2

ROBOT TAGLIAERBA

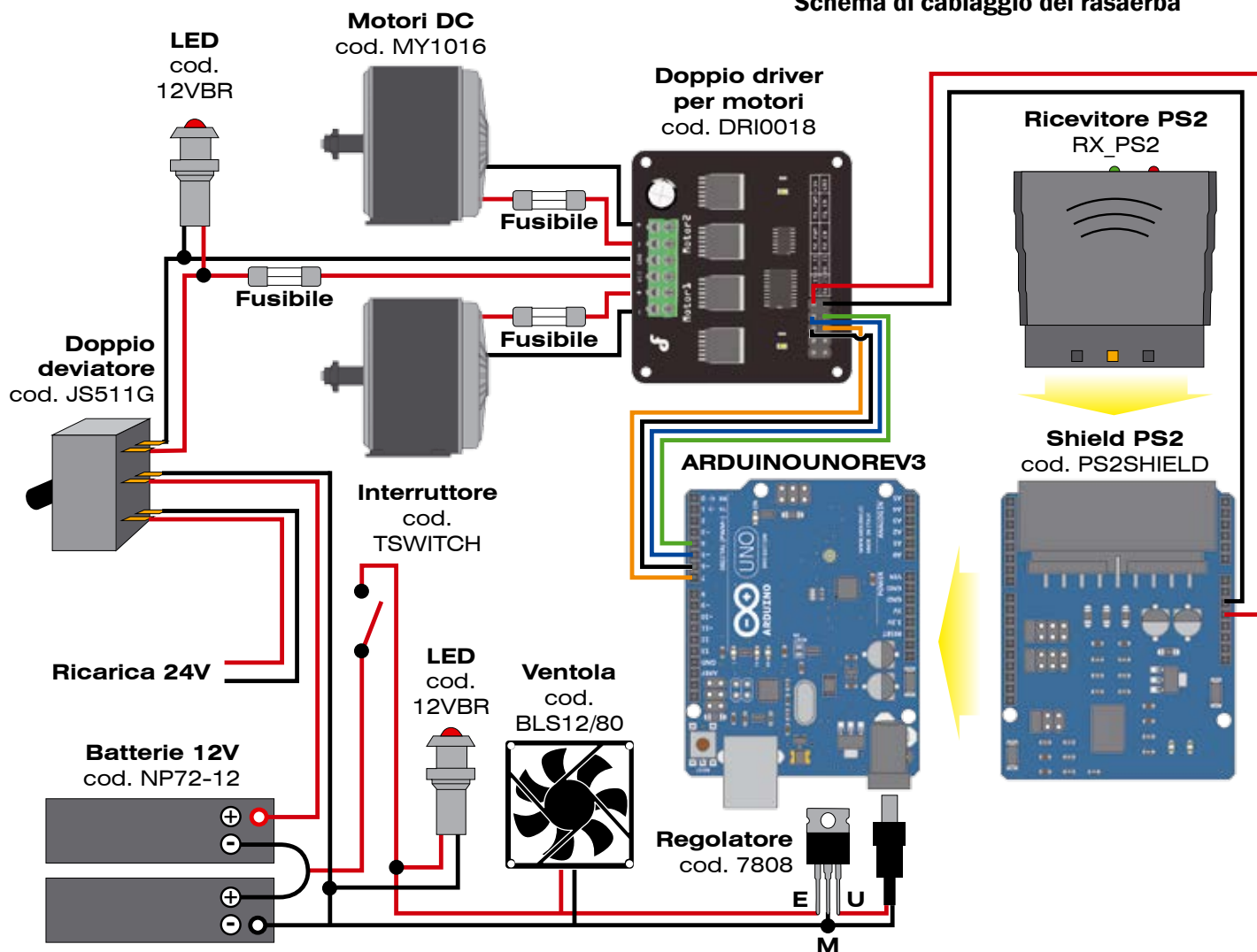
di SAMUEL MAGISTRO

Anche se il tempo è un po' capriccioso e il sole si alterna a qualche pioggia di troppo, possiamo dire che la bella stagione è finalmente arrivata; il primo sole ci fa tornare la voglia di una giornata all'aria aperta e magari di un barbecue in giardino. Ma un bel prato verde necessita di essere falciato, soprattutto quando il sole batte dopo un periodo di pioggia e l'erba ha tutto quel che le serve per crescere rapidamente e, oltre a dare un aspetto disordinato al giardino, diventa fastidiosa. E allora bisogna rispolverare il tosaerba svegliandolo dal

"letargo" cui l'avevamo lasciato in inverno e mettersi all'opera.

Non tutti sanno che la falciatura del prato è un lavoro che si deve ripetere anche una volta a settimana e con l'inevitabile problema di sporcarsi e sudare sotto il sole, senza parlare poi di chi soffre di allergie varie a sostanze quali polline, graminacee, fieno ed erba appena tagliata, che rendono difficoltoso respirare, provocano gonfiore agli occhi, riniti e continui fastidi. Se avete di questi problemi o semplicemente volete risparmiarvi fatica e sudo-

Schema di cablaggio del rasaerba



re, perché non falciare il prato stando comodamente seduti all'ombra lontano dal tagliaerba? Da qui nasce l'idea di costruire un robot tagliaerba radiocomandato, usando il mitico e sempre attuale "Arduino" abbinato a una "manciatà" di elettronica. Il nostro macchinario è un ibrido con rasaerba a scoppio e trazione elettrica, nel senso che consta di un telaio con ruote in grado di sorreggere un tagliaerba con motore a scoppio, dotato dell'elettronica occorrente al controllo dei motori di trazione.

La scelta di montare un tagliaerba con motore a scoppio a bordo di un carrello a trazione elettrica nasce dal fatto che la maggior

parte degli italiani ne possiede già uno nel proprio capanno degli attrezzi, magari inattivo perché si è rotta una ruota o il manubrio.

Questo robot permette di ridare vita al rasaerba beneficiando della possibilità di comandarlo a distanza via radio, grazie alla trazione elettrica; infatti è una variante dello spazzaneve presentato nei fascicoli n° 191 e n° 192 di Elettronica In, del quale condivide il telecomando e la gestione dei motori.

LA MECCANICA

Il telaio che sorreggerà il tagliaerba a scoppio (Fig. 1) è realizzato con dello scatolare in ferro con

sezione quadra da 3x3 cm.

In un primo momento si era pensato di utilizzare dello scatolare in alluminio perché molto più leggero del ferro, ma la scelta del materiale è poi ricaduta su quest'ultimo in quanto sicuramente più semplice la saldatura, che infatti è stata fatta con una classica saldatrice ad elettrodo. Con l'alluminio, invece, avremmo dovuto avvitare o rivettare le parti data, appunto, la difficoltà di saldatura, correndo il rischio che con le vibrazioni portate dal motore del tagliaerba, viti e rivetti avrebbero potuto staccarsi. Il telaio è composto da un telaio con forma rettangolare avente misure esterne 48x150 cm, co-



Fig. 1

stituito da due barre lunghe di misura 150 cm le quali formano i lati lunghi del rettangolo, ed altre due che formano i lati corti interni, lunghi 42 cm.

Come si nota dalla Fig. 2, le barre corte sono state saldate 10 cm verso l'interno, così da riuscire ad effettuare una buona saldatura da entrambi i lati dello scatolato, sia interno che esterno. La doppia saldatura ha dato la possibilità di realizzare le maniglie per il trasporto ed il sollevamento.

Potete comunque variare le misure in base alla dimensione del vostro tosaerba, vi consigliamo però di non ridurre la dimensione dello scatolato per non

compromettere la robustezza del telaio.

La parte anteriore è costituita da due ruote pivotanti di diametro 20 cm con un'impronta larga 3 cm, che si è deciso di collocare sull'esterno del telaio, in modo da avere maggiore spazio per installare il tagliaerba.

Nella parte posteriore, invece, troviamo i due motori da 24V/100 W che si occupano della trazione del robot stesso.

Quelli utilizzati per il prototipo sono motori che vengono installati nelle carrozzine elettriche e sono stati abbinati ciascuno a una ruota del diametro di 26 cm con un battistrada largo 8 cm, ottimo per il terreno dove il robot verrà usato.

Tali motori sono in grado di spostare 150 kg e di raggiungere velocità di 6 km/h: sono decisamente sovradimensionati, per il nostro utilizzo, in quanto dovranno spostare solamente il peso del tagliaerba (circa 25 kg) e quello delle due batterie da 12 V (circa 18 kg l'una).

Fig. 3



È possibile utilizzare qualunque altro tipo di motori in continua purché siano da 24 V e abbiano una potenza adeguata; ad esempio si possono usare i medesimi del robot spazzaneve (Fig. 3) che troviamo sul sito www.futurashop.it (codice MY1016).

Per fissare il tagliaerba al telaio si è scelta una soluzione molto semplice, ovvero l'utilizzo di piattine in ferro di lunghezza tale da permettere alla lama di distaccarsi dal terreno di minimo 2,5 cm e massimo 6 cm, fissate direttamente al tagliaerba sfruttando fori già esistenti, come vediamo dalla Fig. 4. Comunque è possibile forare il piatto lama qualora i fori non fossero presenti.

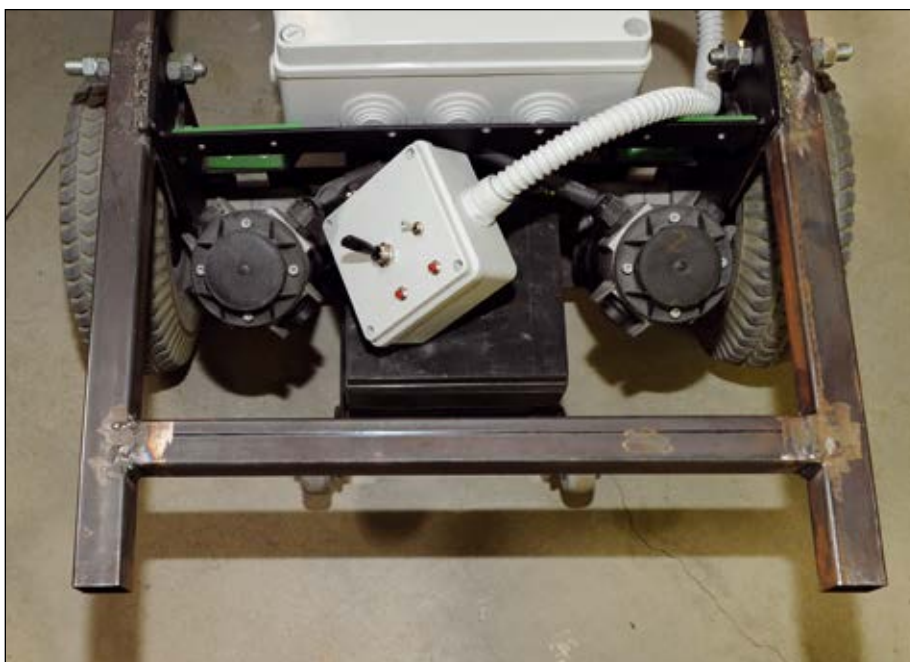


Fig. 2



Fig. 4

Ogni piattina viene fissata al telaio tramite dei bulloni del diametro di 8 mm, lunghi 6 cm. Le piattine sono state precedentemente forate ad una distanza di 1 cm dal bordo, dal lato del piatto lama, mentre nella parte superiore sono stati fatti più fori per usufruire di più possibilità di altezze di taglio.

Il consiglio è quello di fare questo lavoro per ultimo, prendendo le misure direttamente sul prato così da potervi rendere conto della reale altezza di taglio; nel nostro caso, sulle piattine abbiamo eseguito tre fori ad una distanza di 1,5 cm (riferita al centro del foro).

Abbiamo deciso di montare il tagliaerba senza l'utilizzo del sacco di raccolta, così da permettere al falciato di ricadere sul prato e creare un concime naturale.

Abbiamo lasciato il manico del tagliaerba così da non dover modificare il sistema della leva di sicurezza, dove arriva il cavo di comando del freno motore. Il tagliaerba utilizzato è un hit-line 400 b della al-ko (www.al-ko.it) possiede un motore a scoppio briggs & stratton XC 35 con potenza di 2 kW; la scocca è in acciaio e possiede un piatto lama di diametro 40 cm

L'ELETTRONICA

Il cuore del robot è Arduino: per l'esattezza, Arduino Uno Rev. 3

Fig. 5



(COD. ARDUINOUNOREV3); comunque è possibile impiegare altre versioni di Arduino con pin compatibili. Alla scheda Arduino sono abbinati, secondo lo schema visibile nelle pagine precedenti:

- uno Shield PS2;
- un driver motori DRI0018.

Entrambi si acquistano presso Futura Elettronica (www.futura-shop.it) insieme al resto del materiale elettrico ed elettronico occorrente al progetto.

Descriviamo l'elettronica partendo dalla fonte di alimentazione elettrica, che vede due batterie al piombo da 12 volt 26 Ah collegate in serie per fornire la tensione di 24 volt utile all'azionamento dei motori; da qui preleviamo anche i 12 volt che servono ad alimentare Arduino e la ventola da 80x80 mm che consigliamo di montare sotto al driver motori (applicatela alla scatola che conterrà il cablaggio, dopo aver aperto in essa una finestra di dimensioni adeguate) per evitare che il modulo surriscaldi.

Si è deciso di installare due interruttori separati per i 12 volt ed i 24 volt così da poter sezionare completamente le due tensioni, ma soprattutto per poter creare un circuito atto alla ricarica delle batterie: l'interruttore dei 24 volt, infatti, è in realtà un deviatore che ci permette di non dover staccare le batterie per la fase di ricarica, che avviene grazie ad un connettore da applicare al contenitore.

Passiamo ad Arduino e relativo shield: la scheda Arduino va sotto e su di essa si applica lo Shield PS2 tramite gli appositi pin strip; questo shield serve a decodificare il segnale ricevuto dal Ricevitore RX che le va inserito nell'apposito connettore e inviati dal controller della

PlayStation 2.

Dallo schema vedete che il connettore jack dell'alimentazione di Arduino è preceduto da un regolatore di tensione, un 7808 provvisto di dissipatore (basta uno ad "U" da 18 °C/W di resistenza termica) che permette di ricavare dalla linea dei 12 volt una tensione costante di 8 V.

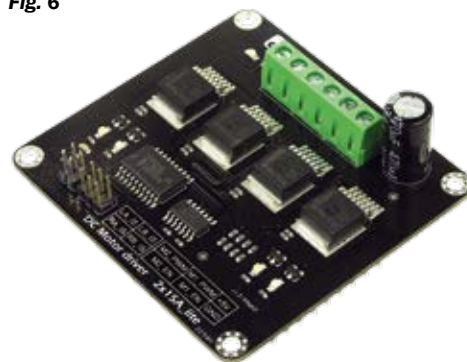
Si è deciso di limitare la tensione ad 8 volt per evitare di surriscaldare Arduino, in quanto, anche se dalle caratteristiche tecniche del costruttore sappiamo che può essere alimentata a 12V, dobbiamo ricordarci che una batteria da 12 volt si considera carica quando ha una tensione di 13,8 volt.

Lo shield PS2 (COD. PS2SHIELD, Fig. 5), tramite un'apposita libreria, ci permette di gestire tutti i pulsanti del controller della console Sony e di poter inviare anche un feedback sfruttando la vibrazione presente del controller.

Sulla parte visibile dello shield si nota subito il connettore dove inserire il ricevitore del controller. Dai connettori laterali dello shield, che ripetono i pin strip di Arduino, colleghiamo i quattro fili di controllo del driver motori; da essi preleviamo anche i 5 volt necessari molto importanti per alimentare la logica del driver.

Sullo shield sono inoltre pre-

Fig. 6



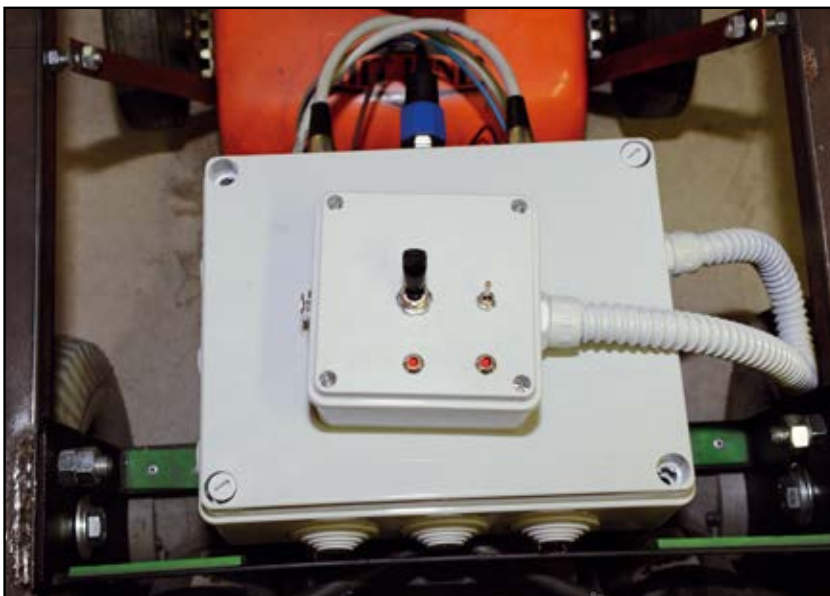


Fig. 8

senti dei jumper per selezionare i pin di TX ed RX di Arduino usati per la connessione seriale e la velocità di comunicazione tra Arduino e lo shield, che deve essere impostata a 9.600 bps; le linee TX ed RX da usare sono i pin 8 e 9; i jumper consentono di utilizzare vari pin in modo da evitare conflitti se eventuali ulteriori shield già utilizzano quelli desiderati.

Ricordiamo che se si decide di usare Arduino Mega, i pin di tx ed rx saranno rispettivamente 10 e 11.

Il driver di potenza (COD. DRI0018, Fig. 6) che ci permette di controllare i motori è appositamente studiato per l'utilizzo con Arduino: infatti sfruttando solo quattro pin, due per abili-

tare il motore corrispondente e due per il segnale PWM con cui pilotarlo, riusciamo a controllare i due motori da 24 volt.

Il driver motori è formato da un doppio ponte ad H ottenuto con due integrati BTS7960; sul modulo si trovano quattro di questi IC, connessi a due a due a ponte, che permettono di gestire due motori in corrente continua con tensione da 5V a 35V, che assorbano una corrente massima di 15A ciascuno.

Sul lato superiore dello stampato del driver motori troviamo inoltre quattro LED in SMD di colore blu e bianco, che ci indicano la direzione dei motori. Sulla faccia sottostante, invece, è presente un dissipatore applicato agli integrati BTS7960 per far

loro dissipare il calore.

Sotto al driver abbiamo voluto inserire una ventola di raffreddamento forzato, visto che il rasaerba verrà prevalentemente impiegato nel periodo estivo. Anche per il driver (come per lo shield PS2) il costruttore rilascia un'apposita libreria che permette, con soli 4 pin di Arduino, di controllare la direzione e la velocità di due motori; quindi sui pin strip del driver andiamo a collegare i pin di Arduino secondo questa corrispondenza:

- pin 4 di arduino - M1_EN del driver;
- pin 5 di arduino - M1_PWM del driver;
- pin 6 di arduino - M2_EN del driver;
- pin 7 di arduino - M2_PWM del driver.

Inoltre i contatti Vcc e GND del pin strip vanno a 5V e GND di Arduino. Per aiutarvi nelle connessioni ricordate che il driver riporta una serigrafia indicante il significato dei contatti.

In serie a ciascun motore è stato posto un fusibile rapido in vetro da 10 ampere.

Il tutto è stato poi messo all'interno di una scatola di derivazione utilizzata per impianti elettrici, delle dimensioni di 240x190x90 mm (Fig. 8), dove sono stati fissati tre connettori per il collegamento dei motori e dell'alimentazione; i due deviatori e il connettore per la ricarica sono stati inseriti in una seconda scatola, questa volta delle dimensioni di 80 x 80 x 40 mm.



I/O digitale Arduino pin DRI0018

4	DIR1
5	PWM1
6	PWM2
7	DIR2

Tabella 1

Listato 1

```
void loop(){
    motori=0;
    if( ps2.getval(p_up)==0 || ps2.getval(p_joy_lu)==100) { // se pulsante up è premuto o il joystick sinistro è in posizione up
        Serial.println("AVANTI"); //stampa sul monitor seriale
        analogWrite (E1,velo); //segnale PWM per controllo motori
        digitalWrite(M1,LOW); //
        analogWrite (E2,velo); //
        digitalWrite(M2,LOW); //
        motori=1;
    }
    if (ps2.getval(p_down)==0 || ps2.getval(p_joy_ld)==100) { // se pulsante down è premuto o il joystick sinistro è in posizione down
        Serial.println("INDIETRO"); //stampa sul monitor seriale
        analogWrite (E1,velo); //segnale PWM per controllo motori
        digitalWrite(M1,HIGH); //
        analogWrite (E2,velo); //
        digitalWrite(M2,HIGH); //
        motori=1;
    }
    if (ps2.getval(p_right)==0 || ps2.getval(p_joy_lr)==100){ // se pulsante right è premuto o il joystick sinistro è in posizione right
        Serial.println("DESTRA"); //stampa sul monitor seriale
        analogWrite (E1,velo); //segnale PWM per controllo motori
        digitalWrite(M1,LOW); //
        analogWrite (E2,velo); //
        digitalWrite(M2,HIGH); //
        motori=1;
    }
    if (ps2.getval(p_left)==0 || ps2.getval(p_joy_ll)==100) { // se pulsante left è premuto o il joystick sinistro è in posizione left
        Serial.println("SINISTRA"); //stampa sul monitor seriale
        analogWrite (E1,velo); //segnale PWM per controllo motori
        digitalWrite(M1,HIGH); //
        analogWrite (E2,velo); //
        digitalWrite(M2,LOW); //
        motori=1;
    }
    if (ps2.getval(p_triangle)==0) { // se il pulsante triangolo è premuto
        Serial.println("TRIANGOLO"); // stampa sul monitor seriale
        if (velo<250) // e se la variabile velocità (velo) è inferiore a 250
        {
            //
            velo=velo+25; // aumenta di 25 il valore della variabile velocità (velo)
        }
        //
        else //
        {
            //
            ps2.vibrate(2, 255); // attiva la vibrazione sul controller se il valore della variabile velocità (velo) è uguale a 250
            delay(500); // pausa 500 ms
            ps2.vibrate(2,0); // disattiva vibrazione
        }
        //
        analogWrite (M1, velo); //
        analogWrite (M2, velo); //
        Serial.println(velo);
    }
    if (ps2.getval(p_cross)==0) { // se premuto il pulsante "X"
        Serial.println("X"); // stampa sul monitor seriale
        if (velo>0 && velo<=250) // se la variabile velocità (velo) è maggiore di zero e inferiore a 250
        {
            //
            velo=velo-25; // diminuisci di 25 il valore della variabile velocità (velo)
        }
        //
        else
        {
            //
        }
    }
}
```

Per caricare le batterie basterà collegare il cavo di ricarica al connettore presente sulla scatola più piccola.

IL FIRMWARE

Terminato il montaggio ed il cablaggio, è possibile quindi collegare il nostro arduino alla presa USB del PC e caricare il firmware, ovvero lo sketch ne-

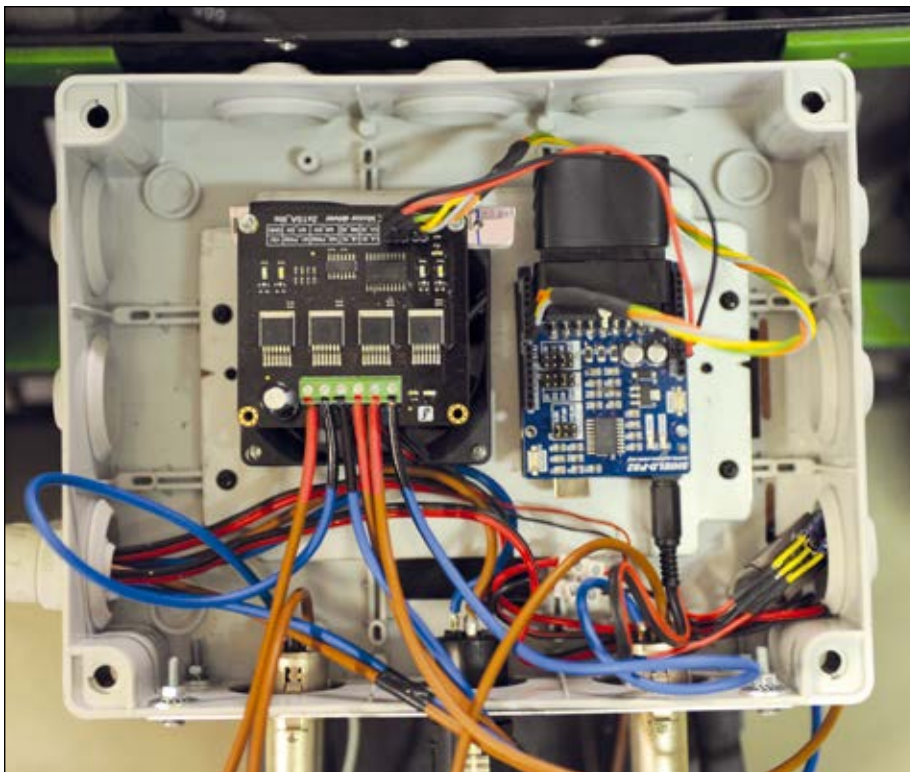
cessario affinché il nostro robot prenda vita. Ricordiamo però di staccare il ricevitore del joystick se usiamo i pin "D0" e "D1" come seriale hardware.

Il firmware è decisamente semplice: comprende infatti l'inclusione della libreria necessaria a far funzionare la shield PS2, che serve ad Arduino per riconoscere lo stato di tutti i tasti del

controller, come da **Tabella 1**.

Si trova poi la dichiarazione dei pin, che sono quelli già elencati a pagina precedente, inizializzati come output per gestire il driver motori. La parte di listato riguardante la dichiarazione dei pin è:

```
pinMode (E1, OUTPUT);
pinMode (M1, OUTPUT);
pinMode (E2, OUTPUT);
```



```
pinMode (M2, OUTPUT);
```

Mentre con i seguenti comandi:

```
Serial.begin(9600);  
ps2.init(9600, 8, 9)
```

impostiamo i pin 8 e 9 rispettivamente RX e TX tra Arduino e lo Shield PS2 ed una velocità di comunicazione di 9.600Bps, che, ricordiamo, deve essere impostata anche dai pin presenti sulla shield tramite gli appositi jumper. Nel **Listato 1**, vedete che il loop è composto da una serie di cicli "if" che permettono di riconoscere la pressione di uno dei tasti ed eseguire uno dei comandi ad esso associato.

Analizziamo nel dettaglio uno dei cicli, fermo restando che quanto detto vale per gli altri:

```
if (ps2.getval(p_right)==0 ||  
ps2.getval(p_joy_lr)==100) {  
Serial.println("DESTRA");  
analogWrite (E1,velo);  
digitalWrite(M1,LOW);  
analogWrite (E2,velo);
```

```
digitalWrite(M2,HIGH);  
motori=1;  
}
```

Il comando **ps2.getval(p_right)==0 || ps2.getval(p_joy_lr)==100** consente di riconoscere la pressione del tasto "freccia destra" o lo spostamento del joystick verso destra e all'avvenimento, con i comandi:

```
analogWrite (E1,velo);  
digitalWrite(M1,LOW);  
analogWrite (E2,velo);  
digitalWrite(M2,HIGH);  
motori=1;
```

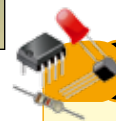
poniamo i motori uno avanti ed uno indietro così da permettere al robot di ruotare. Lo sketch comprende cinque variabili:

- "E1", "E2" corrispondono ai segnali PWM dei due motori;
- "M1", "M2" gestiscono l'attivazione dei motori stessi;
- "velo" gestisce il valore della velocità dei motori.

Si noterà che nel firmware è stata volutamente lasciata la stringa di

comando che permette di "stampare a monitor", così da poter visualizzare l'effettivo funzionamento di Arduino.

Vista quindi la semplicità del firmware, si può dire che è altrettanto semplice telecomandare il nostro robot: basterà infatti premere una volta il tasto "X" sul nostro controller per incrementare la velocità tramite la variabile "velo" oppure diminuirla tramite la pressione del tasto "triangolo"; per far muovere il robot rasoerba nella direzione desiderata sarà sufficiente muovere il controller nella stessa direzione. Buon divertimento! ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Gli accessori, i motori, e tutti gli altri elementi che compongono il rasoerba sono reperibili presso Futura Elettronica. Il firmware utilizzato per programmare il microcontrollore può essere scaricato dal sito della rivista. Il set shield PS2 per Arduino+controller PS2 (cod. SHIELDPS2RTX) è disponibile al prezzo di 46,00 Euro. La board Arduino Uno REV3 (cod. ARDUINOUNOREV3) costa 24,50 Euro. Il doppio driver per motori (cod. DRI0018) è disponibile al prezzo di 65 euro. I motori 24Vdc/250W 2650rpm (cod. MY1016-24VDC) sono disponibili al prezzo di 57,00 Euro ciascuno. I codici degli altri componenti sono indicati nello schema di cablaggio in corrispondenza di ogni prodotto.

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

RANDA PhotoSharing



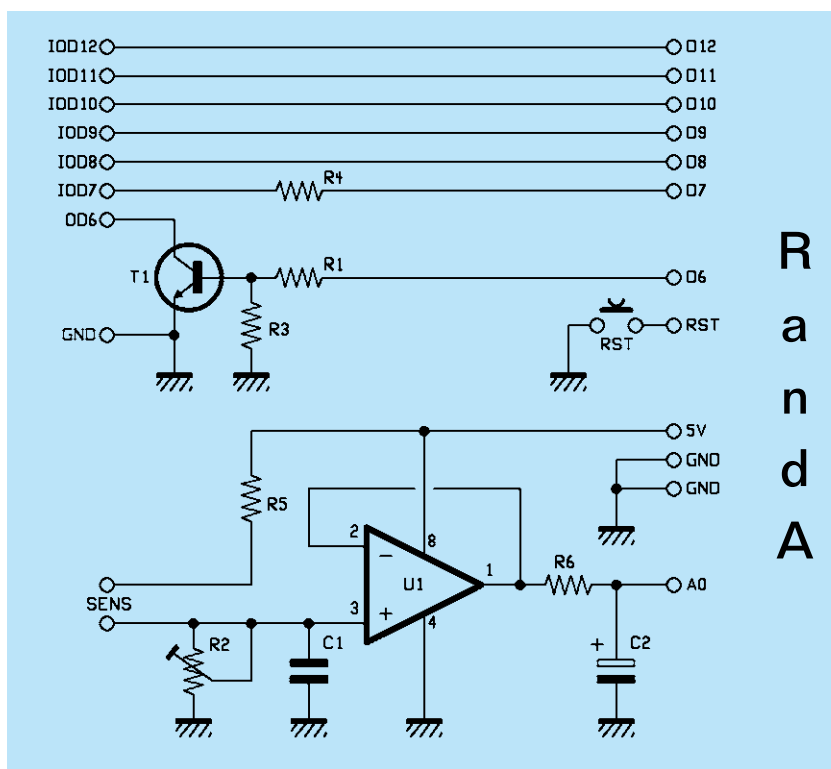
Scatta la foto, scegli l'effetto e condividila! Raspberry Pi 2 e RandA si uniscono per dare vita a un'applicazione interattiva che abbiamo presentato alla recente Maker Faire di San Mateo.



di LUCA e MARCO BELLAN

Alla Maker Faire tenutosi lo scorso maggio a San Mateo (California – U.S.A.), abbiamo presentato in anteprima un progetto nato dalla necessità di mostrare ai visitatori le potenzialità di RandA unita a Raspberry Pi, in modo suggestivo e divertente. Sono state più di cinquecento le foto scattate dai visitatori tramite RandA PhotoSharing -questo è il nome del

progetto- alcune delle quali condivise automaticamente sui social network di Open Electronics (Twitter e Facebook). In questo articolo vi spiegheremo passo dopo passo come assemblare e programmare il sistema, soffermandoci sulla configurazione di Raspberry Pi, per poi passare all'installazione e all'utilizzo del software RandA PhotoSharing. Il progetto che andre-

R
a
n
d
A

mo a realizzare sarà un misuratore multi-tematico nel quale potremo impostare, con un pizzico di empirica amenità, il tema della misurazione: amore, amicizia, intelligenza...

L'utente, attraverso diverse schermate, potrà scattarsi una foto (scegliendo eventualmente un effetto da applicarci), misurare la propria conducibilità elettrica (afferrando due poli conduttori) che sarà poi utilizzata per determinare un punteggio relativo al tema e visualizzare il risultato della misurazione posto in sovrapposizione nella foto precedentemente scattata (Fig. 1); potrà poi scegliere se condividere la foto sui social network oppure riceverla via e-mail. Questa applicazione può rivelarsi utile anche in termini di marketing, perché gli indirizzi e-mail inseriti dagli utenti per ricevere la propria foto vengono salvati in un database per poi essere esportati in un file Excel; allo stesso tempo la condivisione sui social network delle foto scattate contribuisce ad incrementare la propria visibilità sul web: infatti su ogni foto saran-

no presenti degli hashtag specifici e un logo personalizzabile.

L'utente interagirà con la nostra applicazione attraverso un monitor, una tastiera e tre pulsanti che serviranno a procedere nelle diverse schermate: inizio, scelta effetto foto, scatto foto e misurazione, condivisione sui social network, invio della foto appena scattata al proprio indirizzo e-mail e chiusura della sessione.

COMPONENTI E ASSEMBLAGGIO

Per realizzare questo progetto avremo bisogno dei seguenti componenti (reperibili su www.futurashop.it):

- Raspberry Pi 2, per eseguire l'applicazione con l'interfaccia grafica tramite un monitor, elaborare foto e inviarle;
- Raspberry Pi Camera, per scattare le foto agli utenti;
- SD-Card (consigliata da 8 GB o più), per il sistema operativo Raspbian e salvare le foto
- RandA;
- shield dedicato a questo progetto, per collegare i pulsanti e i poli conduttori a RandA;

- monitor con ingresso HDMI;
- mouse e tastiera USB;
- tre pulsanti, che avranno la funzione di Prev/Sì, OK, Next/No;
- due poli conduttori che i partecipanti al "gioco" dovranno impugnare per effettuare la misurazione;
- una connessione ad Internet con cavo ethernet oppure chiavetta WiFi.

Inoltre necessiteremo di un account Gmail e alcuni account di social network come Facebook e Twitter (opzionali) su cui la nostra applicazione posterà le foto. Per condividere le foto sui social network e salvarle in Google Drive utilizzeremo il sito www.ifttt.com; IFTTT è un servizio gratuito e molto comodo in quanto ci permette di impostare delle operazioni che verranno eseguite solo alla ricezione di una e-mail con determinati requisiti (recipes); per esempio, dopo aver impostato un "recipe" che carica una foto su Facebook prelevandola dall'allegato di una e-mail, possiamo inviare all'indirizzo trigger@recipe.ifttt.com una e-mail con un oggetto contenente "#photofb" e con allegata una foto: IFTTT preleverà in automatico l'allegato dall'e-mail e caricherà la foto sull'account Facebook precedentemente associato.

LO SHIELD ARDUINO

Per interconnettere a RandA e tramite essa a Raspberry Pi i tre pulsanti, abbiamo previsto un semplicissimo shield che realizza anche l'interfaccia verso il "sensore di emozioni"; con riferimento al relativo schema elettrico, illustrato in queste pagine, vediamo che i pulsanti si collegano ai pin IOD8, IOD9, IOD10, che portano alle linee rispettivamente D8, D9, D10 di RandA. I due elettrodi cilindrici che si impugnano per misurare le

emozioni vanno collegati ai contatti SENS, senza un preciso ordine: nello shield, uno porta all'uscita 5V di RandA e l'altro all'ingresso di un buffer realizzato dall'amplificatore operazionale U1, l'uscita del quale termina sulla linea A0 di RandA. Notate che gli elettrodi permettono di inserire il corpo dell'utente in un circuito partitore di tensione che pilota il buffer: tramite la resistenza R5, la linea 5V alimenta una mano attraverso l'apposito elettrodo cilindrico, quindi la debole e -teniamo a precisarlo a scanso di equivoci- innocua corrente che fluisce nel corpo, termina sul secondo elettrodo cilindrico e quindi trimmer R2, determinando all'ingresso non-invertente dell'operazionale un potenziale che è funzione sia del valore resistivo inserito con R2 (che permette di tarare la sensibilità del sistema), sia della conducibilità elettrica del corpo di chi tocca gli elettrodi. Tale conducibilità varia, nello stesso soggetto, in base alla sudorazione, alle emozioni ecc.

Per questo la tensione applicata all'operazionale e restituita di pari valore (ben filtrata dalla cella passa-basso R6/C2 per trattenere i disturbi impulsivi ed evitare

brusche variazioni dovute all'assestamento delle mani sui contatti) alla linea A0 di RandA, rispecchia lo stato d'animo dell'utente collegato. Sullo shield prende posto un pulsante che permette di resettare RandA e si trova anche un transistor montato a emettitore comune in connessione open-collector, pilotato in base dalla linea D6 di RandA e il cui collettore è connesso al contatto OD6; tale NPN può essere usato per attivare eventuali LED, avvisatori acustici ecc., ma al momento è "spare", vale a dire di scorta e perciò non utilizzato.

REALIZZAZIONE PRATICA

Per assemblare il tutto, colleghiamo la Raspberry Pi Camera facendo attenzione che la parte

blu del flat sia rivolta verso la porta ethernet di Raspberry Pi, montiamo RandA con il relativo supporto e successivamente montiamo la shield di RandA PhotoSharing sopra a RandA. Colleghiamo ai seguenti pin dello shield i tre pulsanti: pin 8 per il pulsante che avrà la funzione "Next/No", pin 9 per il pulsante che avrà la funzione "OK" e pin 10 per il pulsante che avrà la funzione "Prev/Yes"; i tre pulsanti avranno un estremo in comune, connesso a GND di RandA. Cabliamo un polo conduttore all'uscita 5V e l'altro all'ingresso A0 di RandA; successivamente colleghiamo mouse, tastiera, cavo ethernet o chiavetta WiFi (necessari per la parte di configurazione). Dopo aver preparato la SD-Card e configurato Raspbian (spiegheremo nel paragrafo successivo come fare), alimentiamo con 5V/2A RandA, che a sua volta alimenterà Raspberry Pi.

CONFIGURAZIONE

Possiamo scaricare l'immagine per Raspberry Pi già pronta per questo progetto dal sito di Elettronica In e passare direttamente alla fase di caricamento dello sketch su RandA, oppure possiamo scarica-

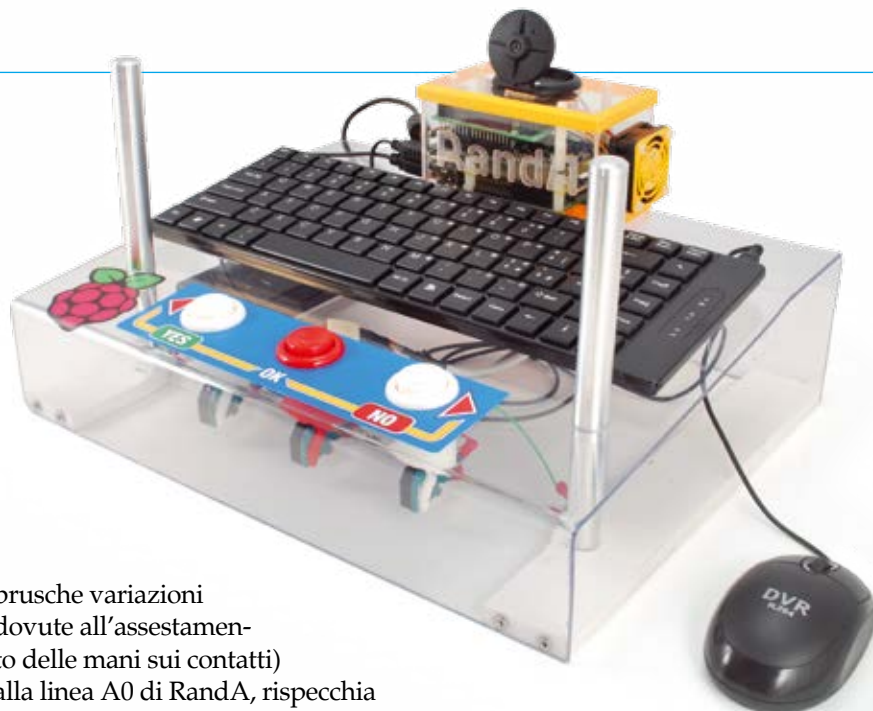
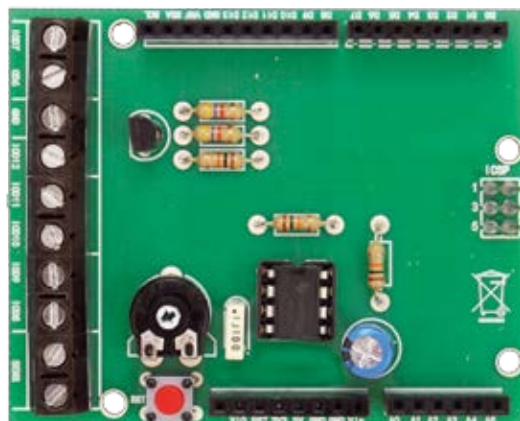
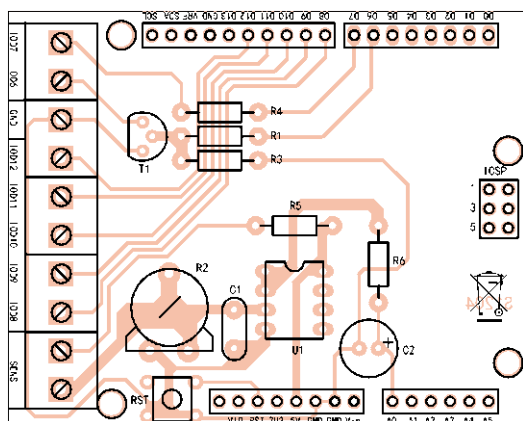


Fig. 1 - Il prototipo di RandA PhotoSharing.



Elenco Componenti:

R1: 4,7 kohm
R2: Trimmer 220 kohm MO
R3, R5: 10 kohm
R4: 470 ohm
T1: BC547
C1: 100 nF ceramico

C2: 100 μ F 16VL elettrolitico
U1: LM358
RST: Microswitch

Varie:
- Zoccolo 4+4

- Strip Maschio/Femmina 6 vie
- Strip Maschio/Femmina 8 vie (2 pz.)
- Strip Maschio/Femmina 10 vie
- Strip Maschio/Femmina 2x3 vie
- Circuito stampato S1204

re dal sito ufficiale di Raspberry Pi l'immagine dell'ultima versione di Raspbian e seguire tutta la procedura di configurazione.

Nel caso scegliessimo di partire con l'immagine già pronta, dovremo ricordarci di modificare il file `/etc/ssh/ssmtp.conf` come indicato nella sezione di configurazione delle librerie per l'invio dell'e-mail. Scriviamo l'immagine di Raspbian su SD Card con il software Win32DiskImager reperibile gratuitamente a questo indirizzo: sourceforge.net/projects/win32diskimager/. Inseriamo poi l'SD-Card nella slot di Raspberry Pi e alimentiamo RandA per accendere il tutto. I successivi passaggi della configurazione potranno essere eseguiti sia tramite l'ambiente grafico di Raspberry Pi, sia da remoto in SSH con client come PuTTY.

Durante la prima accensione sarà visualizzato il tool di configurazione di Raspberry Pi, con cui dovremo compiere diverse operazioni:

1. Expand Filesystem;
2. Enable Boot to Desktop/Scratch:

selezioniamo la seconda voce "Desktop Log in...":

3. Internalisation Options:

- a. Change Locale: con la barra spaziatrice deseleggiamo `en_GB.UTF-8 UTF-8` e selezioniamo invece `it_IT ISO-8859-1` (settandola successivamente come *Default locale for the system environment*);
- b. Change Timezone: selezioniamo *Europe* e successivamente *Rome*;
- c. Change Keyboard Layout: scegliamo il modello di tastiera che stiamo utilizzando (solitamente *Generica 105 tasti (Intl) PC*), successivamente selezioniamo *Altro* e poi *Italiana* come lingua della tastiera, scegliamo di nuovo *Italiana* come disposizione della tastiera e infine confermiamo cliccando su *Ok* due volte;

4. Enable Camera: *Enable*;

5. Advanced Options:

- a. SSH: *Enable*;
- b. PC: rispondiamo *Yes* a entrambe le domande.

Spostiamoci con il tasto TAB sulla

scritta *<Finish>* e confermiamo con *INVIO* per riavviare il sistema. A questo punto dovremo disporre di una connessione ad Internet per procedere: se disponiamo di un cavo ethernet colleghiamolo a Raspberry Pi, altrimenti seguiamo la procedura contenuta nel riquadro "Installazione e configurazione chiavetta WiFi per Raspberry Pi" che trovate nelle prossime pagine. Dopo aver connesso Raspberry Pi ad internet, lanciamo il seguente comando per aggiornare il sistema e, infine, riavviarlo:

```
sudo apt-get update && sudo
apt-get upgrade && sudo rpi-
update && sudo reboot
```

La procedura di aggiornamento potrà richiedere molto tempo e a volte dovremo confermare l'installazione di alcuni pacchetti premendo S e poi INVIO. Scarichiamo ora l'ultima versione del setup di RandA dal sito <https://github.com/open-electronics/RandA/releases>. Una volta scaricato il pac-

chetto, al suo interno troveremo il manuale da seguire per installare RandA (*InstallationREADME.TXT*) e i relativi file di installazione. Dopo aver installato RandA, dovremo cambiare la porta di ascolto del servizio Tomcat per evitare conflitti con il servizio Apache che installeremo successivamente. Infatti Tomcat è utilizzato per "hostare" il sito del pannello di controllo in Java di RandA, che a noi non servirà; dovremo invece utilizzare Apache per "hostare" il sito del pannello di controllo della nostra applicazione, in modo che, digitando l'indirizzo IP del nostro Raspberry Pi su qualsiasi browser, verremo rediretti al pannello di controllo di RandA PhotoSharing. Spostiamoci quindi nella directory dove risiede la configurazione di Tomcat:

```
cd /home/apache-tomcat-7.0.47/
conf
```

Apriamo il file `server.xml` con il comando:

```
nano server.xml
```

Editiamo la riga "`<Connector port = "80" ...`" sostituendo 80 con 8080. Adesso chiudiamo e salviamo il file con CTRL+X, Y e INVIO; successivamente riavviamo Raspberry Pi:

```
sudo reboot
```

Disattiviamo ora il risparmio energetico di Raspberry Pi editando questo file:

```
sudo nano /etc/lightdm/lightdm.conf
```

Aggiungiamo la seguente riga sotto alla sezione [SeatDefaults]:

```
xserver-command=X -s 0 dpms
```

Chiudiamo e salviamo il file con

CTRL+X, Y e INVIO: in questo modo Raspberry Pi non disattiverà l'output dell'HDMI dopo alcuni minuti di inattività.

Lanciamo i seguenti comandi in successione per installare Apache, PHP e il server MySQL che ci serviranno per far funzionare il pannello di controllo Web di RandA PhotoSharing:

```
sudo apt-get install apache2
apache2-doc apache2-utils
sudo apt-get install libapache2-
mod-php5 php5 php-pear php5-
xcache
sudo apt-get install php5-mysql
sudo apt-get install mysql-ser-
ver mysql-client
```

Durante l'installazione di alcuni pacchetti verrà richiesta la conferma, premere S e INVIO per confermare. Quando ci sarà chiesta una password per il database inseriamola e teniamola a portata di mano poiché ci sarà richiesta in seguito. Ora installiamo PHPMyAdmin che

ci aiuterà a gestire il database di RandA PhotoSharing:

```
sudo apt-get install phpmyadmin
```

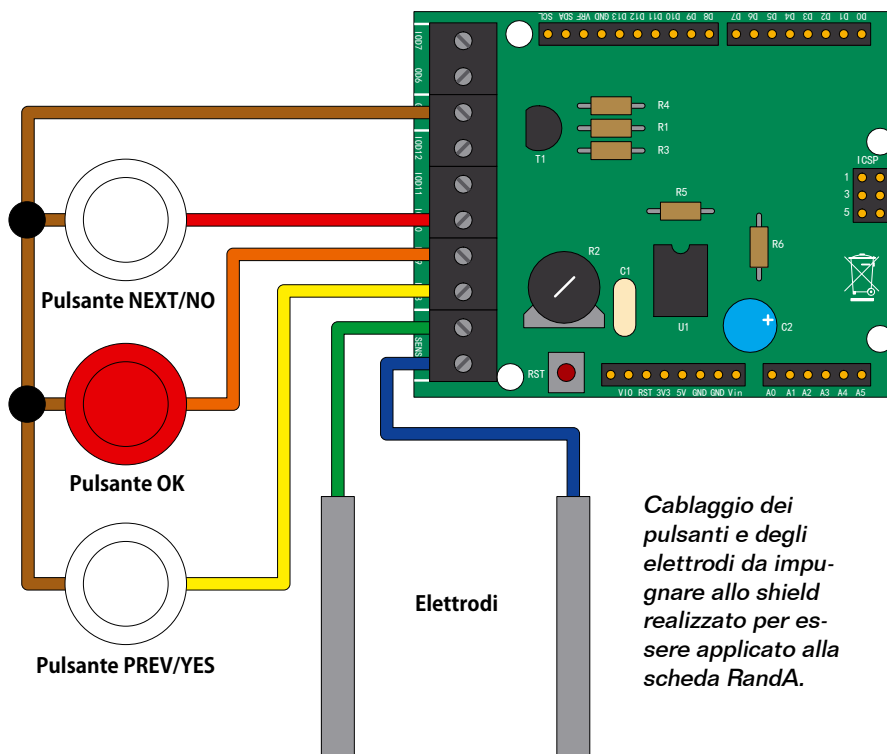
Quando ci verrà chiesto quale Web Server configurare automaticamente, con lo SPAZIO selezioniamo "apache2", poi confermiamo premendo TAB e INVIO. Alla domanda "Configurare il database di PHPMyAdmin con dbconfig-common?" rispondiamo "Sì". Terminata l'installazione, apriamo il file di configurazione per apache2:

```
sudo nano /etc/apache2/apache2.conf
```

e aggiungiamo al termine di esso la riga:

```
Include /etc/phpmyadmin/apache.conf
```

Chiudiamo e salviamo il file con CTRL+X, Y e INVIO; riavviamo



apache2 per rendere effettive le modifiche:

```
sudo /etc/init.d/apache2 restart
```

Lanciamo il comando:

```
sudo visudo
```

e aggiungiamo alla fine del file la seguente stringa:

```
www-data ALL=(ALL) NOPASSWD: ALL
```

Chiudiamo e salviamo il file con CTRL+X, Y e INVIO: in questo modo daremo i permessi ai file PHP per agire sulle foto scattate

(lettura, modifica e cancellazione). Passiamo ora ad installare le librerie che serviranno al nostro script Python (il programma vero e proprio) per interagire con il database MySQL e modificare le immagini. Lanciamo quindi i seguenti comandi in successione:

```
sudo apt-get install python-dev
sudo apt-get install python-imaging-tk
sudo apt-get install python-mysqldb
sudo apt-get install imagemagick
```

Installiamo il mail server e le utility che ci serviranno per inviare e-mail da Python lanciando i seguenti comandi in successione:

```
sudo apt-get install ssmtp
sudo apt-get install mailutils
```

Modifichiamo ora il file *ssmtp.conf* con il comando:

```
sudo nano /etc/ssmtp/ssmtp.conf
```

in modo che risulti come segue:

```
root=postmaster
mailhub=smtp.gmail.com:587
hostname=Raspberry Pi Pipi
AuthUser=UtenteGmail@gmail.com
AuthPass=PasswordGmail
UseSTARTTLS=YES
```

Ricordiamoci di sostituire "UtenteGmail" e "PasswordGmail" con i dati del nostro account, chiudiamo e salviamo il file con CTRL+X, Y e INVIO.

Modifichiamo il file *revalias*:

```
sudo nano /etc/ssmtp/revalias
```

Aggiungendo ora la riga seguente:

```
root:root@gmail:smtp.gmail.com:587
```

Chiudiamo e salviamo il file con

CTRL+X, Y e INVIO. Cambiamo il permesso del file *ssmtp.conf* con il seguente comando:

```
sudo chmod 774 /etc/ssmtp/ssmtp.conf
```

Riavviamo il sistema per rendere effettive tutte le modifiche:

```
sudo reboot
```

Abbiamo completato la configurazione di Raspberry Pi installando e configurando tutto il necessario, ora dobbiamo scaricare il pacchetto di RandA PhotoSharing; posizioniamoci quindi sotto alla cartella corretta:

```
cd /var/www
```

e scarichiamo il pacchetto di RandA PhotoSharing con il comando:

```
sudo git clone https://github.com/open-electronics/randaps.git
```

La cartella che verrà creata (riferitevi alla Fig. 2) conterrà:

- **admin/**: cartella che contiene i file del pannello di controllo Web;
- **data/**: cartella che contiene tutte le grafiche necessarie al funzionamento (temi, overlay, logo, ...);
- **photos/**: cartella che ospiterà tutte le foto che verranno scattate;
- **randaps.py**: eseguibile di RandA PhotoSharing;
- **randaps_sketch.hex**: sorgente compilato per RandA pronto per essere caricato;
- **randaps_sketch.ino**: sorgente per RandA PhotoSharing, nel caso in cui volessimo effettuare delle modifiche;
- **RandA-PhotoSharing.sql**: installazione del database MySQL;
- **README.pdf**: guida completa per il setup e l'utilizzo di RandA PhotoSharing (traduzione di



Installazione e configurazione chiavetta WiFi per Raspberry Pi

questo articolo in inglese).

Accediamo, con un browser qualunque, all'indirizzo `http://IP_RASPBERRY_PI/phpmyadmin` avendo cura di sostituire la dicitura "IP_RASPBERRY_PI" con l'indirizzo IP effettivo. Facciamo il login utilizzando come utente root e come password quella che abbiamo impostato durante l'installazione di MySQL e PHPMyAdmin.

Spostiamoci sulla tab *Privilegi*, facciamo clic su *Aggiungi un nuovo utente* e compiliamo i campi come indicato qui di seguito:

- Utente: [Utilizza campo text:] *randaps*;
- Host: [Locale] *localhost*;
- Password: [Utilizza campo text:] *randaps*;
- Reinserisci: *randaps*;
- Database per l'utente: *creare un database con lo stesso nome e concedi tutti i privilegi*;
- Privilegi globali: *selezionare tutti*.

Adesso facciamo clic sul pulsante *Crea utente*: sul lato sinistro della pagina dovremmo vedere comparire il database *randaps* (se così non fosse aggiorniamo la pagina); successivamente facciamo clic sul nome del database appena creato e selezioniamo la tab *Importa*. Dobbiamo importare il file SQL contenuto del pacchetto di installazione di RandA PhotoSharing; scegliamo quindi il file e facciamo click su *Esegui*. Editiamo il file eseguibile di RandA PhotoSharing:

```
sudo nano /var/www/randaps/randaps.py
```

Nella sezione "CUSTOMIZABLE VARIABLES" inseriamo la password del nostro account di Gmail al posto della scritta "INSERT_YOUR_E-MAIL_PASSWORD", facendo attenzione a mantenere le virgolette iniziali e finali; salviamo e chiudiamo il file con CTRL+X, Y

In alternativa alla connessione ethernet, è possibile impiegare una soluzione wireless affacciando a Internet il sistema mediante WiFi. Allo scopo bisogna procurarsi un dongle WiFi USB e inserirlo nella presa USB di Raspberry Pi, apriamo il terminale e digitiamo il comando:

```
dmesg | more
```

Controlliamo che Raspberry Pi abbia riconosciuto la periferica: dobbiamo premere SPAZIO fino a quando non troviamo una riga simile a "Product: 802.11n WLAN Adapter ..." e per uscire premiamo il tasto Q. Editiamo il file delle interfacce di rete con il comando:

```
sudo nano /etc/network/interfaces
```

Inseriamo le righe per la connessione alla rete WiFi; alla fine il file dovrebbe risultare così:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0

iface wlan0 inet dhcp
    wpa-ssid "Nome della rete WiFi"
    wpa-psk "Password della rete WiFi"
```

Chiudiamo e salviamo il file con la combinazione di tasti CTRL+X, Y e INVIO; successivamente lanciamo il seguente comando per ricaricare le impostazioni di rete:

```
sudo service networking reload
```

Con il comando `ifconfig` possiamo verificare che il dongle WiFi sia connesso e abbia ottenuto un indirizzo IP. La configurazione è terminata.

e INVIO.

Carichiamo ora lo sketch su RandA con il seguente comando:

```
ArduLoad /var/www/randaps/randaps_sketch.hex
```

Arrivati a questo punto, passiamo a configurare gli account che serviranno a RandA PhotoSharing per inviare le foto:

- **Gmail:** RandA PhotoSharing utilizzerà questo account per spedire le foto a IFTTT e all'utente (se richiesto);
- **IFTTT:** ogni volta che riceverà una mail (dall'account di Gmail) provvederà a caricarla su Google Drive (se impostato) o condividerla su Facebook o Twitter (se impostato).

Accediamo con il nostro account a Gmail, visitiamo il seguente indirizzo: `https://www.google.com/settings/security/lesssecureapps` e abilitiamo l'accesso per le app meno

sicure: in questo modo RandA PhotoSharing potrà inviare e-mail a nome di questo account. Creiamo un account IFTTT (`www.ifttt.com`) e registriamo i seguenti recipes:

- Salva le foto su Google Drive: `https://ifttt.com/recipes/192360-send-gmail-attachments-to-google-drive`;
- Carica le foto su Twitter: `https://ifttt.com/recipes/129743-twitter-`

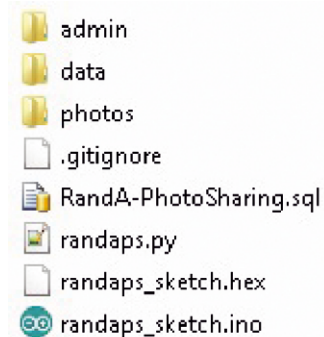


Fig. 2 - Disposizione dei file nella cartella *randaps*.

Control Panel

PHOTOS
SETTINGS

Theme

Maker
▼

Add new theme

Value
▼

Standby

Start screen
▼

Logo

X
750

Y
655

Overlay

X
10

Y
520

Result

X
110

Y
655

Photos

Off

On

Off

Off

On

On

Save to SD card

Save to cloud

Send to social

Send via eMail

Fig. 3
Il pannello di controllo.

subject-w-tv-body-tweet-attachment-photo;

- Carica le foto su Facebook: <https://ifttt.com/recipes/13714-upload-photo-to-facebook-from-e-mail>.

PANNELLO DI CONTROLLO

Per accedere al pannello di controllo da qualsiasi dispositivo connesso alla stessa rete di RandA PhotoSharing occorre digitare: http://IP_RASPBERRY_PI/randaps/admin. Avuto l'accesso, inseriamo come utente "admin" e come password "randaps". La pagina si divide in due sezioni: la prima elenca tutte le foto scattate (se facciamo clic sulla data/ora le possiamo visualizzare), se l'utente ha dato o meno il consenso per i social, e l'eventuale e-mail inserita; possiamo anche generare un file Excel con tutti i dati raccolti, op-

pure cancellare tutti i dati (sia dal database che nella cartella "photos") tramite i due pulsanti nella parte inferiore della tabella. La seconda sezione, come mostrato in Fig. 3, contiene tutte le impostazioni di RandA PhotoSharing: dopo averle modificate occorre riavviare il programma, oppure attendere che venga caricata la Start screen (schermata iniziale). Di seguito analizziamo le impostazioni:

- **Theme:** il tema della misurazione (grado di maker, amore, intelligenza, amicizia ecc.);
- **Add new theme:** consente di aggiungere un nuovo tema; nel campo Value dovremo scrivere il nome fisico del tema (ad esempio "love") mentre nel campo Description potremo scrivere la descrizione del tema (esempio: Love Meter); dopo aver aggiun-

to un nuovo tema comparirà un messaggio che ci ricorderà di posizionare i file immagine del tema sotto alla cartella data;

- **Standby:** la prima schermata mostra l'immagine iniziale del tema oppure uno slideshow di tutte le foto scattate fino al momento in cui è visualizzata;
- **Coordinate elementi:** coordinate X e Y in pixel degli elementi che verranno posizionati sulla foto (logo, overlay, result); potremo agire a piacimento su questi valori per cambiare la disposizione dei tre elementi nella foto, tenendo presente che la foto sarà sempre scattata nel formato 1280x720 (a meno che non si modifichi la relativa variabile di configurazione contenuta in randaps.py) e l'origine degli elemento (X=0 e Y=0) è l'angolo superiore sinistro della foto;
- **Save to SD card:** se attivato, lascia le foto nella cartella "photos", altrimenti le cancella;
- **Save to cloud:** se attivato, salva una copia delle foto in Google Drive;
- **Send to social:** se attivato, chiede all'utente se può caricare la foto su Facebook/Twitter (in base a configurazione IFTTT);
- **Send via e-mail:** se attivato, chiede all'utente se vuole ricevere la sua foto via e-mail;
- **Sender e-mail:** l'e-mail del nostro account Gmail che si occuperà di inviare le foto scattate all'utente e a IFTTT;
- **e-mail object:** oggetto dell'e-mail che verrà inviata;
- **IFTTT object:** in questo campo possiamo inserire i tag che serviranno a IFTTT per caricare le foto sui social network;
- **e-mail body:** editor HTML che permette di creare il corpo dell'e-mail che verrà inviata;
- **text_start:** testo che verrà mostrato nella Start screen;
- **text_preview:** testo che verrà

- mostrato durante la scelta dell'effetto da applicare alla foto;
- **text_measurement**: testo che verrà mostrato durante la misurazione dell'emozione;
 - **text_photo**: testo che viene mostrato quando viene scattata la foto;
 - **text_wait**: testo che invita ad attendere dopo che viene scattata la foto;
 - **text_social**: testo con cui possiamo chiedere all'utente se vuole caricare la foto appena scattata sui nostri social network;
 - **text_email**: testo con cui possiamo chiedere all'utente se vuole ricevere la foto appena scattata al proprio indirizzo e-mail;
 - **text_yes**: traduzione della parola "Sì";
 - **test_no**: traduzione della parola "No";
 - **text_end**: verrà mostrato nella schermata Foto risultato (se sono disattivate entrambi le schermate Social ed e-mail) per invitare l'utente a premere OK per concludere.

PERSONALIZZAZIONE

Le immagini che verranno applicate in sovraimpressione ad ogni foto scattata sono contenute nella cartella "data"/; quindi ogni volta che creiamo un nuovo tema dal pannello di controllo dovremo ricordarci di creare le seguenti immagini:

- **[tema]_screen.gif**: immagine GIF (1920x1080) che verrà ridimensionata in base alla risoluzione corrente dello schermo e caricata nello Start screen secondo il tema impostato (ad esempio *love_screen.gif*);
- **[tema]_overlay.png**: immagine PNG (dimensioni libere) che rappresenterà il tema impostato in modo stilizzato (ad esempio *love_end.gif*);
- **[tema]_end.gif**: immagine GIF (1920x1080) che verrà ridimensionata in base alla risoluzione dello schermo e mostrata alla fine dell'esecuzione (ad esempio *love_overlay.png*).

Il file **logo.png** è un'immagine PNG (dimensioni libere) che viene applicata alla foto una volta scattata e non cambia rispetto al tema. Potremo personalizzare ulteriormente l'applicazione agendo sul file *randaps.py* nella sezione "CUSTOMIZABLE VARIABLES". Per farlo, apriamo il file con il comando:

```
sudo nano /var/www/randaps/randaps.py
```

Una volta completate le modifiche è sufficiente chiuderlo e salvarlo con CTRL+X, Y e INVIO. Per aggiungere o modificare un utente del pannello di con-

trollo web, dobbiamo utilizzare PHPMyAdmin: accediamo all'indirizzo http://IP_RASPBERRY_PI/phpmyadmin (sostituendo "IP_RASPBERRY_PI" con l'indirizzo IP effettivo), selezioniamo il database "randaps" e, successivamente, la tabella "users". Facciamo clic sulla tab "Inserisci", compiliamo il campo "user" e il campo "password"; il campo "password" è codificato in MD5. Per codificare qualsiasi stringa di testo, basta utilizzare uno dei numerosi servizi di codifica MD5 disponibili sul web.

FUNZIONAMENTO

Per lanciare RandA PhotoSharing basta aprire il terminale dal desktop grafico di Raspberry Pi (non in SSH) e scrivere il comando:

```
sudo python /var/www/randaps/randaps.py
```



Listato 1

```
//Inizializzo pin pulsanti
byte pinNext=8;
byte pinOk=9;
byte pinPrev=10;

bool dx=false;
bool sx=false;
bool ok=false;

bool dxTX=false;
bool sxTX=false;
bool okTX=false;

unsigned long value=0;
unsigned int measurements=0;

void setup() {
  //Apro comunicazione seriale con RandA
  Serial.begin(9600);
  pinMode(pinNext, INPUT);
  pinMode(pinPrev, INPUT);
  pinMode(pinOk, INPUT);
  pinMode(A0, INPUT);
  //Abilito resistenze pull-up
  digitalWrite(pinNext, HIGH);
  digitalWrite(pinPrev, HIGH);
  digitalWrite(pinOk, HIGH);
}

void loop(){
  char input = Serial.read();
  if(input=='1'){
    //Se ricevo 1 da RandA: restituisco pulsante premuto
    if (dx && !dxTX){
      dx=false;
      dxTX=true;
      Serial.println("1");
    }else if (sx && !sxTX) {
      sx=false;
      sxTX=true;
      Serial.println("0");
    }else if (ok && !okTX) {
      ok=false;
      okTX=true;
      Serial.println("2");
    }else{
      Serial.println("9");
    }
  }else if(input=='2'){
    //Se ricevo 2 da RandA: restituisco misurazione dai poli
    measureIN();
    int average= value/measurements;
    Serial.println(average);
    value=0;
    measurements=0;
  }
  buttons();
  delay(10);
}

void buttons(){
  if(digitalRead(pinNext)==LOW) dxTX=false;
  if(digitalRead(pinNext)==HIGH && !dxTX) dx=true;

  if(digitalRead(pinPrev)==LOW) sxTX=false;
  if(digitalRead(pinPrev)==HIGH && !sxTX) sx=true;

  if(digitalRead(pinOk)==LOW) okTX=false;
  if(digitalRead(pinOk)==HIGH && !okTX) ok=true;
}

void measureIN(){
  //Leggo per 10 volte input analogico per misurare il voltaggio
  for ( measurements=0; measurements<10; measurements++){
    int val;
    val=analogRead(A0);
    value+=val;
    delay(20);
  }
}
```

Il primo avvio potrebbe richiedere diversi secondi perchè il software sta ridimensionando tutte le immagini contenute nella cartella "data" in base alla risoluzione corrente; per chiudere RandA PhotoSharing basta premere F11 (abilita/disabilita modalità tutto schermo) e fare clic sulla X in alto a destra.

Le schermate che appariranno (in sequenza) sono le seguenti:

1. **Start screen:** visualizza l'immagine iniziale del tema oppure uno slideshow delle foto fatte (in base alle impostazioni) [Attende OK per andare avanti];
2. **Preview:** visualizza immagine della Raspberry Pi Camera, con i bottoni Prev e Next si scorrono gli effetti [Attende OK per andare avanti];
3. **Misurazione:** invita a toccare i due poli (5V e A0) e intanto mostra un conto alla rovescia alla fine del quale scatta la foto, infine mostra un invito ad attendere mentre la foto viene elaborata;
4. **Social:** se attivata, questa schermata mostra la foto risultato e chiede se la si vuole caricare sui social network (collegati a IFTTT) [Attende Prev o Next, che in questo caso valgono come Sì e No];
5. **e-mail:** se attivata, questa schermata mostra la foto risultato e chiede se si vuole ricevere la versione in alta definizione via e-mail offrendo una textbox dove inserirla [Attende la pressione di uno dei tre pulsanti per andare avanti];
6. **Foto risultato:** se sia Social che e-mail (punti 4 e 5) non sono attivati, questa schermata mostra semplicemente la foto risultato [Attende OK per andare avanti];
7. **End screen:** ringrazia l'utente tramite un'immagine personalizzabile (intanto in background

si occupa di inviare l'e-mail e condividere la foto sui social network).

A questo punto il programma ritorna allo Start screen (punto 1) caricando i settaggi del pannello di controllo e applicando le eventuali modifiche alle impostazioni. Per "foto risultato" si intende la foto che RandA PhotoSharing ha scattato durante la misurazione, con l'effetto che abbiamo selezionato, un eventuale logo, e l'immagine di overlay relativa al tema impostato, con all'interno il risultato della misurazione effettuata: per esempio, se il tema

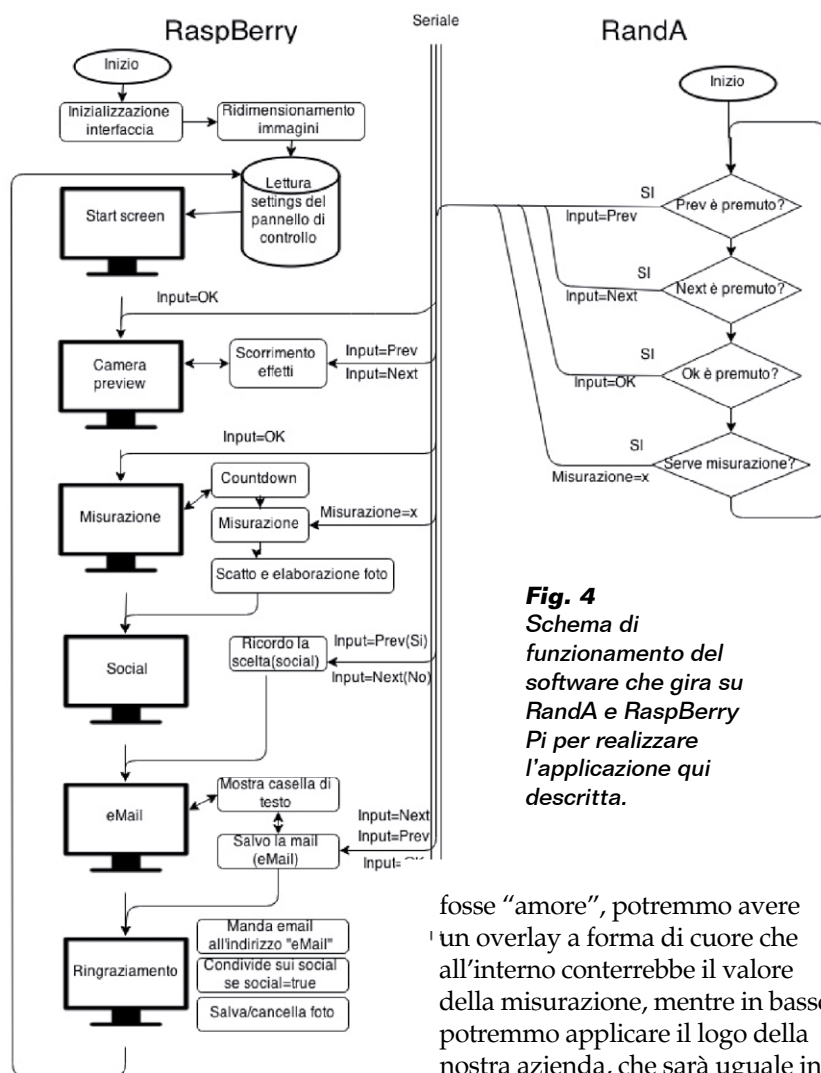
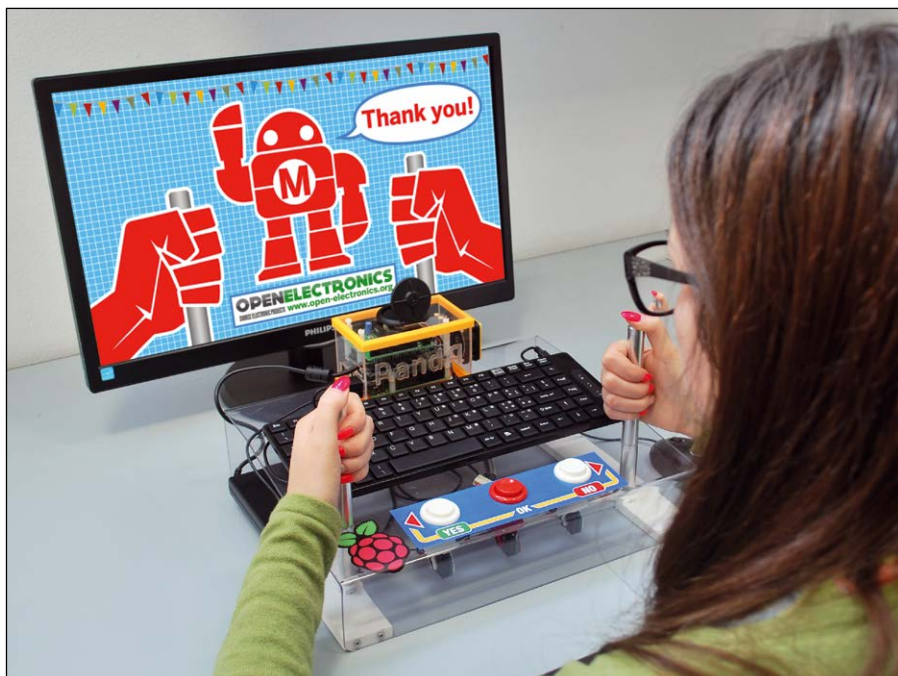
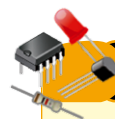


Fig. 4
Schema di funzionamento del software che gira su RandA e RaspBerry Pi per realizzare l'applicazione qui descritta.

fosse "amore", potremmo avere un overlay a forma di cuore che all'interno conterrebbe il valore della misurazione, mentre in basso potremmo applicare il logo della nostra azienda, che sarà uguale in

tutte le foto.

Possiamo vedere lo schema di funzionamento del software in Fig. 4. Nel Listato1 trovate lo sketch da caricare in RandA, che si occuperà di ritornare a RaspBerry Pi i pulsanti premuti e la tensione misurata. Buon divertimento!



per il MATERIALE

Tutti i componenti sono di facile reperibilità e posso essere acquistati presso Futura Elettronica. Il master del circuito stampato, così come l'immagine di Raspbian e lo sketch di RandA, sono scaricabili dal sito della rivista.

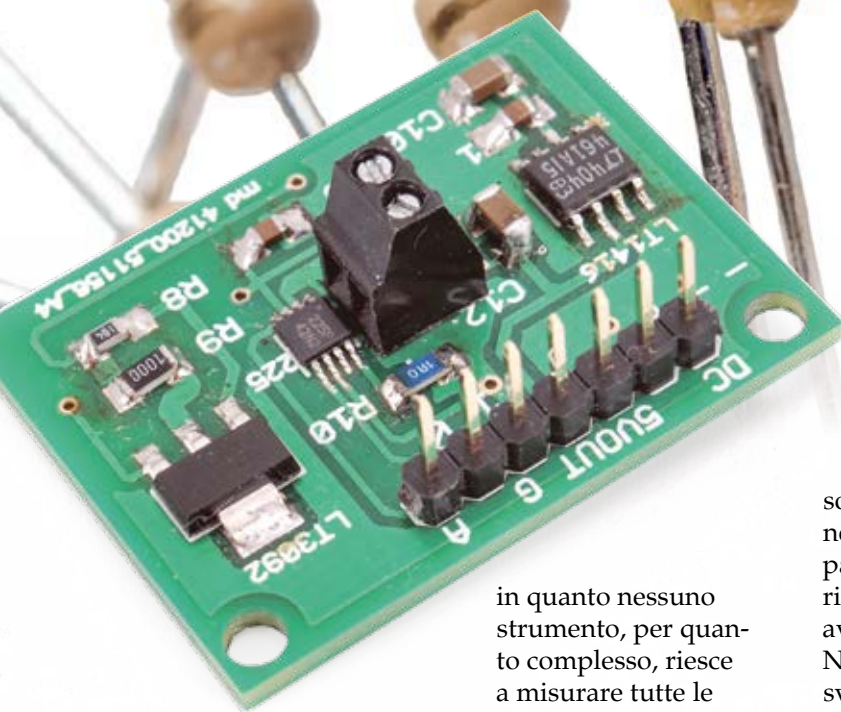
Raspberry Pi 2 Tipo B (cod. 8326274RS) è reperibile a Euro 46,00 Euro; il camera module per Raspberry Pi (cod. 7757731RS) costa 31,00 Euro; la scheda RandA, (cod. RANDA) è disponibile al prezzo di 39,00 Euro.

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

MILLIOHMMETRO DIY

dell'Ing. VINCENZO MENDOLA

Convertitore corrente/tensione con il quale misurare le resistenze di valore inferiore ad 1 Ω , sia in modalità stand-alone che tramite una scheda a microcontrollore dotata di ADC, come ad esempio mbed.

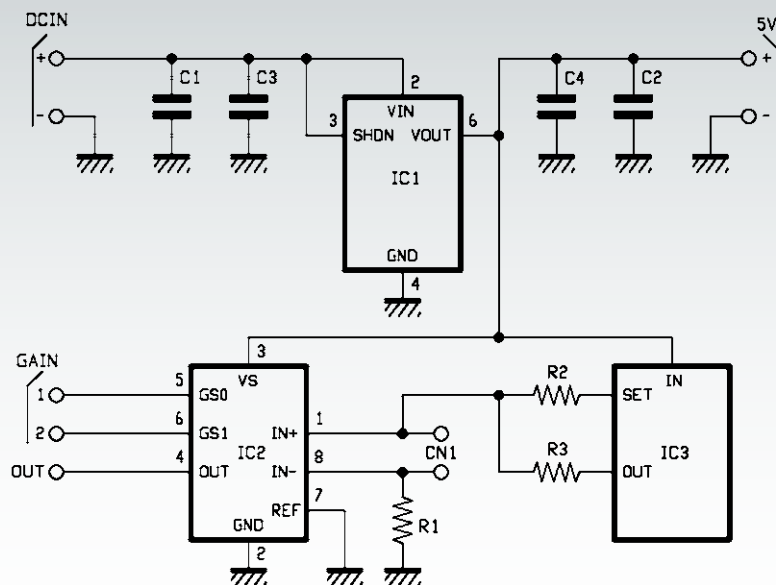


Avere a disposizione la strumentazione adatta alla misura che si deve compiere è determinante; se si vogliono misurare più parametri, è necessario disporre di più strumenti, di tipo diverso,

in quanto nessuno strumento, per quanto complesso, riesce a misurare tutte le grandezze utilizzate nel campo dell'elettronica e della radio frequenza. La strumentazione professionale ha solitamente costi proibitivi per l'hobbista medio e siccome difficilmente basterà il possesso di un unico strumento per

soddisfare le proprie necessità, spesso l'appassionato è costretto a rinunciare del tutto ad avere certi strumenti. Negli ultimi anni lo sviluppo di sistemi a microcontrollore a basso costo ha permesso di risolvere parzialmente questo problema: grazie anche alla grande capacità di calcolo disponibile su ogni computer fisso o portatile che sia, oggi è possibile trovare a prezzi interessanti

strumentazione che delega la parte di vero e proprio calcolo sia al proprio computer, sia alla propria scheda di sviluppo (in questo caso generalmente il lavoro è stato svolto da altri appassionati per applicazioni in ambito non professionale). È sempre più frequente trovare in rete applicazioni relativamente avanzate che trasformano, per fare un esempio, una scheda Beaglebone



Black opportunamente interfacciata ad un dongle USB per ricevere il digitale terrestre, in un semplice analizzatore di spettro. A differenza degli strumenti stand-alone che sono quindi autonomi in tutto per tutto, questi necessitano di un software che, invece di "girare" sul microprocessore dell'elettronica che costituisce lo strumento vero e proprio, si appoggia al sistema operativo del computer al quale

si interfaccia. In questo modo si riesce generalmente ad abbattere di molto i costi, trasferendo molte delle capacità dello strumento vero e proprio, alla qualità e complessità del software che gestisce l'elettronica che realizza di fatto la misura.

Questo trasferimento dei costi computazionali ai sistemi di calcolo che già possediamo, ha permesso in alcuni casi anche agli hobbisti di dotarsi di stru-

mentazione che aveva costi proibitivi in passato. Seguendo la filosofia del rendere a portata di tutte le tasche la strumentazione di misura, oggi vi vogliamo presentare un milliohmometro la cui autocostruzione è facile ed economica; il circuito pone come unica condizione un pizzico di manualità che serve per montare i componenti elettronici di tipo SMD da cui è composto.

Seguendo la logica dei sistemi di tipo break out board, questo semplice strumento potrà essere utilizzato da chiunque desideri effettuare misure nel campo dei milliohm, senza dover investire cifre improponibili in strumentazione professionale; per questo motivo il milliohmometro potrà anche essere semplicemente usato insieme ad un comunissimo multimetro digitale di cui ogni hobbista è tipicamente in possesso. Ovviamente lo strumento potrà essere anche interfacciato con semplicità ad una qualsiasi scheda di sviluppo a microcontrollore dotata di A/D Converter.

Il condizionatore senza elettricità

La costante ricerca di soluzioni a basso impatto ambientale ai problemi quotidiani è un must dei nostri tempi e non a caso sempre più soggetti sono impegnati nel proporre tecnologie per limitare i consumi elettrici.

Tra gli utilizzatori più energivori si annoverano i condizionatori d'aria, responsabili di ben il 40% del consumo globale di elettricità e del 36% delle emissioni totali di CO₂. In questo contesto non si può non accogliere con favore l'arrivo di FREESCOO (FREE Solar COOLing), il primo condizionatore per abitazioni alimentato



ad energia solare, che permette di avere aria condizionata senza attingere alla rete elettrica, ma ricavando sul posto l'elettricità occorrente mediante pannelli fotovoltaici integrati nel sistema e due batterie di stoccaggio. FREESCOO può essere usato anche nella stagione invernale come pompa di calore, a patto che via sia sufficiente irraggiamento solare.

SCHEMA ELETTRICO

Nel numero 192 di febbraio scorso, vi abbiamo presentato due semplici schede di current sense. Adattando in maniera semplice quel tipo di circuito, è possibile effettuare la misura di resistenza: per misurare la corrente ci siamo serviti di una resistenza di valore noto, la cui caduta di tensione ci permetteva di determinare con facilità, tramite la legge di Ohm, la corrente circolante; ora, in maniera duale, fissiamo la corrente circolante nella resistenza che in questo caso costituirà l'incognita nella relazione che lega tensione e corrente nella legge di Ohm e, come nel caso precedente, misuriamo la caduta di tensione ai capi della resistenza di cui vogliamo determinare il valore. Semplice, no?!

Per ottenere questo risultato abbiamo preso lo stesso circuito già presentato a Febbraio con l'INA225, cui abbiamo aggiunto un generatore di corrente costante programmabile della Linear Technology, LT3092 (<http://www.linear.com/docs/27892>), che fa circolare una corrente di valore costante e noto nella resistenza da misurare.

L'integrato LT3092 è in grado di erogare una corrente di valore compreso tra 0,5 e 200 mA, con un'accuratezza iniziale dell'1%; il rapporto tra R2 e R3 determina il valore della corrente erogata -espresso in microampere- secondo la seguente relazione:

$$I=10 (R2/R3)$$

Nel nostro caso la corrente è stata fissata al valore di 10 mA ($R2=100 \text{ kohm}$; $R3=100 \text{ ohm}$) per due motivi, il primo dei quali è utilizzare come sorgente di alimentazione la tensione erogata da un LT1461 (<http://www.linear.com/docs/1342>), che è un riferi-

mento di tensione di precisione che può erogare correnti superiori a 50 mA (si tratta dello stesso componente che abbiamo già utilizzato nell'articolo dei current sense).

Il secondo motivo è poter misurare correnti dell'ordine del decimo di milliohm senza rischiare di saturare l'INA225, anche con guadagno pari a 100, impostazione che permette di misurare velocemente anche con un multimetro la resistenza e di avere facilmente il valore espresso in milliohm ($1\Omega = 1V$). Ovviamente nulla vieta di utilizzare valori differenti di corrente e di utilizzare un altro dispositivo per fornire la tensione di alimentazione richiesta, a seconda delle vostre necessità. A tal proposito ricordiamo che l'INA225 va alimentato con una tensione compresa tra 2,7 – 36 V, mentre per l'LT3092 questa deve essere nell'intervallo 1,2 ÷ 40 V. È chiaro che se utilizzerete un ADC, la tensione massima che gli potrà essere applicata all'ingresso dovrà essere tenuta in considerazione nella scelta dei precedenti parametri, in quanto, come già visto nel fascicolo di febbraio scorso, la dinamica in uscita all'INA225 è legata alla tensione di alimentazione di questo integrato.

Vediamo nuovamente insieme le principali caratteristiche di questo ottimo current sense della Texas Instruments: l'INA225 (<http://www.ti.com/lit/gpn/ina225>) ha un'elevata accuratezza del guadagno (+/- 0,3 % massimo), impostabile su quattro valori predefiniti (25, 50, 100, 200), selezionabili tramite il livello logico di 2 pin (GSO e GS1), indicati come "GAIN" "1" e "0" sullo strip posto sul bordo del PCB (si veda la **Tabella 1** per ulteriori dettagli). Il dispositivo permette

Tabella 1

GSO	GS1	guadagno
GND	GND	25
GND	VS	50
VS	GND	100
VS	VS	200

di effettuare misure di corrente in maniera bidirezionale: per garantire tale caratteristica è necessario fornire una tensione di riferimento (reference voltage) che assicura la dinamica d'uscita richiesta per entrambi i versi della corrente, la cui ampiezza massima non deve essere necessariamente simmetrica (si veda il data-sheet alla sezione "Bidirectional Operation" per ulteriori dettagli); in questo caso verrà utilizzato in modalità unidirezionale e quindi il pin "REF" dell'INA225 è collegato al potenziale del riferimento di massa.

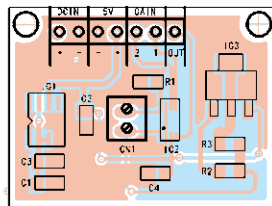
Per quanto riguarda il riferimento di tensione vale quanto già visto nell'articolo relativo ai current sense: l'integrato LT1461-5 (uscita a 5V), accetta in ingresso tensioni fino a 20 V ed è caratterizzato da un basso dropout (solamente 300 mV). I due condensatori da 1µF (in ingresso) e da 2,2 µF (in uscita) devono essere necessariamente ceramici, in modo da garantire un bassissimo valore di ESR (Equivalent Series Resistance), che assicura l'elevata accuratezza dichiarata dal costruttore. Le connessioni del riferimento di precisione sono semplici e si limitano a quattro pin: tensione d'ingresso, tensione d'uscita, massa e shutdown (funzionalità non utilizzata e quindi il terminale è connesso stabilmente alla tensione d'ingresso per mantenere l'LT1461 sempre "acceso" quando è presente tensione). Ricordiamo che, qualora si decidesse di utilizzare (come consigliato) l'LT1461 anche

Elenco Componenti:

R1: 1 ohm (1206)
 R2: 100 kohm (1206)
 R3: 100 ohm (1206)
 C1: 1 μ F ceramico (1206)
 C2: 2,2 μ F ceramico (1206)
 C3: 100 nF ceramico (1206)
 C4: 100 nF ceramico (1206)
 IC1: LT1461DNS8-5#PBF
 IC2: INA225AIDGKT
 IC3: LT3092EST#PBF

Varie:

- Morsetto passo 2.54mm
- Strip maschio 7 vie
- Circuito stampato S1203



come riferimento di tensione dell'ADC, è necessario scegliere la versione che fornisce la tensione in uscita di valore adeguato al funzionamento del microcontrollore scelto, al fine di evitarne danneggiamenti irreversibili, perché come ben sapete non tutti lavorano a 5V (come ad esempio Arduino) ma sono sempre più diffusi quelli che lavorano a tensioni inferiori: specialmente 3,3V (mbed, svariati PIC e dsPIC, eccetera). La tensione a cui connettere l'ADC oppure il multimetro, è il pin "OUT", sul quale si avrà una tensione proporzionale al valore della resistenza da misurare, moltiplicata per il guadagno dell'INA225: se avete seguito il nostro suggerimento, impostando la corrente al valore di 10 mA e il guadagno a 100, un millivolt su Vout corrisponderà a 1m Ω , o, se preferite, 1 Ω di resistenza corrisponderà ad 1V. Noi siamo interessati alle frazioni di ohm, per cui il valore letto ci fornirà direttamente il valore della resistenza in frazioni di V (decine o centinaia di millivolt) e questo rende immediato l'utilizzo dello strumento anche con un qualunque voltmetro digitale

munito di adeguata risoluzione. È necessaria una precisazione: data la natura del tipo di misura, nel range che va dai milliohm alle centinaia di milliohm, prima di usare lo strumento conviene effettuare una semplice "taratura" da utilizzare in fase di calcolo, consistente nella misura della resistenza parassita delle piste e dei conduttori e dei relativi terminali a coccodrillo utilizzati per connettere la resistenza di basso valore al circuito di misura che ha un valore paragonabile a quello del componente che si vuole misurare, per cui è necessario tenerne conto per evitare errori grossolani; per quanto riguarda la corrente erogata dal generatore di corrente, l'inevitabile errore a cui sarà affetta, legato alla tolleranza dei componenti utilizzati è sufficientemente basso da poter essere trascurato (sul nostro prototipo abbiamo misurato un errore di circa 80 μ A) ma nulla vieta di tenerne conto durante la misura. Ovviamente l'accuratezza del nostro strumento non è paragonabile a quella di strumenti professionali ma è comunque utile per chiunque abbia la necessità di fare

misure di frazioni di ohm senza dover spendere una fortuna. Per quanto detto in precedenza, cioè la relativa facilità con cui oggi è possibile realizzare della strumentazione utilizzando le schede di prototipazione facilmente accessibili agli hobbisti, vi mostriamo un esempio in cui abbiamo interfacciato il milliohmmetro all'mbed Freescale FRDM KL-25Z, uno dei nostri preferiti della sempre più numerosa famiglia mbed (ora ARM mbed) perché unisce delle caratteristiche eccezionali ad un prezzo veramente interessante, inferiore a quelle di una scheda Arduino Uno di cui mantiene la disposizione degli strip per facilitare l'uso degli shield per questo sviluppati. Ricordiamo che il KL-25Z (che dispone di un ADC a 16 bit) lavora a 3,3 volt, per cui chi vorrà cimentarsi in questo semplice esempio dovrà tenerne conto per evitare, come già detto in precedenza, danneggiamenti irreversibili della scheda che occorrerebbero se essa venisse collegata direttamente ai 5V (e questo vale ovviamente anche per l'ADC). Consigliamo, almeno a chi non ha particolare esperienza, di utilizzare direttamente schede Arduino o analoghe a 5V per le quali il porting del semplice codice nel listato si potrà effettuare senza troppe difficoltà.

La Fig. 1 mostra il valore misurato tramite il Freescale KL-25Z e il nostro milliohmmetro, di una



Fig. 1

resistenza corazzata commerciale da 5 W di 0,22 Ω , tramite la connessione USB Virtual Serial Port in cui la connessione USB è utilizzata come porta seriale per l'interfacciamento al PC. La schermata è quella dell'emulatore di terminale PuTTY con il quale abbiamo monitorato la connessione seriale relativa alla scheda mbed usata per la misura. Nel **Listato 1** trovate il codice per la scheda mbed FRDM KL-25Z.

REALIZZAZIONE PRATICA

Come già evidenziato, il dispositivo è stato realizzato come breakout board per agevolarne l'integrazione in dispositivi già esistenti o per essere utilizzato in combinazione ad un comune multimetro digitale. Per ridurre le dimensioni e facilitarne l'utilizzo in qualunque applicazione, le unità sono state realizzate con componentistica SMD. Chi vorrà realizzare questi dispositivi in autonomia, potrà farlo utilizzando i file con le tracce per la foto incisione sul nostro sito Internet (www.elettronica.in.it). Trattandosi di un circuito a montaggio superficiale, prima di intraprenderne la realizzazione è opportuno disporre, oltre che di una minima manualità ed esperienza nei montaggi di componentistica SMD, dell'attrezzatura necessaria, consistente in un saldatore di buona qualità (si consiglia una stazione saldatrice con regolazione della temperatura) dotato di punta di dimensione adatta alla tipologia di componenti da saldare, set di pinzette con caratteristiche adatte all'impiego con componenti a montaggio superficiale e fluxante come ad esempio come ad esempio il Kester flux 951 con dosatore di tipo a "penna". Una volta in possesso del ma-

Listato 1

```
#include "mbed.h"

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

float VADC=2.922;           //ADC VOLTAGE REFERENCE (EQUAL TO V SUPPLY)
//float IO=10.0;           //COSTANT CURRENT
//float GAIN=100.0;        //INA225 GAIN
//BOTH NOT REQUIRED IF SET LIKE THIS (1  $\Omega$  = 1 V)

float RO=0.021;             //RESISTANCE OF CONNECTING CABLES (COMPENSATION)
float STAMPA_R = 0;

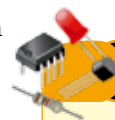
AnalogIn resistor(A0);      //CONNECTED TO INA225 OUT

float ohm()
{
    float R=0.0;
    float RADC=resistor.read();
    R = VADC*RADC-RO;       //IF IO=10mA AND INA225 GAIN 100 -> (1  $\Omega$  = 1 V)
    wait_ms(100);
    return R;
}

int main()
{
    while (1) {
        STAMPA_R= ohm();
        pc.printf("R = ");
        pc.printf("%1.3f",STAMPA_R); //R IN OHM WITH 3 DECIMALS (1m $\Omega$  OF RESOLUTION)
        pc.printf("\n");
        pc.printf("\r");
        wait_ms(500);
    }
}
```

teriale necessario, si consiglia di iniziare a saldare i componenti passivi, per poi passare al montaggio degli integrati. In questo caso l'uso del fluxante, in quantità adeguata, vi aiuterà ad evitare di saldare erroneamente tra di loro i piedini vicini. Utilizzate la pinzetta per mantenere in posizione il dispositivo e iniziate a saldare due piedini diametralmente opposti per mantenere fermo l'integrato. Procedete poi a saldare i piedini rimanenti. Prestate attenzione a non scaldare eccessivamente (oltre il necessario) i componenti, specialmente gli integrati, che potrebbero subire danni irreversibili durante il processo. A montaggio ultimato, con l'aiuto di una lente di ingrandimento verificate che non vi siano "baffi" o gocce di stagno che uniscono componenti attigui o terminali vicini dello stesso integrato.

Il vostro milliohmmetro è dunque pronto per l'uso.



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità e possono essere acquistati presso Futura Elettronica. Il master del circuito stampato può essere scaricato dal sito della rivista. L'integrato Current Sense Amp (cod. INA225AIDGKT) è disponibile a euro 5,60 Euro; il regolatore di tensione (cod. LT1461DHS8-5) costa 5,20 Euro; il regolatore di corrente (cod. LT3092EST) è reperibile al prezzo Euro 6,60 Euro.

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

CORSO MPLAB X



di FRANCESCO FICILI
e VINCENZO GERMANO

Continuiamo il nostro viaggio alla scoperta di MPLab X, il nuovo ambiente di sviluppo integrato sviluppato da Microchip Technology per sostituire l'MPLab IDE. Ci occupiamo di come realizzare applicazioni che impiegano lo stack USB host di Microchip. Settima puntata.

Nella scorsa puntata abbiamo visto come realizzare un'applicazione USB Device su PIC32, servendoci dello stack USB open source di Microchip e dello scheduler introdotto nella puntata 5; adesso facciamo più o meno la stessa cosa, ma utilizzando lo stack USB Host invece del device. L'applicazione di esempio che andremo a realizzare ci permetterà di "loggarci" su una comune chiavetta USB la tensione letta su un canale analogico del microcontrollore che abbiamo collegato a un potenziometro.

LO STACK USB HOST

Sappiamo che la tipica struttura prevista dallo standard USB è costituita da un'unità Host e da una o più unità device. I Device USB non pos-

sono comunicare tra di loro, ma solamente con l'Host, che di fatto assume il ruolo di master del bus USB. Nella maggior parte delle applicazioni che in genere si prendono in considerazione, il ruolo dell'Host è ricoperto da un Personal Computer, il quale, disponendo di un sistema operativo piuttosto complesso (come ad esempio Microsoft Windows o Mac OS) e di risorse hardware virtualmente infinite per l'applicazione, può essere agevolmente configurato dinamicamente (se non ne dispone già) con il relativo driver della periferica USB specifica che deve andare ad interfacciare. Nel caso dell'embedded host, invece, la situazione è piuttosto differente: nella maggior parte delle applicazioni non si dispone di un sistema operativo stile Windows/

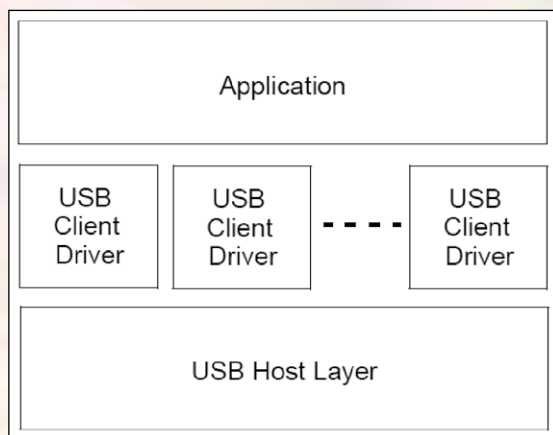


Fig. 1 - Architettura dello stack USB Host.

Linux, ma di un semplice RTOS o, come nel nostro caso, solo di uno scheduler, e non è possibile aggiornare il sistema con un nuovo driver all'occorrenza. Bisogna quindi trattare il problema in maniera differente, considerando l'applicazione specifica che utilizzerà il bus e progettando il firmware di conseguenza.

ARCHITETTURA DELLO STACK USB HOST

L'architettura dello stack USB Host di Microchip è rappresentata schematicamente in Fig. 1, nella quale potete vedere che consiste in tre layer, ossia Application, USB Client

Driver Layer e USB Host Layer, di seguito descritti.

- **Application Layer:** questo è lo strato applicativo dell'implementazione e raggruppa tutte le funzioni di alto livello dell'applicazione specifica. In genere viene sviluppato da zero dallo sviluppatore dell'applicazione.
- **USB Client Driver Layer:** come sappiamo, ogni dispositivo USB implementa una determinata funzione (ad esempio un mouse, una tastiera, una memoria, ecc.). Ognuno di questi dispositivi ha un'implementazione specifica differente e di conseguenza necessita di un driver specifico per poter essere interfacciato. Questo strato implementa proprio i vari client driver di tutti i possibili dispositivi USB che possono essere connessi all'embedded Host.
- **USB Host Layer:** questo layer fornisce un'astrazione del protocollo USB e in particolare si occupa dell'implementazione dei seguenti servizi:
 - identificazione del dispositivo;
 - enumerazione del dispositivo;
 - gestione del client driver specifico;
 - astrazione della comunicazione con il dispositivo.

Quando un dispositivo viene connesso al bus, il layer host lo identifica leggendone i descrittori (specifiche strutture identificative definite dalla specifica USB 2.0), dopodiché controlla se un driver del dispositivo è presente all'interno di una specifica lista di client driver e se ne trova uno adatto lo inizializza per consentirne l'utilizzo ai livelli superiori.

L'operazione di identificazione ed enumerazione viene gestita in background da uno specifico task realizzato sotto forma di una macchina a stati, la cui implementazione è visibile in Fig. 2.

Come si può vedere, una volta collegato, un dispositivo passa attraverso vari stati, prima di poter essere "battezzato" come configurato e quindi passare in stato di running, dove può essere effettivamente utilizzato dall'host.

CLIENT DRIVER TABLE

Come abbiamo visto in precedenza, ogni dispositivo USB (o famiglia di dispositivi) ha uno specifico driver (client driver) che ne permette la gestione. Dato che un host deve

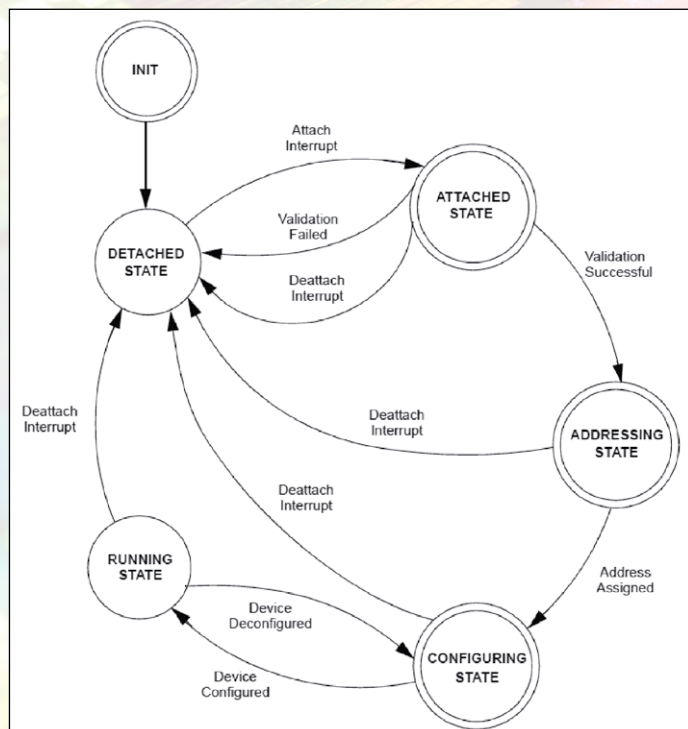


Fig. 2 - Implementazione della macchina a stati che gestisce l'enumerazione di un dispositivo USB nello stack USB Host di Microchip.

in genere supportare più di un dispositivo (questo può avvenire nel caso in cui debba essere supportata una classe di dispositivi o anche nel caso in cui si supporti un dispositivo singolo ma composito, ossia che implementa più di una funzione, con relativa interfaccia, al suo interno), quest'ultimo implementa generalmente più di un client driver. Nell'implementazione dello stack USB Host di Microchip, questo requisito viene soddisfatto implementando una tabella, che prende il nome di Client Driver Table (o CDT). Ogni entry di questa tabella corrisponde ad un singolo client driver e contiene un puntatore alla funzione di inizializzazione ed alla callback che si occupa della gestione degli eventi (event handling callback). La Fig. 3 rappresenta schematicamente l'implementazione della Client Driver Table. Quando un device viene connesso al bus, il layer host ne legge i descrittori e determina se il dispositivo può essere supportato o no. In caso affermativo, la corretta entry della client driver table viene indicizzata e viene chiamata la routine di inizializzazione. Una volta inizializzato il dispositivo, quando un particolare evento viene generato in corrispondenza di una qualche attività, lo specifico event handler viene chiamato per la gestione dell'evento specifico.

TARGETED PERIPHERAL LIST

Come detto in precedenza, un'implementazione embedded host, a differenza di un Personal Computer, può supportare solo un limitato set di dispositivi USB. Questo set, nell'implementazione di Microchip, è definito da una Targeted Peripheral List (o TPL), ossia una lista di dispositivi supportati.

I dispositivi supportati possono essere identificati in una delle due seguenti modalità:

- Combinazione VID/PID;
- Combinazione classe/sottoclasse/protocollo.

Un esempio di un elemento di una TPL che utilizza la combinazione classe/sottoclasse/protocollo è riportato in Fig. 4.

La specifica USB impone che ogni disposi-

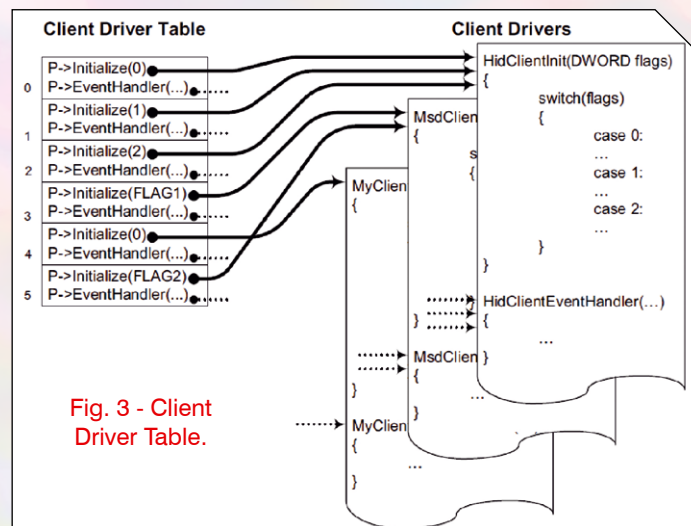


Fig. 3 - Client Driver Table.

tivo sia identificato per mezzo di una VID (acronimo di Vendor ID) e un PID (Product ID). La combinazione di questi due numeri identifica una specifica famiglia di prodotti di uno specifico venditore, in modo da poter determinare un'implementazione specifica e come supportarla. Un altro metodo fornito dalla specifica per determinare il comportamento di un generico dispositivo USB, e di conseguenza progettare l'apposito client driver, consiste nell'identificare la classe di appartenenza. Sempre secondo la specifica, i dispositivi USB sono suddivisi in classi, e all'interno di esse in sottoclassi e protocolli supportati. Identificando questa combinazione è possibile determinare il comportamento specifico di un generico dispositivo. Sia il VID/PID che la combinazione classe/sottoclasse/protocollo possono essere agevolmente ricavate da un dispositivo esaminando i suoi descrittori.

Nello stack USB Host di Microchip, la TPL è modellata come una tabella che associa l'identificato del dispositivo (che può essere determinato con uno dei due sistemi descritti in precedenza) con un elemento della Client Driver Table. In questo modo, quando un dispositivo è collegato, la TPL viene analizzata per determinare se il dispositivo è supportato dall'host, ed in caso positivo, un indice indica quale entry della Client Driver Table deve essere utilizzata per gestire il dispositivo. Oltre a fornire un riferimento per la CDT, nella TPL

Description	Class Name	Class Code	Subclass Code	Protocol
Flash Drive	Mass Storage	0x08	0x06	0x50

Fig. 4 - Elemento di una TPL identificato tramite la combinazione classe/sottoclasse/protocollo.

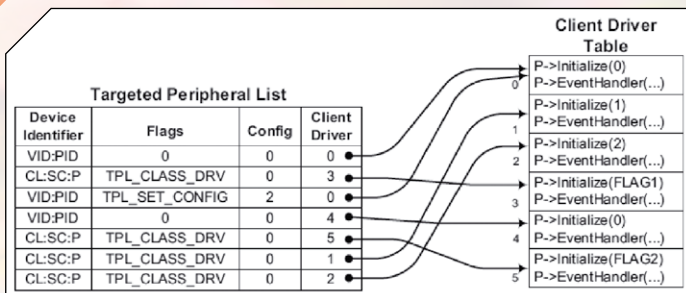


Fig. 5 - Relazione TPL - Client Driver Table.

sono contenute altre informazioni come flag e parametri di configurazione che possono essere passati allo specifico client driver in funzione dei requisiti dell'applicazione specifica.

La Fig. 5 fornisce una rappresentazione schematica di quanto esposto in precedenza.

Si noti, sempre dalla Fig. 5 che una singola entry della TPL può puntare allo stesso driver nella CDT (nell'esempio la prima e la terza); questo permette, quando dei dispositivi sono identificati per mezzo della combinazione VID/PID, di utilizzare lo stesso driver, nel caso l'implementazione della loro interfaccia sia effettivamente la stessa.

Utilizzate in combinazione, la TPL e la CDT costituiscono un potente meccanismo per la gestione di un gruppo eterogeneo di dispositivi USB che debbano essere supportati da un singolo embedded host. Chi volesse maggiori informazioni sulla struttura dello stack USB host di Microchip può fare riferimento ad alcune interessanti application notes, come ad esempio l'AN1140 e l'AN1141.

LA CLASSE USB MSD

Quanto descritto vale per ogni implementazione USB Host realizzata con lo stack USB di Microchip. Tuttavia, come abbiamo visto in precedenza, a seconda del dispositivo che andiamo ad interfacciare, l'implementazione specifica dell'applicazione lato Host cambierà, in quanto bisognerà implementare di sicuro lo specifico driver e anche qualche altro strato aggiuntivo in funzione dell'applicazione. Fortunatamente anche in questo Microchip ci viene incontro, fornendoci una serie di client driver per le più comuni classi di dispositivi USB (come ad esempio flash drive, dispositivi HID, stampanti, ecc.) e anche altri strati aggiuntivi necessari per alcune applicazioni (ad esempio nel caso della classe MSD vengono forniti anche dei componenti per la gestione del file system e dell'interfaccia SCSI). Tali pacchetti sono parte dello stack USB (o più

genericamente delle MLA) ed esiste anche un'implementazione general purpose (chiamata generic host) che costituisce di fatto un template per l'implementazione di client driver per le classi non ufficialmente supportate dallo stack.

L'applicazione specifica che analizzeremo nel dettaglio in questa puntata è l'implementazione di un Host USB capace di gestire una classe MSD (mass storage device) con file system, quindi implementeremo di fatto un embedded host capace di interfacciare le comuni Pen Drive USB. La struttura che andremo a realizzare si appoggia sull'architettura descritta in precedenza ed aggiunge alcuni strati proprio al di sopra del mass storage client driver, come visibile in Fig. 6. Tali strati aggiuntivi sono:

- **SCSI Command Set**; questo livello gestisce l'interfaccia SCSI verso la memoria;
- **File System**; implementa la gestione del file system (è utilizzato il tipo FAT32) e costituisce l'effettiva interfaccia di alto livello utilizzata dall'applicazione per interagire con il disco;
- **Application**; questo strato implementa l'applicazione specifica, in questo caso una applicazione di datalogging.

Lo strato di gestione del file system mette a disposizione diverse API per la gestione del disco, come funzione di scrittura, di lettura, gestione delle directory ed altro ancora. Per una descrizione dettagliata delle varie API è possibile consultare l'AN1045: "Implementing

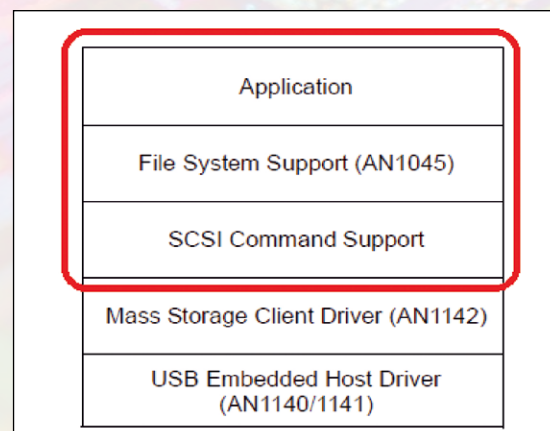


Fig. 6 - Architettura di un'applicazione host che supporta la classe USB MSD.

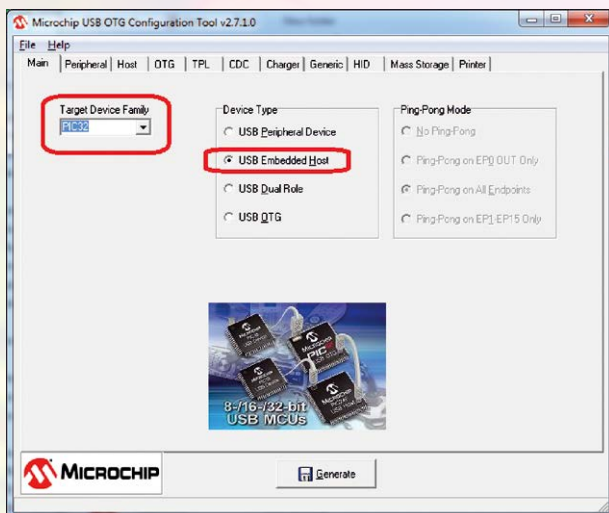


Fig. 7 - Schermata principale del tool UsbConfig.

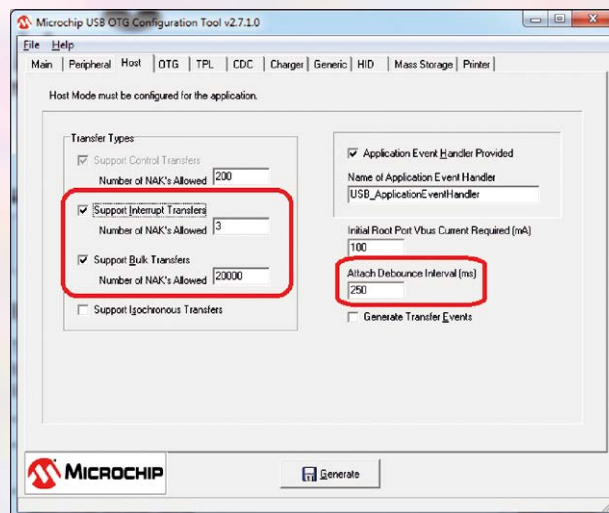


Fig. 8 - Configurazione dei parametri Host.

File I/O Functions Using Microchip's Memory Disk Drive File System Library".

CONFIGURAZIONE DI UN PROGETTO USB HOST

Come abbiamo potuto notare dai paragrafi precedenti, l'implementazione di un'applicazione USB host che interagisca con una memoria USB può risultare un'operazione di una certa difficoltà, in quanto il framework host è ancora più complesso dell'implementazione device: in particolare, anche la sola configurazione del framework per l'applicazione specifica può essere una procedura decisamente complicata da eseguire. In un progetto USB host realizzato con lo stack USB di Microchip, tutte le configurazioni relative al framework sono contenute all'interno della coppia di file `usb_config.c/usb_config.h`. In particolare oltre alla TPL ed alla CDT, sono contenute tutte le impostazioni generiche e specifiche delle particolari classi supportate. Poiché tali configurazioni variano da applicazione in applicazione (così come anche la composizione della TPL e della CDT), Microchip ha realizzato un'applicazione, chiamata appunto UsbConfig, che permette di autogenerare il codice relativo alla configurazione tramite un'interfaccia grafica intuitiva. Vediamo come funziona questa applicazione generando i file relativi al progetto pratico che illustreremo in questa puntata. La schermata iniziale del tool si presenta come in Fig. 7 e ci permette di selezionare la piattaforma e la tipologia di dispositivo (oltre all'embedded host e al device sono presenti anche delle implementazioni ibride, come il dual role e l'OTG, che non sono

trattate in questa puntata). Scegliamo PIC32 come piattaforma e Embedded Host come "device type" e andiamo avanti, spostandoci sul tab "Host". Da qui è possibile scegliere il tipo di trasferimento da utilizzare e altre opzioni di base come il nome della callback di event handling, la corrente massima da erogare, ecc. Selezioniamo Interrupt e bulk come modalità di trasferimento dati (la modalità isocrona non è necessaria per l'applicazione), 100 mA come massima corrente e portiamo l'attach debounce interval a 250 ms, in modo da poter supportare anche i dispositivi più lenti. Non supportiamo i transfer events e manteniamo il nome predefinito per l'application event handler. In Fig. 8 è riportato uno screenshot della configurazione da inserire. Spostiamoci ora sul tab TPL per creare la "Targeted Peripheral List", come illustrato

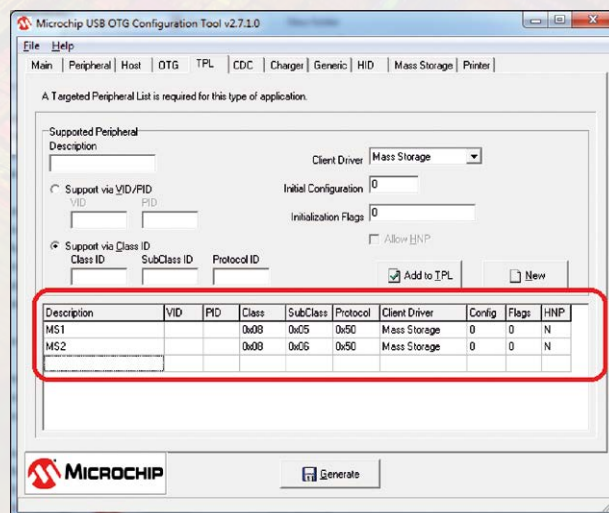


Fig. 9 - Creazione della TPL.

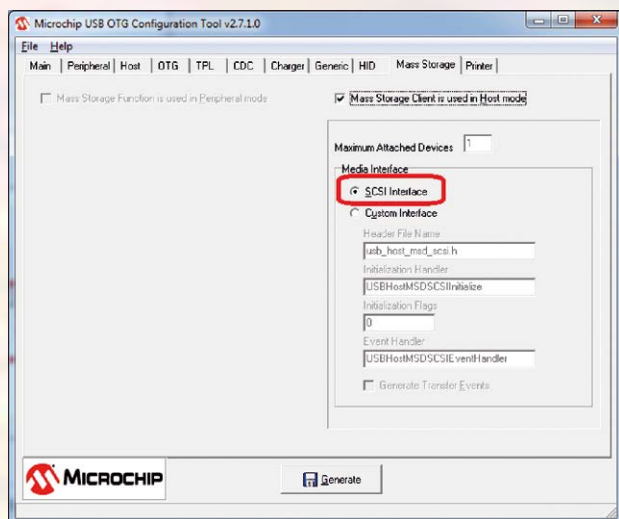


Fig. 10 - Impostazione dell'interfaccia verso il disco.

in Fig. 9. In questo tab bisogna inserire le varie entry della TPL, che consistono nei parametri identificativi del dispositivo (con i due metodi descritti in precedenza, combinazione VID/PID o combinazione classe/subclasse/protocollo), il relativo client driver (per questo è presente un dropdown box che

contiene i client driver più comuni, oppure è possibile utilizzare la generic class per implementarne uno nuovo), oltre ad eventuali flag e configurazioni. Nel nostro caso inseriremo due entry, entrambe collegate al Mass Storage Client Driver: la prima identificata dalla terna classe/subclasse/protocollo 0x08 – 0x05 – 0x50 e la seconda dalla terna 0x08 – 0x06 – 0x50. Questo perché alcune periferiche mass storage utilizzano la sottoclasse 0x05 mentre altre la 0x06. In questo modo siamo certi che il nostro embedded host le riconosca entrambe, collegando sempre lo stesso client driver. In entrambi i casi non inseriamo flag né opzioni particolari.

Non ci rimane che spostarci sul tab “Mass Storage” per inserire i dettagli dell'interfaccia specifica, come illustrato in Fig. 10. Nel nostro caso utilizzeremo l'interfaccia SCSI per la gestione del disco, quindi spuntiamo la relativa voce e completiamo. A questo punto è possibile cliccare sul pulsante “Generate”

Listato 1

```
// *****
// USB Embedded Host Targeted Peripheral List (TPL)
// *****

USB_TPL usbTPL[] =
{
    { INIT_CL_SC_P( 0x08ul, 0x05ul, 0x50ul ), 0, 0, {TPL_CLASS_DRV} } // MS1
    ,
    { INIT_CL_SC_P( 0x08ul, 0x06ul, 0x50ul ), 0, 1, {TPL_CLASS_DRV} } // MS2
};
```

Listato 1 - Implementazione C della TPL generata tramite il tool.

Listato 2

```
// *****
// Client Driver Function Pointer Table for the USB Embedded Host foundation
// *****

CLIENT_DRIVER_TABLE usbClientDrvTable[] =
{
    {
        USBHostMSDInitialize,
        USBHostMSDEventHandler,
        0
    },
    {
        USBHostMSDInitialize,
        USBHostMSDEventHandler,
        0
    }
};
```

Listato 2 - Implementazione C della CDT generata tramite il tool.

per creare i file; se apriamo il file `usb_config.c`, troviamo l'implementazione in C della TPL e della CDT, riportate, rispettivamente, nel **Listato 1** e nel **Listato 2**.

ESEMPIO PRATICO: DATALOGGER USB HOST

A questo punto possiamo passare alla descrizione dell'applicazione di esempio che presentiamo in questa puntata. Come abbiamo già detto nei paragrafi precedenti, la nostra idea è quella di realizzare una applicazione di datalogging che salvi su una comune per drive USB i valori letti da un canale analogico collegato ad un potenziometro. Un'applicazione interna di controllo darà il via e terminerà la fase di log. L'architettura del software è illustrata in **Fig. 11** e come si può vedere è basata, come di consueto ormai, sullo scheduler presentato nelle puntate precedenti. Come al solito, per fare in modo che lo scheduler lavori correttamente, trattandosi di un'implementazione non pre-emptive, dovremo sviluppare i nostri task utilizzando il paradigma del multitasking cooperativo. Come possiamo vedere dallo schema architetturale, sono stati previsti i quattro task seguenti.

- **UsbTask**: questo task si occupa di chiamare il task che mantiene le macchine a stati del driver USB Host (`USBTasks()`).
- **LogTask**: questo task implementa le macchine a stati dell'applicazione di logging. In particolare viene implementata una macchina a stati che si occupa del controllo (`LogControl()`) ed una macchina a stati che si occupa della gestione effettiva della memoria (`LoggingStateMachine()`).
- **AdcTask**: questo task implementa la macchina a stati che legge i dati dal convertitore analogico digitale e genera gli eventi di logging che vengono intercettati dal task di log.
- **LedTask**: questo task si occupa della gestione dei LED di sistema.

Oltre che sullo scheduler, la nostra architettura si appoggia sullo stack USB Host presentato in precedenza (e in particolare sull'implementazione del client driver MSD) per la gestione dell'USB e sulle librerie di sistema e sui driver delle periferiche per la

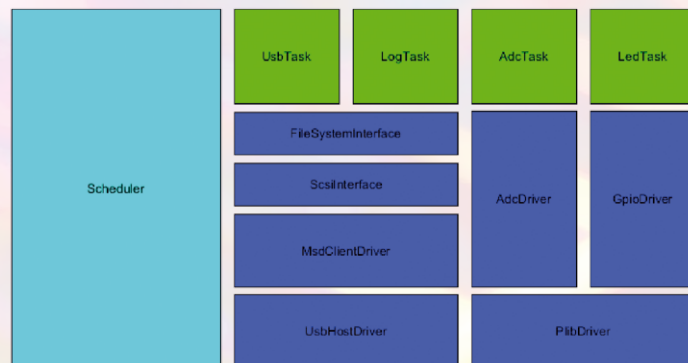


Fig. 11 - Architettura software.

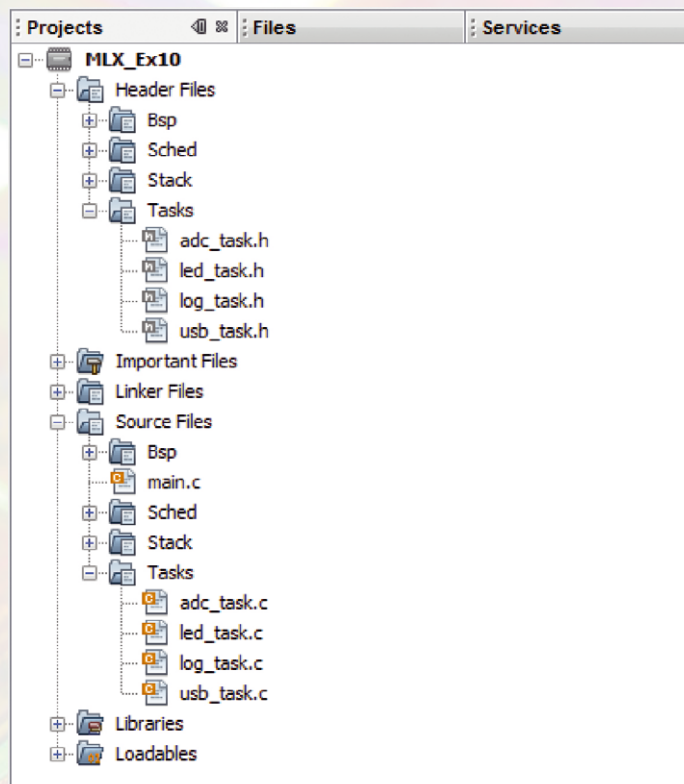


Fig. 12 - Struttura del Project Manager.

Listato 3

```
/* Array of task to call */
void (*TaskArray[ACTIVE_TASK_NUMBER]) (void) =
{
    LedTask,
    UsbTask,
    LogTask,
    AdcTask
};

/* Array of task period timeouts */
UINT16 TaskPeriodTimeoutMs[ACTIVE_TASK_NUMBER] =
{
    LED_TASK_PERIOD_MS,
    USB_TASK_PERIOD_MS,
    LOG_TASK_PERIOD_MS,
    ADC_TASK_PERIOD_MS,
};
```

Listato 3 - Array task e periodicità.

Listato 4

```

/*****
* Function:      void AdcStateMachine (void)
* Input:         None
* Output:        None
* Author:        F.Ficili
* Description:   Manage adc data acquisition state machine
* Date:          11/04/15
*****/
void AdcStateMachine (void)
{
    static AdcSmStateType AdcSmState = Idle;
    static UINT16 AcqDelayCounter = 0;
    UINT16 AdcDataRaw = 0;

    switch (AdcSmState)
    {
        case Idle:
            /* Check logger status */
            if (ReceiveEvt(&LoggerReady))
            {
                /* Jump to WaitLogDelay */
                AdcSmState = WaitLogDelay;
            }
            break;

        case WaitLogDelay:
            /* Increment counter */
            AcqDelayCounter++;

            /* If counter expires */
            if (AcqDelayCounter >= ACQ_DELAY_MS)
            {
                /* Reset counter */
                AcqDelayCounter = 0;
                /* Jump to AcquireData */
                AdcSmState = AcquireData;
            }
            break;

        case AcquireData:
            /* Get data */
            AdcDataRaw = ReadAdcData();
            /* Convert data to string */
            itoa(AdcDataBuffer, AdcDataRaw, 10);

            /* Jump to LogData */
            AdcSmState = LogData;
            break;

        case LogData:
            /* If a stop log evt is not received */
            if (!ReceiveEvt(&StopLog))
            {
                /* Log data event */
                GenerateEvt(&LogDataEvt);
                /* Jump to WaitLogDelay */
                AdcSmState = WaitLogDelay;
            }
            else
            {
                /* Stop log event */
                GenerateEvt(&CloseLogFile);
                /* Jump to Idle */
                AdcSmState = Idle;
            }
            break;

        default:
            break;
    }
}

```

Listato 4 - Implementazione della macchina a stati del task Adc.

UsbTask	10 ms
LogTask	50 ms
AdcTask	100 ms
LedTask	100 ms

Tabella 1 - Periodicità dei task.

gestione dell'ADC e del GPIO. La Fig. 12 illustra il Project Manager del progetto e mette in evidenza i quattro task che costituiscono lo strato applicativo. La tabella dei task dello scheduler, in questo caso, si presenta come nel Listato 3.

Da punto di vista delle periodicità sono stati utilizzati i valori riportati nella Tabella 1. Passiamo adesso ad un'analisi più dettagliata di alcuni task (analizzeremo AdcTask e LogTask, in quanto i due task UsbTask e LedTask effettuano più che altro operazioni di servizio), per comprendere come sono stati sviluppati, partendo dall'AdcTask. Questo task ha il compito di acquisire il canale analogico che deve essere loggato, ed inviare i relativi dati (già convertiti in formato stringa di testo) al task che si occupa della memorizzazione. Come al solito optiamo per una struttura di tipo macchina a stati finiti, la cui implementazione è riportata nel Listato 4. Come si può vedere sono stati previsti i seguenti quattro stati: Idle, WaitLogDelay, AcquireData, LogData. Inizialmente la macchina controlla se è stato ricevuto l'evento LoggerReady, che viene prodotto dal task LogTask,

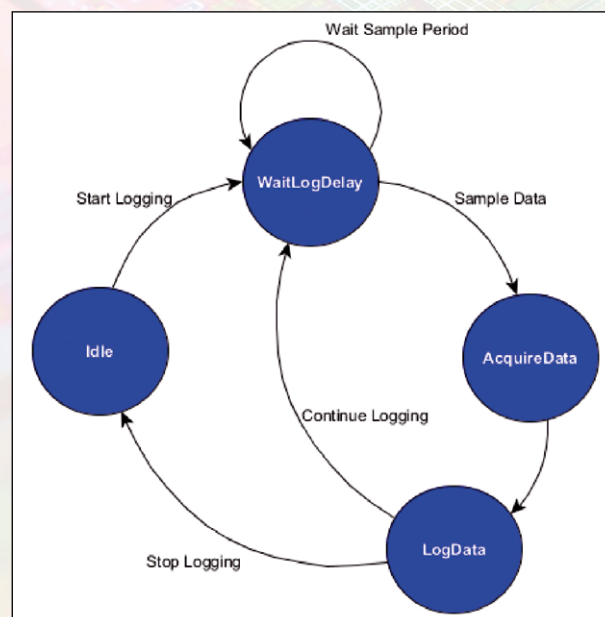


Fig. 13 - Macchina a stati AdcStateMachine.

non appena la periferica MSD è stata enumerata e correttamente inizializzata. Questo evita che vengano prodotti eventi di log prima che la memoria sia pronta a poterli memorizzare. A questo punto la macchina a stati entra in un ciclo all'interno del quale, allo scadere del periodo di campionamento il dato viene acquisito e convertito in stringa (WaitLogDelay, AcquireData). Poi, nello stato LogData, si controlla se è stato ricevuto o meno un evento di stop log; se l'evento non è stato ricevuto si invia l'evento LogDataEvt e si continua con l'acquisizione (tornando allo stato WaitLogData), altrimenti si invia l'evento CloseLogFile (per indicare al task di log di terminare e chiudere il file) e si ritorna nello stato di Idle. La rappresentazione grafica della macchina a stati relativa alla funzione AdcStateMachine è riportata in **Fig. 13**. Passiamo adesso all'esame dell'implementazione del task LogTask, che implementa tutte le operazioni relative al log dei dati sulla memoria USB. Questo task implementa due macchine a stati: la prima (LogControl) si occupa di controllare l'attività di log, generando un evento di stop dopo un certo periodo di tempo prefissato, mentre la seconda (LoggingStateMachine) si occupa di loggare i dati ricevuti dal task AdcTask sulla memoria USB. Il codice sorgente della macchina a stati LogControl è riportato nel **Listato 5**. L'implementazione è piuttosto semplice, in quanto viene semplicemente gestito un contatore per tenere traccia del tempo trascorso e generare l'evento di stop log quando il contatore stesso supera una certa soglia temporale, quindi non ci dilunghiamo e passiamo direttamente all'analisi della macchina a stati LoggingStateMachine, il cui codice è riportato nel **Listato 6**.

Questa macchina a stati sfrutta l'interfaccia di alto livello fornita dalla File System Interface per gestire direttamente la memoria. Le API utilizzate da questo task sono:

- BYTE USBHostMSDSCSIMediaDetect (void); verifica se un dispositivo MSD-SCSI è connesso al bus USB (ed è stato correttamente enumerato);

- int FSInit(void): Inizializza il file System;
- FSFILE * FSfopen(const char * fileName, const char *mode); apre un file (con la modalità indicate) e ritorna un puntatore allo stream di dati;

Listato 5

```

/*****
 * Function:      void LogControl (void)
 * Input:         None
 * Output:        UINT8
 * Author:        F.Ficili
 * Description:    Generates stop log event
 * Date:          19/04/15
 *****/
UINT8 LogControl (void)
{
    /* State machine state*/
    static SecCounterStateType SecCounterState = WaitSecDelay;
    /* Delay Counter */
    static UINT16 DelayCounter = 0;
    /* Seconds Counter */
    static UINT16 SecondsCounter = 0;

    switch (SecCounterState)
    {
        case WaitSecDelay:
            /* Increase counter */
            DelayCounter++;
            /* Check counter */
            if (DelayCounter >= SECOND_DELAY_MS)
            {
                /* Reset counter */
                DelayCounter = 0;
                /* Increase sec counter */
                SecondsCounter++;
                /* Jump to IncreaseCnt*/
                SecCounterState = IncreaseCnt;
            }
            else
            {
                /* Do nothing, satisfy MISRA */
            }
            break;

        case IncreaseCnt:
            /* Check acq timeout */
            if (SecondsCounter >= ACQ_TIMEOUT_S)
            {
                /* Generate stop log evt */
                GenerateEvt (&StopLog);
                /* Jump to Stop */
                SecCounterState = Stop;
            }
            else
            {
                /* Jump back to WaitSecDelay */
                SecCounterState = WaitSecDelay;
            }
            break;

        case Stop:
            break;

        default:
            break;
    }
}

```

Listato 5 - Implementazione della macchina a stati LogControl.

Listato 6

```

/*****
* Function:      void LoggingStateMachine (void)
* Input:         None
* Output:        None
* Author:        F.Ficili
* Description:   Logging task state machine
* Date:          11/04/15
*****/
void LoggingStateMachine (void)
{
    /* Logging state machine state */
    static LoggingSmStateType LoggingSmState = CheckDeviceDetected;
    static MassStorageStatusType MsStatus = DeviceDetached;
    static FSFILE *LogFile;
    static UINT8 DelayCounter = 0;

    switch (LoggingSmState)
    {
        case CheckDeviceDetected:
            /* Check device status */
            if (USBHostMSDSCSIMediaDetect())
            {
                /* Device attached */
                MsStatus = DeviceAttached;
                /* Jump to CheckFsInitialized */
                LoggingSmState = CheckFsInitialized;
            }
            else
            {
                /* Device detached */
                MsStatus = DeviceDetached;
            }
            break;

        case CheckFsInitialized:
            /* If file system is initialized */
            if (FSInit())
            {
                /* Jump to OpenFile */
                LoggingSmState = OpenFile;
            }
            break;

        case OpenFile:
            /* Open file */
            LogFile = FSfopen("LogFile.txt", FS_WRITE);
            /* Jump to WaitOpening */
            LoggingSmState = WaitOpening;
            break;

        case WaitOpening:
            /* Increment counter */
            DelayCounter++;

            /* If counter expires */
            if (DelayCounter >= OPENING_FILE_DELAY_MS)
            {
                /* Ready event */
                GenerateEvt(&LoggerReady);
                /* Reset counter */
                DelayCounter = 0;
                /* Jump to WaitLogEvent */
                LoggingSmState = WaitLogEvent;
            }
            break;

        case WaitLogEvent:
            /* If a log data */
            if (ReceiveEvt(&LogDataEvt))
            {
                /* Jump to WriteLogData */
                LoggingSmState = WriteLogData;
                /* Loggin started */
                LogStatus = Logging;
            }
    }
}

```

(Continua)

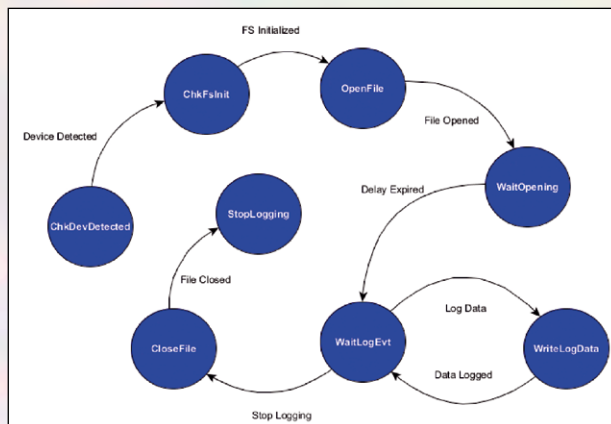


Fig. 14 - Rappresentazione grafica della macchina a stati LoggingStateMachine.

- `size_t FSfwrite(const void *data_to_write, size_t size, size_t n, FSFILE *stream);` scrive l'array di dati passato sul file puntato dallo stream;
- `int FSfclose(FSFILE *fo);` Chiude il file passato tramite il riferimento fo.

Tramite queste API di alto livello è relativamente facile scrivere un'applicazione di datalogging. Come appare dal listato, sono stati identificati 8 stati, definiti dal tipo enumerativo riportato nel **Listato 7**. I primi due stati della macchina controllano se la periferica è collegata ed è stata correttamente enumerata e se il file system è inizializzato. Successivamente viene aperto il file, utilizzando la API `FSfopen` e viene generato un ritardo per dare il tempo al client driver di eseguire l'operazione. A questo punto la macchina si mette in attesa degli eventi di log provenienti dal task `Adc`. Se viene ricevuto un evento di log si salva il dato e si torna nello stato di ascolto (`WaitLogData`) altrimenti si chiude e si va nello stato conclusivo (`StopLogging`).

Come si può notare l'implementazione (riportata in forma grafica in **Fig. 14**) è piuttosto semplice, questo perché sfruttiamo appieno le potenzialità delle infrastrutture su cui l'applicazione si appoggia (lo scheduler e lo stack USB host), che fanno la maggior parte del "lavoro sporco", permettendo all'applicazione di operare ad alto livello. Per testare la nostra applicazione è sufficiente scaricare l'applicazione sulla demoboard utilizzata a supporto del corso, collegare una chiavetta USB sulla porta USB host e dare alimentazione. L'applicazione, una volta enumerato il dispositivo USB inizierà l'attività di logging, che

Listato 6 (segue)

```

    }
    else if (ReceiveEvt(&CloseLogFile))
    {
        /* Jump to CloseFile */
        LoggingSmState = CloseFile;
        /* Logging stopped */
        LogStatus = NotLogging;
    }
    else
    {
        /* Do nothing */
    }
    break;

case WriteLogData:
    /* Write data to USB key */
    FSfwrite(AdcDataBuffer,1,ADC_BUFFER_LENHT,LogFile);
    /* Put new line */
    FSfwrite("\r\n",1,2,LogFile);

    /* Jump back to WaitLogEvent */
    LoggingSmState = WaitLogEvent;
    break;

case CloseFile:
    /* Close file */
    FSfclose(LogFile);
    /* Jump to Stop */
    LoggingSmState = StopLogging;
    break;

case StopLogging:
    /* Do nothing */
    break;

default:
    break;
}
}

```

Listato 6 - Implementazione della macchina a stati LoggingStateMachine.

Listato 7

```

/* Logging state machine state typedef */
typedef enum LoggingSmStateEnum
{
    CheckDeviceDetected,
    CheckFsInitialized,
    OpenFile,
    WaitOpening,
    WaitLogEvent,
    WriteLogData,
    CloseFile,
    StopLogging,
} LoggingSmStateType;

```

Listato 7 - Typedef per definire il tipo enumerativo LoggingSmStateType.

si concluderà dopo un certo periodo di tempo (circa 30 secondi). In Fig. 15 è riportato il grafico dell'acquisizione generato con Excel.

CONCLUSIONI

In questa puntata abbiamo analizzato in dettaglio la parte host dello stack USB di microchip e costruito un'applicazione di datalogging di una certa complessità. Nella prossima ed ultima puntata analizzeremo un altro complesso e potente stack di Microchip, ossia lo stack TCP/IP.

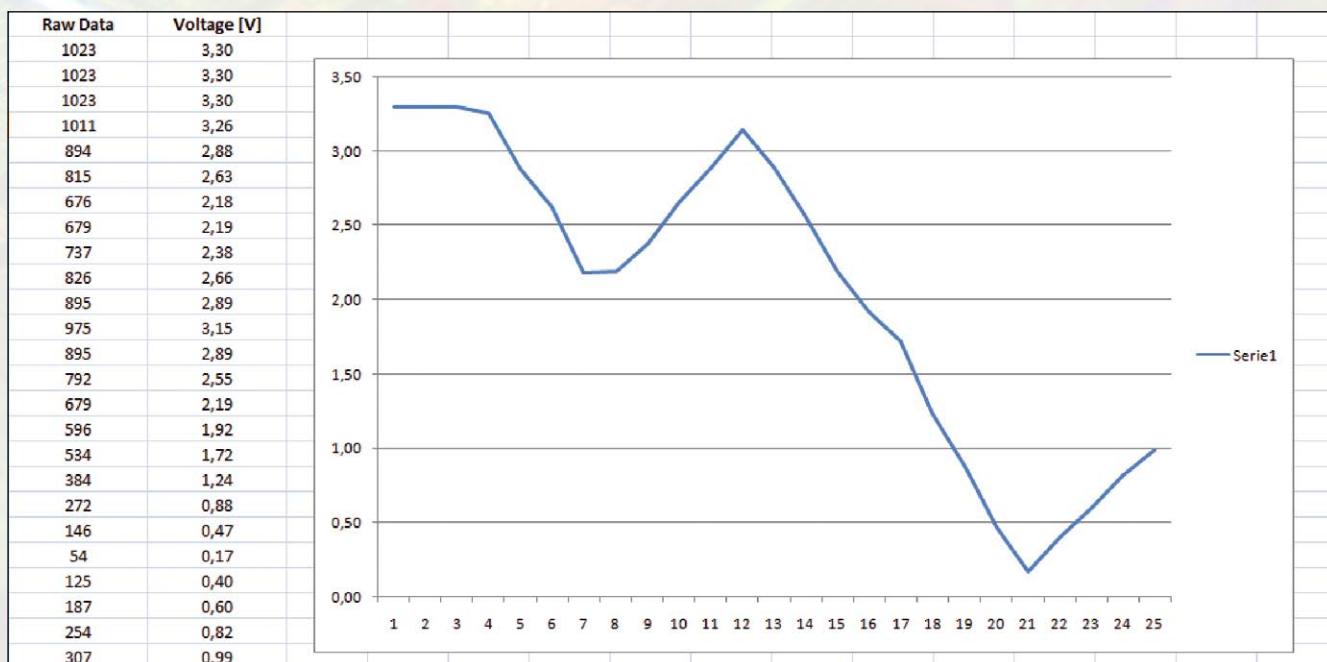


Fig. 15 - Grafico dell'acquisizione generato con MS Excel.