



Plastico dinamico controllato da LabVIEW tramite Arduino Mega, che riproduce l'attrazione "Jungle Rapids" presente nel parco di divertimenti Gardaland.

IL LUNA PARK GESTITO DA ARDUINO

..... di ALESSIO PEROTTINO

Durante l'estate milioni di persone affollano i parchi di divertimento, in Italia come in Europa. Per mantenere vivo l'interesse verso le attrazioni che caratterizzano l'uno o l'altro parco, il settore tecnico di ciascuno è alla costante ricerca di novità che stupiscano sempre di più i visitatori e che garantiscano un certo afflusso di pubblico negli anni a venire. Dai ghiacci dell'artico alle antiche rovine della città di Atlantide, dai set cinematografici dei più celebri film ad attrazioni spericolate: questi i mix vincenti dei parchi di divertimento, che spiegano il grande interesse del pubblico. Nei parchi di divertimento l'elettronica occupa un ruolo fondamentale: ogni attrazione viene gestita da uno o più "cervelli" elettronici (tipicamente dei PLC) che governano i flussi in ingresso (quasi sempre segnali di sensori o comandi impartiti dall'operatore tramite il pannello di controllo) e i flussi dati

in uscita (riguardanti motori, attuatori, luci, audio ed effetti speciali).

Alessio Perottino, uno studente appassionato di elettronica e parchi di divertimento, ha deciso di portare come progetto per il suo esame di maturità la riproduzione in scala dell'attrazione "Jungle Rapids" di Gardaland (da lui denominata "Mekong Rapids") ricostruendo anche l'impianto elettrico e il software di controllo. In questo articolo ripercorreremo, passo per passo, il suo lavoro.

La tipologia di attrazione "Jungle Rapids" di Gardaland è presente in molti parchi di divertimento nel mondo, appartiene alla famiglia delle "raft ride" (oppure "river rapids") e il suo scopo è quello di simulare -in maniera in vero realistica- un percorso su gommoni in mezzo alle rapide di un fiume: gli ospiti dell'attrazione vengono fatti salire, per mezzo di una piattaforma rotante, che funge da stazione,



Fig. 1 - L'attrazione Jungle rapids di Gardaland.

su gommoni di forma circolare in grado di ospitare nove persone ciascuno. Lasciata la zona di imbarco, il gommone percorre il corso d'acqua artificiale, attraversa l'interno di un vulcano-modello in cui sono presenti degli effetti di luce per simulare la lava, supera delle cascate e dei giochi d'acqua, per poi arrivare in un piccolo laghetto artificiale dove è stato riprodotto un vortice d'acqua che fa avanzare i gommoni; alla fine del percorso i gommoni vengono allineati tramite dei cilindri rotanti per poi imboccare una rampa in salita che li riporta in stazione. Tutto il percorso, che contiene oltre 1.200 metri cubi d'acqua e 23 gommoni, è leggermente in discesa per creare una corrente che porta i gommoni dall'uscita della stazione fino alla

fine (**Fig. 1**); per riportare l'acqua a livello della stazione sono presenti diverse pompe industriali che lavorano senza sosta, mentre per attivare gli effetti durante il percorso sono impiegati sensori di prossimità e fotocellule; il PLC per il controllo dei sensori più utilizzato all'interno dei parchi di divertimento e in questa attrazione è il Siemens S7.

Affascinati da questo genere d'attrazione abbiamo pensato di portarcene a casa una versione da tavolo, ovvero di realizzarne un modello sufficientemente d'effetto.

IL PROGETTO

Il prototipo del percorso è stato ricostruito in scala 1:100 prima con il software SketchUp e poi su dei fogli da disegno tecnico;

il plastico, rinominato "Mekong Rapids", è stato realizzato con polistirolo espanso. L'intera struttura è in scala 1:100, progettata basandosi su immagini satellitari e da numerose foto scattate in loco per ricreare anche i più piccoli particolari.

Ricalcando il percorso dalle mappe aeree, è stato realizzato il progetto su carta, completato il quale l'autore è passato ad intagliare il percorso in un blocco di polistirolo 150x80 centimetri ricreando il dislivello che fa scorrere l'acqua. In questa fase ha preso la decisione di escludere una parte del percorso originale perché avrebbe allungato la struttura del plastico oltre i due metri, rendendone difficile il trasporto; non dimentichiamo che doveva essere portato a scuola per la dimostrazione...

In **Fig. 2** possiamo vedere il plastico durante le varie fasi di lavorazione che portano al lavoro quasi completo.

Il plastico del Mekong Rapids è stato realizzato su tre lastre di polistirolo adeguatamente intagliate e scavate per ricreare i canali in cui far scorrere l'acqua. Dopo aver ricostruito il percorso, l'autore ha iniziato a scavare il canale utilizzando un apposito



Fig. 2 - Preparazione della base del plastico, ottenuta scavando del polistirolo espanso.

strumento, chiamato “traforo per polistirolo” (tipo il modello cod. VTFC di Futura Elettronica), composto da una bacchetta in ferro riscaldata che intaglia il materiale evitando la creazione delle tanto odiate (perché finiscono dappertutto...) palline, levigandone i bordi.

Per renderlo dinamico sotto ogni aspetto, il plastico è stato equipaggiato con:

- 2 pompe acquatiche;
- 2 motoriduttori 12 V 14 rpm;
- 4 servomotori a 5 V;
- 30 LED bianchi a luce calda;
- 1 fotocellula (laser + fotoresistenza);
- 2 schede Arduino Mega.

Dopo aver dipinto il fondale del plastico si è passati a realizzare la scenografia dell'attrazione, che è stata stampata interamente con la nostra 3Drag grazie alla collaborazione con l'I.T.I. “L. Cobianchi” (Fig. 3).

Prima di posizionare la scenografia l'autore ha proceduto al posizionamento e al cablaggio di tutti i componenti elettronici: due pompe per acquari, utilizzate per riportare l'acqua alla stazione di partenza, due motoriduttori da 12V 14 rpm per muovere la piattaforma rotante e la rampa che riporta i gommoni in stazione, un nebulizzatore ad ultrasuoni (utilizzato assieme ad una ventola per simulare l'uscita del fumo dalla cima del vulcano), quattro servomotori da 5V per simulare la caduta di alberi e rocce durante il percorso, una fotocellula (costruita impiegando un diodo laser e una fotoresistenza), trenta LED a luce calda per simulare l'illuminazione notturna del plastico e due Arduino Mega; il primo si occupa di gestire le periferiche principali dell'attrazione ed è direttamente interfacciato con il software LabVIEW (utilizzato per



Fig. 3 - Montaggio dell'elettronica e della parte idraulica.



Fig. 4 - Il Plastico completato.

ricreare il Pannello Frontale, che costituisce il pannello di controllo virtuale), mentre il secondo serve per gestire gli effetti speciali durante il percorso e il deposito gommoni a lato della stazione (dove vengono parcheggiati i gommoni durante la chiusura dell'attrazione).

Tutto il plastico è alimentato tramite un alimentatore che eroga 24 Vcc da cui poi, tramite dei regolatori lineari di tensione (7812 e 7805), sono ricavate le tensioni (12V e 5V) necessarie al funzionamento di tutti gli elementi del plastico; tutti questi elementi (tranne il nebulizzatore

a ultrasuoni per il vulcano) sono gestiti dal primo Arduino Mega tramite relé e transistor.

Abbiamo detto che il primo Arduino Mega è stato utilizzato per interfacciare l'elettronica del plastico al pannello di controllo dell'attrazione ricreato con il software LabVIEW (Fig. 5); per realizzare questa interfaccia è stato caricato in Arduino Mega uno sketch che rimane in ascolto sulla seriale, mentre, tramite delle SubVI, sono stati associati i pin di Arduino ai pulsanti presenti sul pannello di controllo: con questo metodo diventa molto semplice controllare direttamen-



Fig. 5 - Il Pannello Frontale in LabVIEW.

te ogni singolo pin di Arduino utilizzando l'interfaccia seriale. Il software creato è stato frutto di attente ricerche nel settore dei parchi di divertimento e più specificatamente nella tipologia dell'attrazione "river rapids", esaminando tutte le operazioni che vengono svolte per la gestione e la manutenzione.

Si è così deciso di strutturare il programma in tre macro modalità: *attrazione chiusa al pubblico*, *attrazione in manutenzione ordinaria e straordinaria* e, la modalità più importante, *attrazione aperta al pubblico* (tutti i dispositivi presenti azionabili).

Esaminiamo ora le tre modalità partendo proprio da quest'ultima: sul pannello di controllo è possibile azionare tutti i dispositivi contraddistinti dal colore verde come la pedana rotante per l'imbarco, la salita per riportare alla quota di partenza i gommoni e tutti gli effetti speciali.

La gestione di tali effetti avviene tramite uno scambio di informazioni tra i due Arduino Mega, i quali vanno a creare un sistema retroazionato che permette di sapere in qualunque momento in quale stato sono i singoli attuatori degli effetti speciali e se ci sono eventuali errori o guasti; l'azionamento degli effetti speciali prende il via da un segnale proveniente dalla fotocellula: quando il gommone passa attraverso il fascio luminoso, il programma fa partire la sequenza di avvio delle scene.

Al termine della sequenza degli effetti, il primo Arduino attende il consenso dal secondo per poter riavviare la sequenza al passaggio del successivo gommone. In qualsiasi momento è possibile azionare il sistema di emergenza: nel caso in cui venga premuto l'apposito pulsante, tutta l'attrazione si ferma lasciando accese solamente le pompe, per evitare

che il percorso si svuoti nel lasso di tempo tra il lancio dell'allarme e la risoluzione del problema.

Nella modalità di manutenzione ordinaria è possibile utilizzare tutti i dispositivi contraddistinti dal colore giallo: il nastro di risalita dei gommoni e i servomotori che aprono e chiudono il deposito; con la modalità di manutenzione straordinaria invece è possibile controllare singolarmente ogni dispositivo del plastico per verificare ed eventualmente sistemare problemi circoscritti ad un singolo elemento.

La modalità "attrazione chiusa al pubblico" è stata la più semplice da realizzare, in quanto tutti gli attuatori sono in stato di riposo e tutti gli elementi dell'attrazione vengono disattivati.

Analizzando il **Listato 1**, che ci mostra un'estratto dello sketch di Arduino, possiamo notare la gestione degli effetti che vengono attivati ogni volta che la fotocellula rileva il passaggio di un gommone. Nello specifico, all'inizio della funzione loop possiamo vedere il costrutto di rilevazione dell'impulso: quando un gommone passa davanti alla fotocellula viene settata la variabile "conta" al valore 1. Da quel momento in avanti vengono attivati in sequenza gli effetti del vulcano (uscita fumo dalla cima) e la caduta degli alberi e delle rocce, solamente se questi sono stati singolarmente abili-

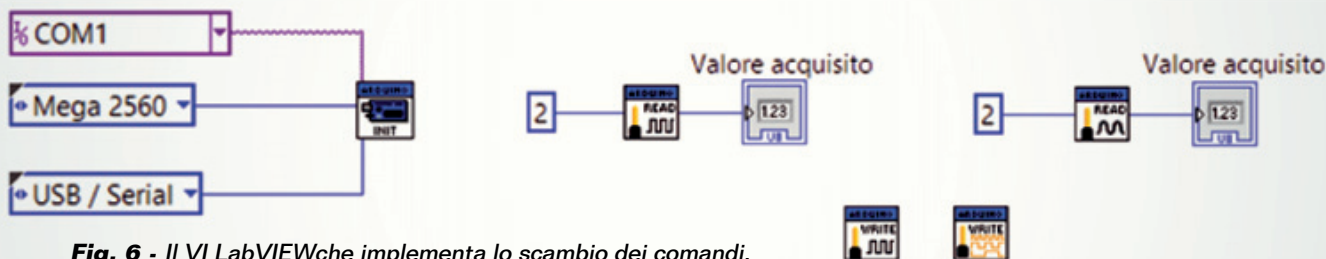


Fig. 6 - Il VI LabVIEW che implementa lo scambio dei comandi.

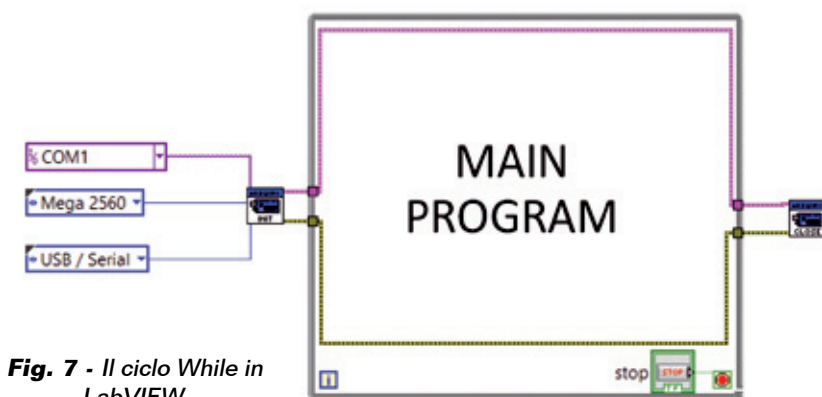


Fig. 7 - Il ciclo While in LabVIEW.

tati dal pannello di controllo in LabVIEW.

Ogni volta che viene eseguito un effetto, come la caduta delle rocce o degli alberi, questi vengono ripristinati subito dopo il passaggio del gommone.

IL SOFTWARE

La programmazione delle due schede "Mega" è stata articolata su strade diverse, che andremo ad analizzare di seguito. Su ogni Arduino viene caricato un firmware che permette la comunicazione con LabVIEW utilizzando la trasmissione seriale dei comandi. Vediamo dunque il software delle due schede Arduino.

ARDUINO MEGA (b0)

Gestione Attrazione

La Arduino Mega principale (chiamato b0, ossia brain 0), atto alla gestione delle periferiche principali, è configurato con il software LabVIEW, adeguatamente interfacciato al Personal

Computer per ricreare virtualmente un vero e proprio pannello di controllo come quello che l'operatore del parco di divertimenti manovra. Utilizzando dei SubVI forniti dal Plugin Arduino presente in LabVIEW è possibile associare i piedini fisici del microcontrollore ad azioni di comando eseguite all'interno del software. È sufficiente impostare i parametri di comunicazione tra Personal Computer e Arduino, come porta seriale, tipo di microcontrollore e tipologia di comunicazione, per consentire ai due di scambiarsi dati e comandi molto semplicemente (Fig. 6). Per garantire uno scambio continuo di informazioni è necessario inserire il programma all'interno di un While Loop, ossia un ciclo che viene interrotto solo dal manifestarsi di una condizione o da una variabile (in questo caso dal pulsante di stop) che interrompe la comunicazione e spegne il programma (Fig. 7).

ARDUINO MEGA (b1)

GESTIONE EFFETTI SPECIALI e DEPOSITO

A differenza dell'Arduino Mega b0, il b1 è stato programmato con il linguaggio C, che è caratterizzato da una serie di comandi e librerie che, una volta richiamati, attuano una certa azione sui pin dell'Arduino.

Il programma del b1 è stato creato per gestire le interazioni in movimento con i gommoni (effetti speciali) e le aperture del deposito.

Gli effetti speciali sono caratterizzati principalmente dagli elementi di seguito descritti.

- Vulcano: struttura intagliata nel polistirolo, che contiene una vasca con mezzo litro d'acqua. Qui è inserita una Ultrasonic Fogger Machine (macchina della nebbia ad ultrasuoni) che quando viene azionata investe l'acqua con vibrazioni acustiche ad alta frequenza (frequenza ultrasonica, appunto) creando una nebbia densa e fredda; insomma, funziona come un aerosol sonico. Per simulare l'eruzione, una ventola risucchia il fumo e lo spinge all'esterno.
- Alberi e rocce in caduta: tramite un segnale PWM fornito dalle uscite analogiche di Arduino, si comanda l'apertura angolare dei servomotori da modellismo che simulano la caduta degli alberi e delle rocce.



Fig. 8 - Dettagli della costruzione del vulcano.

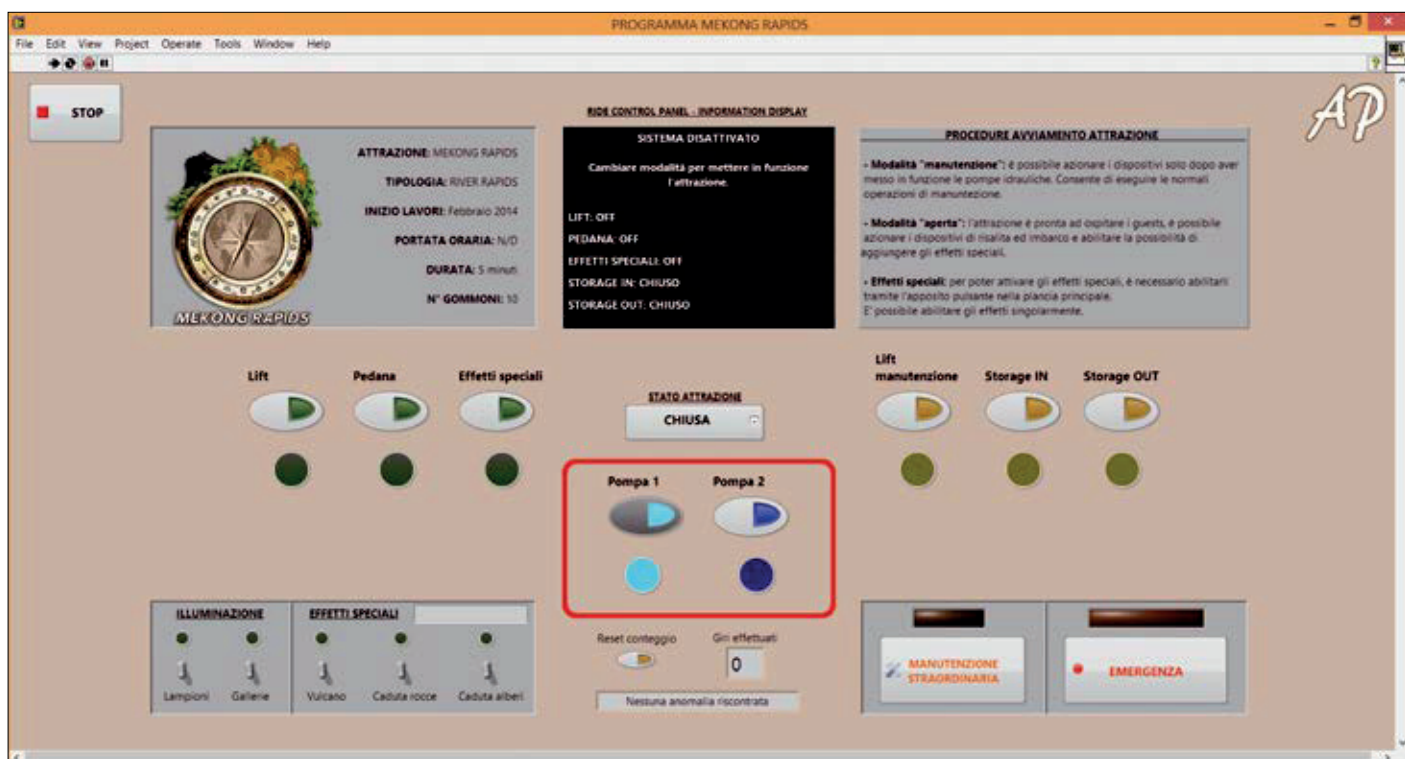


Fig. 9 - Comando delle pompe idrauliche.

ACCENSIONE DELL'ATTRAZIONE

Dopo aver eseguito con successo tutte le operazioni di inizializzazione del programma di gestione, è necessario abilitare alcuni dispositivi fondamentali per il funzionamento dell'attrazione, come ad esempio le pompe idrauliche (Fig. 9).

Il software è programmato in modo che, nel caso vi fosse

un'anomalia alle pompe che ne comporti il totale spegnimento, si attivi la situazione d'allarme che disabilita tutti i dispositivi mobili dell'attrazione; ciò, nella realtà dei parchi di divertimento permette un'eventuale evacuazione degli ospiti in totale sicurezza. Siccome il plastico del Mekong Rapids deve funzionare in maniera più realistica possibi-

le, prima di qualsiasi operazione ordinaria (manutenzione o con attrazione aperta), è necessario abilitare almeno una delle due pompe di circolazione dell'acqua. Attivando le due pompe è possibile abilitare le varie modalità di funzionamento tramite l'apposito menu a tendina "Stato attrazione" al centro della pagina.

Le modalità di gestione principa-

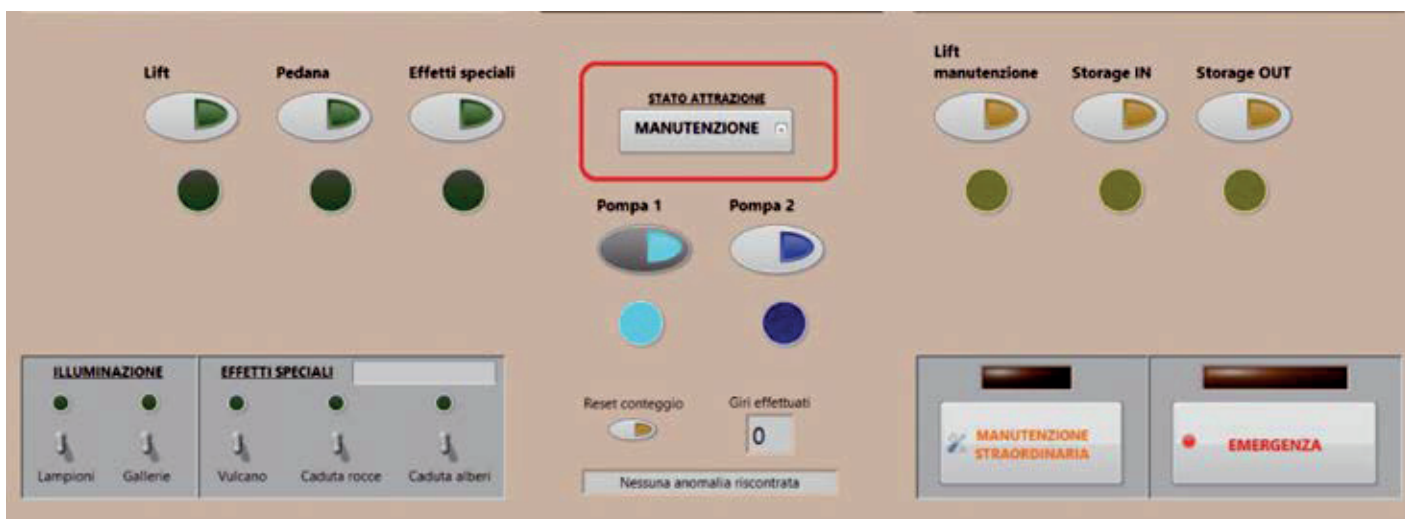


Fig. 10 - Manutenzione.

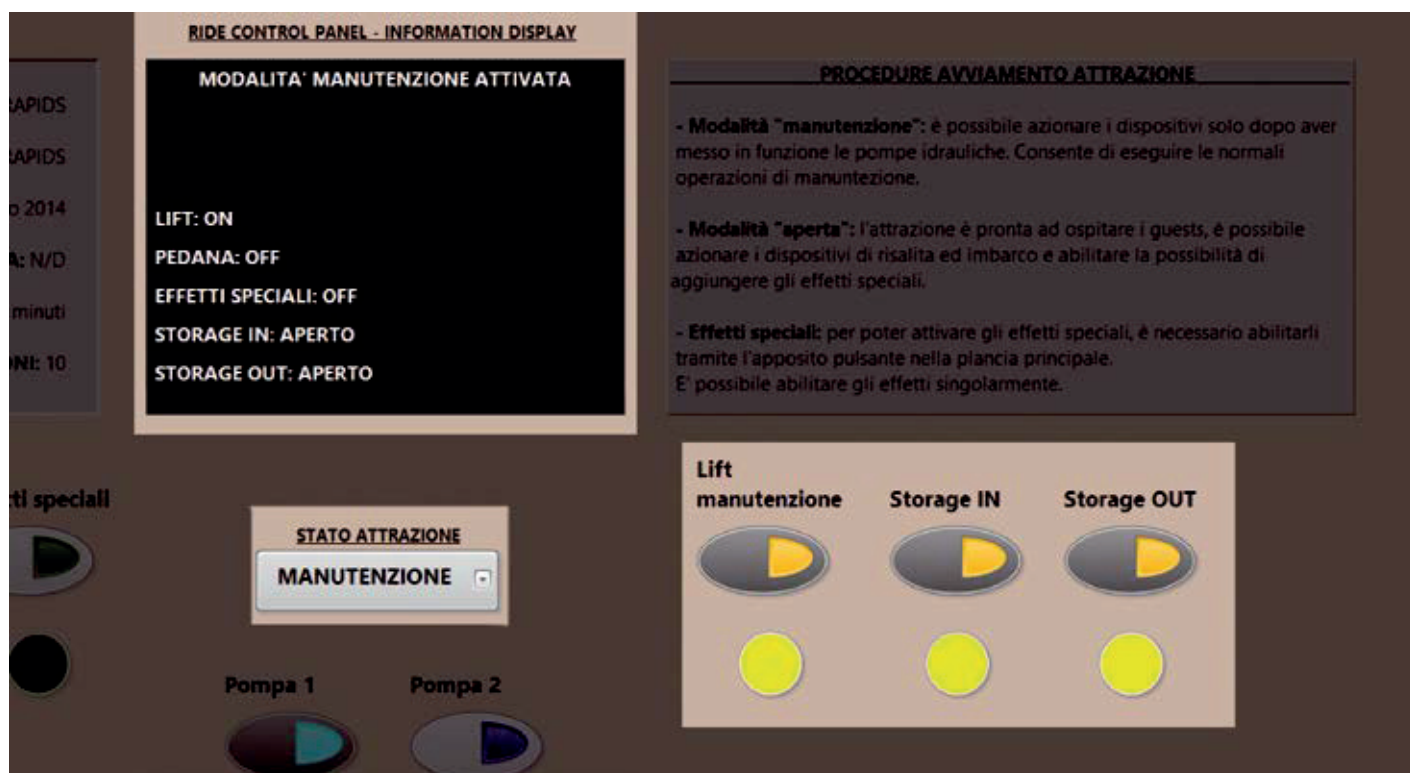


Fig. 11 - Manutenzione straordinaria.

li sono tre: chiusa, manutenzione e aperta.

OPERAZIONI MANUTENZIONE ORDINARIA

Come anticipato nel paragrafo precedente, una delle tre modalità disponibili è quella "manutenzione ordinaria", utilizzata per tutte le operazioni pre/post-apertura dell'attrazione quali

l'apertura/chiusura del deposito gommoni (storage) e l'attivazione della rampa di risalita (lift). Selezionate dal menu a tendina al centro della pagina la voce "manutenzione" (Fig. 10). In questa modalità si potranno utilizzare principalmente i tre dispositivi posti sul lato destro dello schermo contraddistinti dal colore giallo:

- Lift manutenzione = attivazione manuale della rampa di risalita dei gommoni;
- Storage IN = apertura/chiusura ingresso deposito gommoni;
- Storage OUT = apertura/chiusura uscita deposito gommoni.

L'apertura dell'ingresso e dell'uscita dello storage avviene in sequenza, quindi mai in simultanea per motivi di sicurezza: nell'area di manutenzione può attivarsi un solo motore alla volta. Cliccando sugli appositi pulsanti, vengono attivati i dispositivi e segnalato il loro stato attraverso un indicatore luminoso sul pulsante e in formato testo sul display al centro della schermata.

OPERAZIONI DI MANUTENZIONE STRAORDINARIA

Per tutte le operazioni di manutenzione straordinaria (da eseguire previa richiesta scritta del perito manutentore), è consentito utilizzare l'intero sistema con comandi manuali; all'occorrenza



Fig. 12 - Aprire l'attrazione.

L'elettronica nei parchi di divertimento



Gestire sistemi di automazione come quelli che governano le attività di un parco divertimenti è un compito adatto ai PLC; il PLC (Controllore a Logica Programmabile) è un controllore specializzato in origine nella gestione o controllo dei processi industriali. Il PLC esegue un programma ed elabora segnali digitali ed analogici provenienti da sensori e diretti agli attuatori presenti in un impianto. Ciascun PLC è composto da elementi base, come l'alimentatore, le schede di interfacciamento dei segnali (input/output) e da una CPU (il cervello del PLC) contenente una scheda di memoria EPROM destinata al salvataggio del programma. Ogni PLC è poi configurabile a seconda delle esigenze aggiungendogli schede particolari per infinite tipologie di sistemi. I linguaggi di programmazione principali sono:

- Ladder Diagram: programmazione visuale tramite l'uso di simboli elettrici;

- Instruction List: linguaggio di alto livello, diffuso nelle maggior ditte di programmazione permette una programmazione più strutturata rispetto al ladder.

All'interno dei Parchi divertimento, il PLC usato più frequentemente è il Siemens S7 ed è impiegato in moltissimi ambiti: apparati di sicurezza del Parco, gestione del sistema interno di ciascuna attrazione, programmazione live show ecc.

In questo riquadro vedete uno dei cinque armadi di controllo dell'attrazione Dive Coaster: il dive coaster.

I circuiti presenti nel rack si occupano della stazione di partenza, una montagna russa con 45 metri di discesa verticale (87°); i treni sono formati da 3 vetture ed essendo floorless, ovvero privi di pavimento, la stazione è equipaggiata con delle pedane mobili pneumatiche che vengono azionate una volta

za, quindi, bisogna fare clic sul pulsante del pannello di controllo "manutenzione straordinaria" per abilitare tutti i dispositivi manualmente (Fig. 11).

OPERAZIONI DI APERTURA ATTRAZIONE

Dopo aver completato le opera-

zioni di manutenzione (se necessarie), l'operatore ha il compito di rendere operativa l'attrazione e fruibile dal pubblico. Selezionando la modalità "aperta" si potranno abilitare i seguenti dispositivi:

- Lift = impianto di risalita gomme;

- Pedana = stazione d'imbarco rotante;
- Effetti speciali = abilita il menu effetti speciali sul pannello.

Se si vuole aprire l'attrazione, nel menu a tendina al centro del pannello di controllo, selezionare la voce "aperta" (Fig. 12).

Tutti i dispositivi, se non già azionati in precedenza, sono in stato di riposo (spenti).

È possibile quindi azionarli manualmente premendo i pulsanti di abilitazione.

Di norma per rendere pienamente operativa l'attrazione, occorre abilitare tutti i dispositivi di colore verde.

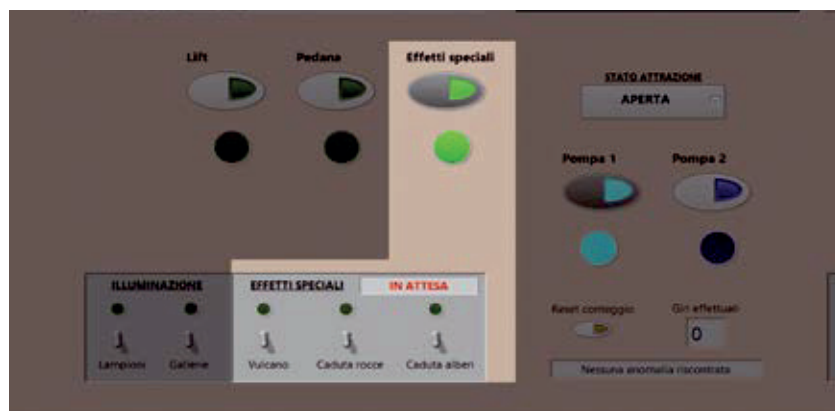


Fig. 13 - Gestione degli effetti speciali.

che il convoglio è nella corretta posizione per permettere ai visitatori di scendere in totale sicurezza. In questo caso il PLC è impiegato nel posizionamento corretto del treno, pilotando i motori brushless del sistema "transfer" (motori con ruote che agganciano il treno facendolo muovere in avanti o indietro), allineandolo ai cancelletti di ingresso/uscita. Sui binari all'interno della stazione sono posizionate sei coppie di sensori, due coppie per ciascuna vettura, poste sui due lati: queste forniscono l'input al PLC comunicandogli quando il treno è nella posizione corretta e segnalandolo tramite un indicatore luminoso sul pannello all'operatore. Automaticamente il treno si ferma, in attesa dell'attivazione delle pedane. L'operatore, tramite l'apposito pulsante, invia l'ordine al PLC di attivare il compressore che immetterà nel sistema pneumatico l'aria necessaria alle pedane per potersi aprire.

Per garantire a tutti gli ospiti la massima sicurezza è necessario un sistema di sicurezza che monitora tutto ciò che succede a bordo dell'attrazione; all'interno delle attrazioni più complesse, come i roller coaster, vengono utilizzati molteplici sensori per altrettanti utilizzi e scopi. La regola principale del sistemista è quella di porre almeno una coppia di sensori per ogni singola postazione: ciò crea ridondanza, mantenendo la massima sicurezza anche in caso di guasto di uno dei sensori. In figura, è rappresentato il sensore di prossimità utilizzato per la verifica del passaggio del treno all'interno delle sezioni del tracciato. Le linee principali di trasmissione dati nel parco numero 1 in Italia sono realizzate in fibra ottica, un particolare cavo che sfrutta la rifrazione della luce per trasferire a 100Mbit/sec. Poiché usa la luce per trasmettere i segnali, il cavo in fibra ottica non è influenzato da disturbi elettromagnetici (EMI)



o a radiofrequenza (RFI). Tutti i segnali sono convertiti in impulsi luminosi quando entrano nel cavo, e sono riconvertiti in segnali elettrici quando escono dal cavo. Ciò significa che il cavo in fibra ottica è in grado di fornire segnali che si mantengono più precisi, è in grado di raggiungere distanze maggiori e ha una maggiore larghezza di banda rispetto al cavo di rame o di altri metalli. Per mantenere distanziati i treni che percorrono simultaneamente il Roller Coaster è impiegato un sistema di blocco composto da sezioni di rotaia dotate di sistemi di

start e stop basati su catene, ruote, freni. Due treni non possono trovarsi nello stesso blocco, quindi in caso di guasto che blocca un convoglio quello immediatamente seguente viene arrestato. Il tutto è gestito da un Siemens che acquisisce i dati da sensori di prossimità elettromagnetici che rilevano una piastra metallica sporgente dal fondo del convoglio. Utilizzando un interruttore di prossimità il PLC può contare il numero di vagoni e capire quando un treno è passato integro e totalmente da un blocco.

EMERGENZA

Premendo il tasto emergenza posto in basso a destra, tutti i dispositivi saranno immediatamente bloccati e gli indicatori luminosi emetteranno luce lampeggiante. Terminata la situazione d'emergenza, cliccare nuovamente sul pulsante. Tutti i dispositivi torneranno automaticamente a funzionare.

EFFETTI SPECIALI

Sul pannello di controllo sono presenti il pulsante di abilitazione effetti (al centro) e il menu di attivazione manuale dei singoli effetti speciali in basso a sinistra (Fig. 13). Per abilitare il menu effetti speciali, è necessario premere il pulsante al centro del pannello "effetti speciali". Abilitato il menu degli effetti speciali,

il sistema di gestione è portato "in attesa", come indicato in rosso sul display.

Qui è possibile attivare/disattivare i singoli effetti:

- Vulcano = emissione di fumo e accensione LED rossi;
- Caduta rocce = caduta di rocce dalla terrazza panoramica;

- Caduta alberi = caduta albero post-ponte.

Selezionando gli effetti, al passaggio del gommone attraverso la fotocellula sarà attivata la sequenza degli effetti speciali ed il sistema verrà portato allo stato "in esecuzione".



Fig. 14 - Comando dell'illuminazione.

Listato 1

```
void loop() {  
  
    servo_alberi.write(5);  
    servo_rocce.write(15);  
  
    //Inizio rilevazione impulso gommone  
  
    adesso = digitalRead(12);  
    if (precedente == 0 && adesso == 1) {  
        conta += 1;  
        precedente = adesso;  
    } else {  
        precedente = adesso;  
    }  
    digitalWrite(24, conta);  
    Serial.println(conta);  
    delay(70);  
  
    //Fine rilevazione impulso gommone  
  
    //-----INIZIO PROGRAMMA EFFETTI -----  
  
    if (conta >= 1) { //Se impulso = 1  
  
        //-----VULCANO-----  
  
        p_vulcano=digitalRead(7);  
        if (p_vulcano == 1) { //Se pulsante LabVIEWvulcano =1  
            digitalWrite(26, HIGH); //Attiva vulcano  
            Serial.println("Attiva vulcano");  
        }  
        delay(d_vulcano); //Attendi  
        digitalWrite(26, LOW); //Spegni vulcano  
        Serial.println("Disattiva vulcano");  
        Serial.println(" ");  
        delay(2000);  
  
        //-----ALBERI-----  
  
        p_alberi=digitalRead(8);  
        if (p_alberi == 1) { //Se pulsante LabVIEWalberi =1  
            Serial.println("caduta alberi");  
            pos_alberi=caduta_alberi(); //Richiamo funzione caduta alberi  
        } else {  
            pos_alberi=5;  
        }  
        Serial.print("pos alberi= ");  
        Serial.println(pos_alberi);  
        delay(d_alberi); //Attendi  
        if (pos_alberi==50){  
            salita_alberi(); //Richiamo funzione salita alberi  
            Serial.println("salita alberi");  
            Serial.println(" ");  
        }  
        delay(2000);  
  
        //-----ROCCHE-----  
  
        p_rocce=digitalRead(9);  
        if (p_rocce == 1) { //Se pulsante LabVIEWrocce = 1  
            Serial.println("caduta rocce");  
            pos_rocce=caduta_rocce(); //Richiamo funzione caduta rocce  
        } else {  
            pos_rocce=15;  
        }  
        Serial.print("pos rocce= ");  
        Serial.println(pos_rocce);  
        delay(d_rocce); //Attendi  
        if (pos_rocce==100){  
            salita_rocce(); //Richiamo funzione salita rocce  
            Serial.println("salita rocce");  
            Serial.println(" ");  
        }  
        conta = 0;  
        digitalWrite(24,0);  
    }  
  
    //-----FINE PROGRAMMA EFFETTI -----  
}
```

Solo a sequenza terminata il sistema tornerà "in attesa", pronto a rieseguire il programma al passaggio del gommone.

ILLUMINAZIONE ARTIFICIALE

Le luci artificiali all'interno dell'area di Mekong Rapids si dividono principalmente in due parti:

- Gallerie vulcano = illuminazione interna alle gallerie;
- Lampioni = illuminazione crepuscolare esterna.

Consigliamo di abilitare entrambe le illuminazioni come si fa nella realtà per evitare situazioni di disagio con gli ospiti più suscettibili e facilmente spaventabili da aree scarsamente illuminate (in special modo all'interno delle gallerie). I lampioni dispongono dell'interruttore crepuscolare, che li accende variandone l'intensità a seconda della luminosità dell'ambiente. È possibile attivare/disattivare le illuminazioni in qualsiasi modalità.

ALTRE FUNZIONI

Oltre ai comandi e indicatori sinora descritti, sul pannello di controllo è disponibile un contatore dei giri totali effettuati dai gommoni del plastico; il conteggio viene incrementato al passaggio del gommone attraverso la fotocellula degli effetti speciali.

CONCLUSIONI

Il progetto non è solo un'esercitazione con finalità didattiche ma un modo -come è in genere per il modellismo dinamico- per imparare attraverso la sperimentazione le tecnologie e valutare le problematiche tecniche insite in realtà, che nel caso specifico sono quelle del funzionamento di un'attrazione da parco di divertimenti, ma che si estendono al mondo dell'automazione. ■

OCTOPUS: LA PIOVRA CHE SUONA IL PIANO

..... di GABRIELE DAGHETTA

Con Fishino e il driver Octopus facciamo suonare una pianola giocattolo che riprodurrà motivi musicali nati con le suonerie Nokia.

Da parecchi anni le tastiere elettroniche, in special modo i sintetizzatori musicali, hanno la possibilità di suonare "da soli" brani, opportunamente pilotati tramite l'interfaccia MIDI o riproducendo file in formato standard

caricati da supporti esterni di memorizzazione di massa. Ma quello che ci siamo proposti di fare è far suonare motivi a nostra scelta a strumenti musicali più abbordabili, come piccole tastiere musicali e pianole giocattolo

acquistabili nei centri commerciali per poche decine di euro, che per evidenti ragioni non dispongono di interfacce dati o slot per supporti di memorizzazione di massa e che quindi non possono essere pilotate elettronicamente. In questi casi l'unica soluzione è farle suonare in maniera tradizionale, agendo sui tasti con qualcosa che trasformi le note dei brani musicali in comandi meccanici. Insomma, un'interfaccia da elettronica a meccanica. Questo lo abbiamo fatto e ha





Fig. 1
Il sistema
carrozzato con
le tre pivre e
applicato alla
pianola.

dato origine al progetto descritto in queste pagine, ossia un gadget che abbiamo sviluppato affinché possa essere utilizzato in varie occasioni, come ad esempio da demo in una vetrina o nel reparto giocattoli di un negozio, come pure da intrattenimento per grandi e piccoli in eventi e manifestazioni, per esempio, di robotica applicata.

Il progetto è nato dal desiderio di poter riprodurre con una tastiera musicale low cost (una tastiera giocattolo) delle musiche standard, però suonandole premendo effettivamente i tasti, pur senza metterci un dito. Abbiamo quindi pensato a una soluzione automatizzata che consiste in un piccolo robot che suona per noi i tasti della pianola. Le musiche che abbiamo scelto non sono file audio qualsiasi ma, per praticità, abbiamo pensato ai ring tone

(vale a dire le suonerie) sviluppati dalla Nokia e definiti RTTTL (acronimo di Ring Tone Text Trasfer Language), “suonati” dai cellulari del costruttore.

La nostra piattaforma robotica, che va applicata alla pianola adattandosi a forma e distanza dei tasti di quest’ultima, è composta da un “cervello” basato sulla ormai popolare scheda Fishino Uno e da un sistema di attuazione realizzato col driver Octopus, già presentato nel fascicolo 203 di Elettronica In. Il tutto è stato “carrozzato” in stile fondale marino, con il risultato visibile nella Fig. 1.

La gestione dei toni è affidata ad uno sketch appositamente creato per la nostra board Fishino Uno, mentre il controllo fisico dei tasti è ottenuto mediante un sistema elettromeccanico composto da 24 servomotori (cod. SERVO206 del catalogo Futura Elettronica, reperibili su www.futurashop.it) gestiti da due shield Octopus (codice OCTOPUS Futura Elettronica) a cui sono collegati gli otto tentacoli di tre amene pivre color rosso stampate in PLA con la stampante 3D, azionati dai servo mediante un semplice ma preciso sistema di leveraggio (Fig. 2). Tramite i 24 servo è possibile gestire due ottave (sulla pianola da

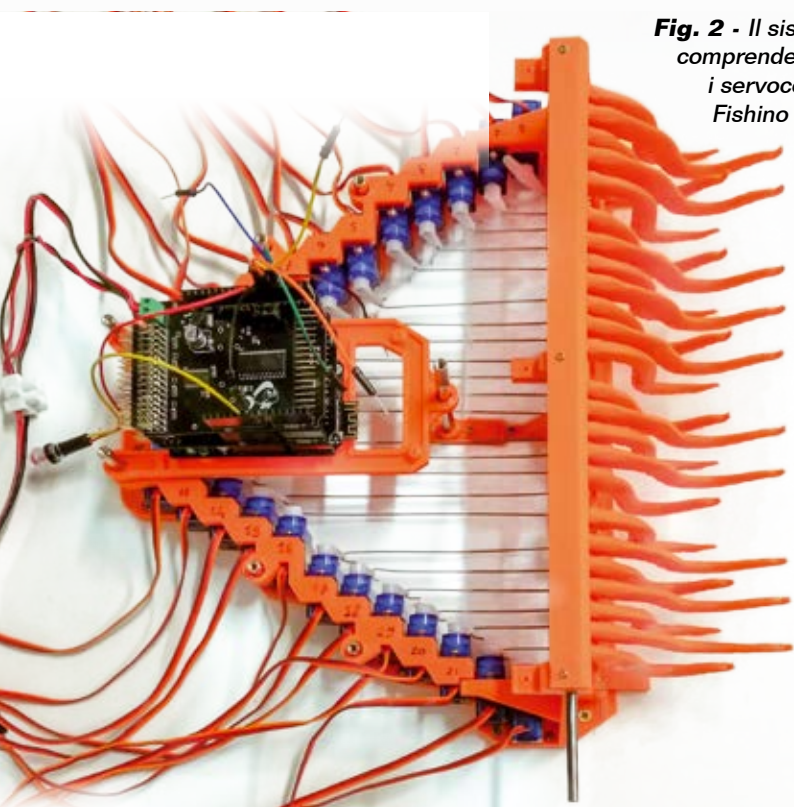
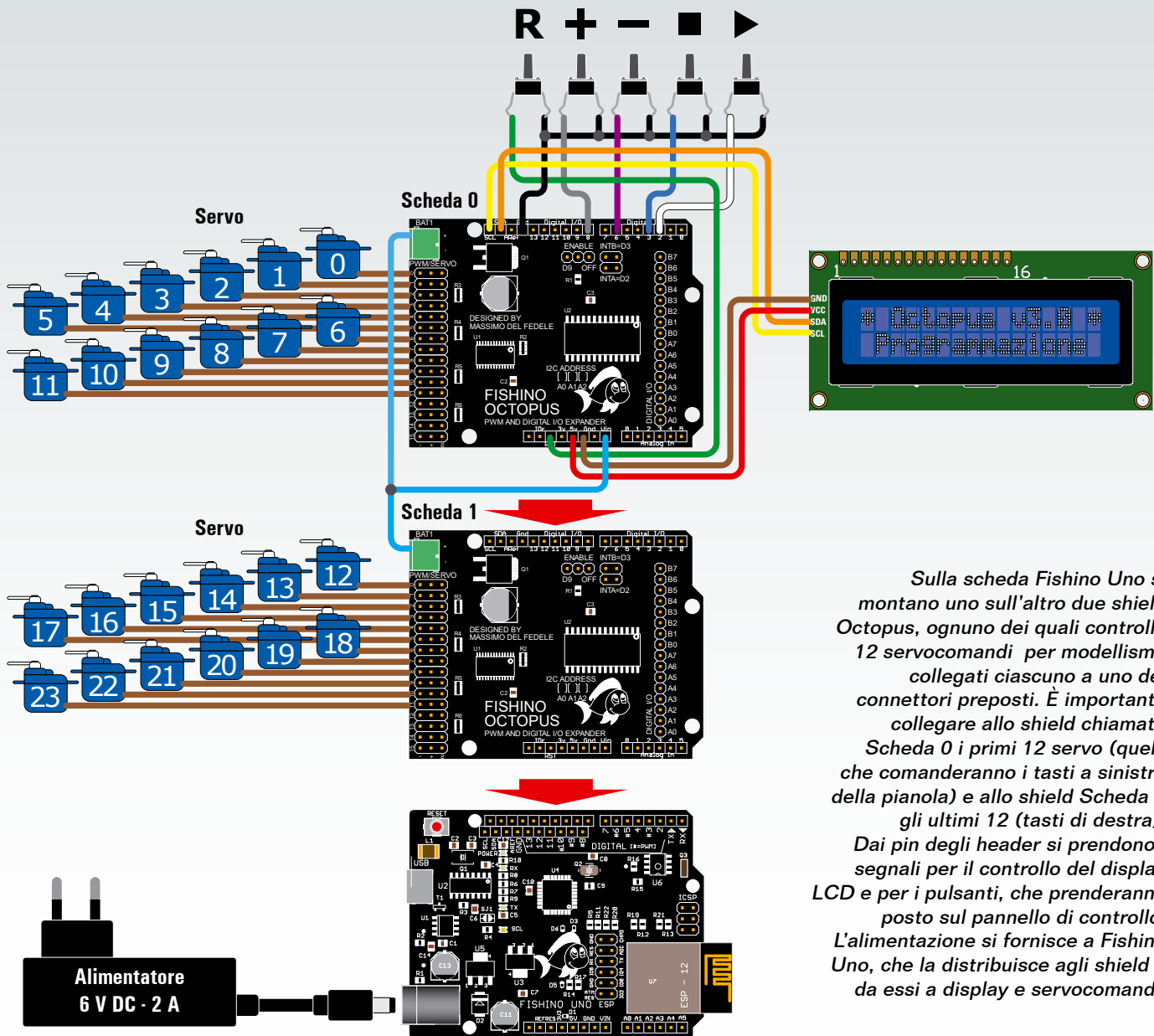


Fig. 2 - Il sistema di attuatori, comprendente la meccanica, i servocomandi e la board Fishino Uno con sopra gli shield Octopus.



Fig. 3 - Il pannello di controllo contenuto in un contenitore stampato in 3D con del PLA.



Sulla scheda Fishino Uno si montano uno sull'altro due shield Octopus, ognuno dei quali controlla 12 servocomandi per modellismo collegati ciascuno a uno dei connettori preposti. È importante collegare allo shield chiamato Scheda 0 i primi 12 servo (quelli che comanderanno i tasti a sinistra della pianola) e allo shield Scheda 1 gli ultimi 12 (tasti di destra). Dai pin degli header si prendono i segnali per il controllo del display LCD e per i pulsanti, che prenderanno posto sul pannello di controllo. L'alimentazione si fornisce a Fishino Uno, che la distribuisce agli shield e da essi a display e servocomandi.

noi utilizzata corrispondono alla 4^a e alla 5^a del pianoforte), quindi 12 tasti per ottava corrispondenti alle 7 note intere (Do, Re ecc..) più i semitoni (ad esempio Do Diesis o RE Bemolle). Per chi volesse approfondire l'argomento, i tasti di ciascuna ottava sono convenzionalmente i seguenti:

C = nota DO
C# = DO Diesis o RE Bemolle
D = nota RE

D# = RE Diesis o Mi Bemolle
E = nota MI
F = nota FA
F# = FA Diesis o Sol Bemolle
G = nota SOL
G# = SOL Diesis o LA Bemolle
A = nota LA
A# = LA Diesis o SI Bemolle
B = nota SI

Le note intere sono i tasti bianchi e quelli neri rappresentano i semitoni standard della tastiera del pianoforte.

IL SISTEMA

Bene, detto ciò possiamo entrare nel vivo del progetto, che consta di un'elettronica (formata da Fishino Uno e due shield Octopus) e una meccanica (l'insieme è visibile, smontato, nella Fig. 2); la prima è stata già meglio descritta nell'articolo che le abbiamo dedicato nel fascicolo di marzo 2016 (dove trovate l'hardware e il firmware necessari a gestirlo) quindi ci limitiamo a dire che gli shield sono due e vengono impi-



Fig. 4 - Tentacolo e relativa asta di rinvio.



Fig. 5
Dettaglio dell'azionamento dei tentacoli da parte dei servocomandi, tramite la relativa asta di rinvio piegata per entrare nel braccetto dei servo.

Per il controllo dei servo ci siamo avvalsi di uno shield, compatibile con le schede Arduino e con la nostra Fishino Uno, che, senza praticamente impegnare risorse hardware, permette di avere a disposizione ben 16 uscite in PWM e 16 ingressi/uscite digitali aggiuntivi. Non solo, le schede sono impilabili fino ad un massimo di 8, consentendo di gestire con Arduino e Fishino fino a 128 I/O digitali e 128 uscite PWM aggiuntive; il tutto reso completamente trasparente all'utilizzatore tramite una libreria realizzata ad-hoc. La libreria software, denominata Octopus, è dotata di alcune particolarità che ne rendono semplicissimo l'utilizzo. Lo shield nasce perché le schede

lati uno sull'altro su Fishino Uno. Gli shield sono gestiti attraverso l'I²C-Bus di quest'ultimo. Sebbene gli shield Octopus dispongano di 16 uscite per controllo di servo, nella nostra applicazione ognuna ne pilota 12; le schede prelevano l'alimentazione per i servocomandi dal pin Vin di Fishino (qui arrivano circa 5,25V) e quella per il proprio funzionamento dal pin 5V. L'intero sistema viene alimentato da un alimentatore switching da 6V -

2A che fornisce tutta la corrente necessaria alla Fishino Uno. L'interfaccia utente è rappresentata da un display LCD 2 righe e 16 caratteri alfanumerici (cod. LCD1602ABLU Futura Elettronica) a cui è stata applicata un'interfaccia I²C (cod. LCDI2CINTERFACE Futura Elettronica) e da un tastierino a 5 pulsanti che permettono di selezionare il tono che si desidera riprodurre ed anche di effettuare la taratura dei servo ed eseguire un test di verifica della pressione dei tasti della pianola. Agli header sono collegati i cinque pulsanti, come segue:

Mem/PLAY -> pin 2
Più -> pin 8
Meno -> pin 6
Esci/STOP -> pin 3

R -> pin di reset/riavvio di Fishino

Invece il display è collegato così:
SDA display -> SDA
SCL display -> SCL
Vcc display -> 5V
GND display -> GND.

I pin sono, anche in questo caso, quelli degli header. L'utilizzo di un display con interfaccia I²C permette di limitare il numero minimo di pin della board Fishino Uno utilizzati.

Nella Fig. 3 potete vedere come abbiamo realizzato il pannello di controllo del nostro sistema, che poi è stato collocato nella parte posteriore dell'insieme, appena sotto all'elettronica di controllo formata da Fishino Uno e dagli shield Octopus.

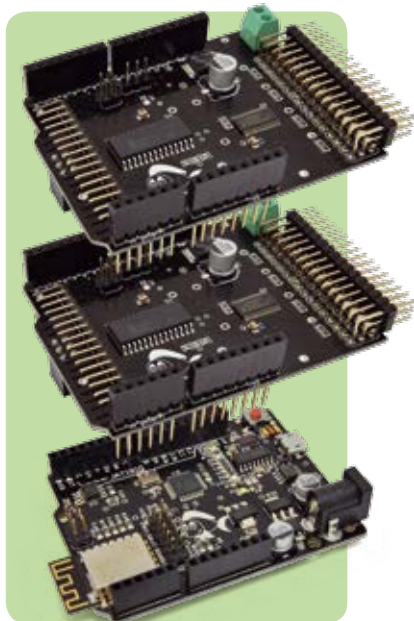


Fig. 6 - Sistema di attuazione dei tentacoli.



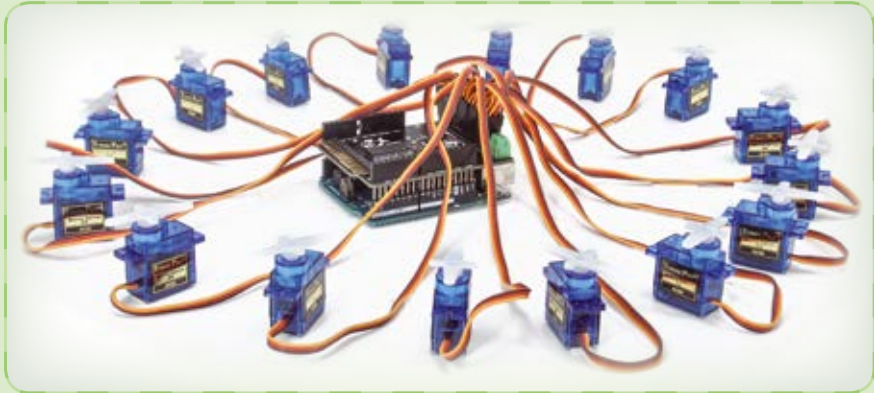
Fig. 7 - Tentacoli e aste di rinvio visti dal retro.

Octopus: lo shield tentacolare

Arduino e compatibili hanno una ridotta quantità di uscite disponibili, specie di I/O cui si può assegnare il PWM. Per esempio, una Arduino/Fishino UNO dispone di sole sei uscite PWM e, a meno di non generare i relativi segnali via software (con notevole impegno del processore), permette il pilotaggio di soli sei servocomandi. Anche gli I/O digitali sono limitati: per Arduino abbiamo un totale di 13 I/O digitali e 6 ingressi analogici, utilizzabili anch'essi in digitale. Tale dotazione sembrerebbe abbondante, se non fosse che molti I/O vengono utilizzati per le periferiche a bordo o dagli shield di espansione. Octopus estende gli I/O sfruttando la sola connessione I²C-Bus; come tutte le

periferiche del genere va indirizzata per assicurare l'univocità dei comandi diretti a ciascuno shield ed allo scopo prevede

tre jumper da impostare diversamente da uno shield all'altro. I jumper consentono 8 combinazioni logiche.



PARTE MECCANICA

Dopo aver analizzato l'elettronica del progetto, occupiamoci della meccanica, che è ugualmente importante perché è la sezione attuatrice che interviene fisicamente sui tasti della pianola. Tutta la struttura necessaria a sostenere i 24 tentacoli e a ospitare gli altrettanti servomotori (provvisi ciascuno di un braccetto avvitato all'alberino con le viti in dotazione), nonché le schede elettroniche (il telaio, quindi...) è stata progettata con Google SketchUp e realizzata mediante stampa 3D con del PLA; il telaio vero e proprio si stampa utilizzando il file STL da noi fornito insieme agli altri file del progetto (sketch ecc.) nel nostro sito Internet www.elettronica.in.it. Al telaio devono poi essere applicati i particolari metallici richiesti per l'assemblaggio delle parti, come il fulcro dei tentacoli; più esattamente, il telaio consta di più parti (lati, elemento di congiunzione posteriore, traversa anteriore...) da unire mediante viti. Ciascun servocomando aziona un tentacolo (anche questo realizzato per stampa 3D il cui file STL è reperibile sul nostro sito insieme agli altri file del progetto)

mediante un'astina di rinvio (opportunamente sagomata) ottenuta stirando un filo in acciaio inox da 0,9 mm di diametro (Fig. 4). Il filo va poi ripiegato per vincolarlo sia dalla parte dell'albero del servocomando, sia dal lato del foro previsto nel tentacolo corrispondente (Fig. 5). L'insieme dei tentacoli è libero di oscillare per assecondare il movimento imposto dai servocomandi; allo scopo ogni tentacolo, disposto affiancato agli altri, è

imperniato su un'asta in metallo da 4 mm di diametro. La lunghezza delle aste di rinvio non è uguale ma va scelta di volta in volta sulla base della distanza tra il braccetto del servocomando e la base della leva di attuazione del tentacolo corrispondente, come si vede nella Fig. 6, che mostra l'insieme della meccanica di un lato (manca la parte destra dei servocomandi e il relativo sostegno. La Fig. 7 dettaglia, invece, l'aggancio delle



Fig. 8 - La meccanica a ridosso della tastiera della pianola.



Fig. 9 - Entrata in programmazione.



Fig. 10 - Programmazione dei servo.



Fig. 11 - Fine programmazione.



Fig. 12 - Esecuzione del test iniziale.



Fig. 13 - Schermata di scelta traccia.

aste alla base dei tentacoli, Le fotografie fatte al prototipo chiariscono meglio d'ogni parola come si assembla la struttura meccanica, la quale, una volta terminata, va montata a ridosso della pianola, frontalmente, in modo che ogni tentacolo prema uno dei tasti.

Siccome è probabile che la pianola si sposti durante il funzionamento a causa della pressione esercitata dai tentacoli, è opportuno vincolare il telaio alla stessa: ad esempio con delle strisce metalliche ripiegate a squadra all'altezza del fondo dello strumento musicale, oppure appoggiando e avvitando la meccanica a una piastra di base in legno o plastica (forex, per esempio...) e fissando alla stessa, una volta determinata la posizione ottimale, la pianola mediante squadrette in plastica opportunamente stampate in 3D e avvitate alla piastra di base). La Fig. 8 propone un possibile fissaggio dell'insieme.

FUNZIONI DEI TASTI

Bene, descritta anche la parte meccanica possiamo vedere come si utilizza e quindi in che modo possiamo far suonare il nostro speciale strumento. Descriviamo di seguito il funzionamento dei pulsanti.

- **Mem/PLAY:** permette di avviare l'esecuzione del tono occupante la posizione visualizzata sul display. Se tenuto premuto all'accensione permette di entrare in "Modalità impostazione posizione servo".
- **Più:** permette di passare al tono successivo. In "Modalità impostazione posizione servo" permette di modificare la posizione del tentacolo al fine di premere + o - il tasto della pianola. Se viene tenuto premuto all'accensione permette di entrare in modalità TEST.

**Guarda Octopus
in azione su YouTube!**



Puoi vedere Octopus, realizzato con la stampante 3Drag, mentre suona il piano in un video demo



che abbiamo pubblicato alla pagina <https://youtu.be/ayyAver7gMg>. Dal nostro canale YouTube www.youtube.com/user/ElettronicaIN potrai anche vedere i video di moltissimi altri nostri progetti proposti in passato.



- **Meno:** permette di passare al tono precedente. In "Modalità impostazione posizione servo" permette di modificare la posizione del tentacolo al fine di premere + o - il tasto della pianola. Se tenuto premuto all'accensione permette di entrare in modalità TEST.
- **Esci/STOP:** permette di interrompere l'esecuzione di un tono. Se viene premuto quando ci si trova in "Modalità impostazione posizione servo" permette di uscire da tale modalità e di tornare alla schermata principale.
- **R:** permette di resettare e/o riavviare il sistema.

Reset della posizione di tutti i servo

Prima di applicare la leva di

comando, facente capo al tentacolo, sul perno di ciascun servo, occorre posizionare tutti i servo a metà corsa (posizione centrale, -90°); tale operazione è necessaria per essere certi che una volta portati a zero i servocomandi, i braccetti siano tutti nella posizione che lascia a riposo i tentacoli. Per effettuare il reset della posizione di tutti i servocomandi bisogna premere e tenere premuto il tasto **Esci/STOP** quindi fornire alimentazione; il display indicherà la scritta "Reset

in corso" dopo-diché tutti i perni dei servo si collocano in posizione centrale e sul display apparirà la scritta "Reset eseguito". Ora rilasciate il tasto. Fatto ciò applicate su ciascun perno la propria levetta in modo che il relativo tentacolo sfiori (senza premere) il corrispondente tasto della pianola.

MODALITÀ PROGRAMMAZIONE

Vediamo adesso come si programma la posizione di pressione del tasto corrispondente al singolo tentacolo. Ad ogni servo deve essere assegnata una posizione che permette la pressione del tasto assegnato. Per effettuare l'impostazione del caso bisogna seguire la procedura seguente.

- 1) Accendere la pianola, premere e tenere premuto il tasto **Mem/PLAY** quindi fornire alimentazione; il display visualizza la scritta "Programmazione" (Fig. 9).
- 2) Rilasciare il tasto. Il servo gestito sarà il primo a sinistra (la prima nota della prima ottava); il display visualizzerà la scritta "Programmazione Servo no.1" (Fig. 10).
- 3) Premere il tasto + o - in modo che il relativo tentacolo prema sul corrispondente tasto quanto basta per emettere la



nota. Raggiunto l'obiettivo, premere nuovamente il tasto **Mem/PLAY** per passare alla programmazione del servo successivo.

- 4) Giunti all'ultimo servo premere il tasto **Mem/PLAY** per concludere la programmazione (sul display apparirà la scritta "Programmazione Completata" (Fig. 11).

TEST DEL SISTEMA

Ora verificate se tutti i tentacoli premono correttamente il tasto assegnato: per fare ciò bisogna:

- accendere la pianola, premere e tenere premuto il tasto Più (o Meno) quindi fornire alimentazione; il display visualizza la scritta "Test" (Fig. 12).
- Rilasciare il tasto.

Ora i tentacoli verranno azionati in successione, da sinistra verso destra, così da verificare la corretta pressione di tutti i tasti ad essi assegnati. Al termine del test il sistema tornerà alla schermata principale "Scelta traccia".

SCELTA TRACCIA / PLAY / STOP

Quando il sistema è in standby, sul display appare la scritta "Scelta traccia" seguita dal numero della traccia (della suoneria) attuale (Fig. 13). Per avviare l'esecuzione della traccia indicata premete il tasto PLAY (il display mostra ad esempio "Play Traccia no.1"); per arrestare l'esecuzione premete STOP.

Per scegliere un'altra traccia premete il tasto Più o il Meno (il comando non ha effetto se il pulsante viene premuto durante l'esecuzione della traccia); prima ricordate di premere STOP.

IL FIRMWARE

Concludiamo questo articolo con un accenno al firmware, nel quale sono contenuti sia i Ringtone da

riprodurre, sia le istruzioni per suonarli. Partiamo proprio dagli RTTTL, dicendo che nel sistema è previsto ne vengano suonati 30: ciò viene definito nell'istruzione `int Tracce_Max = 29;` nell'inizializzazione dello sketch. Notate che 30 corrisponde al valore 29 perché si considera di partire da 0. Modificando tale valore si può modificare il numero massimo di suonerie richiamabile.

Ciascun motivo viene costruito secondo lo standard RTTTL, che prevede la definizione di ogni nota con la relativa durata, separata da una virgola dalla seguente e dalla precedente. Il formato tipico di una stringa è:

`const char PROGMEM Music-SongPlay1[] = "wewish:d=4,o=5,b=140:d,g,8g,8a,8g,8f#,e,c,e,a,8a,8b,8a,8g,f#,d,f#,b,8b,8c6,8b,8a,g,e,8d,8d,e,a,f#,2g.";` che riguarda l'esecuzione del Ringtone *wewish*,

composto dalle note definite dalla lettera con, seguita da uguale, la durata, quindi `d=4`, `o=5`, `b=140` ecc. (`d` è la durata, o l'ottava e `b` le battute). Notate che tali istruzioni possono essere composte scrivendole nell'IDE Arduino, ovvero copiate direttamente dai siti sul web che riportano i Ringtone, dato che li espongono con detta sintassi.

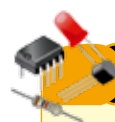
In genere tali siti forniscono dei file che si possono aprire con un editor di testo (sono dei `.txt`); per scrivere la musica vi basta copiare la stringa di testo interessata e incollarla nell'editore dell'IDE di Arduino.

Nello sketch, ciascuna nota viene definita dall'istruzione del tipo: `#define NOTE_C4 262`

che specifica la posizione del servo da far attivare per premere il tasto corrispondente sulla tastiera della pianola.

Quanto all'indirizzamento degli shield, la libreria *Octopus.h* è

preimpostata in modo che le uscite 0÷15 competano allo shield indirizzato come 0, le 16÷32 a quello indirizzato come 1, ecc. Quindi prima di applicare gli shield ad Arduino o Fishino dovete impostare i jumper dell'indirizzo del primo shield come 000 e quelli del secondo come 001, altrimenti non verrà rispettata la corrispondenza tra note e servo-comandi stabilita dal firmware. Lo sketch per Arduino/Fishino è piuttosto voluminoso e, per ragioni di spazio, non possiamo riportarlo in queste pagine; lo potete comunque trovare e scaricare nella sezione Download del nostro sito www.elettronica.in, insieme agli altri file del progetto. Bene, con questo diciamo che abbiamo concluso; lasciamo a voi e alla vostra fantasia eventuali sviluppi e personalizzazioni. ■



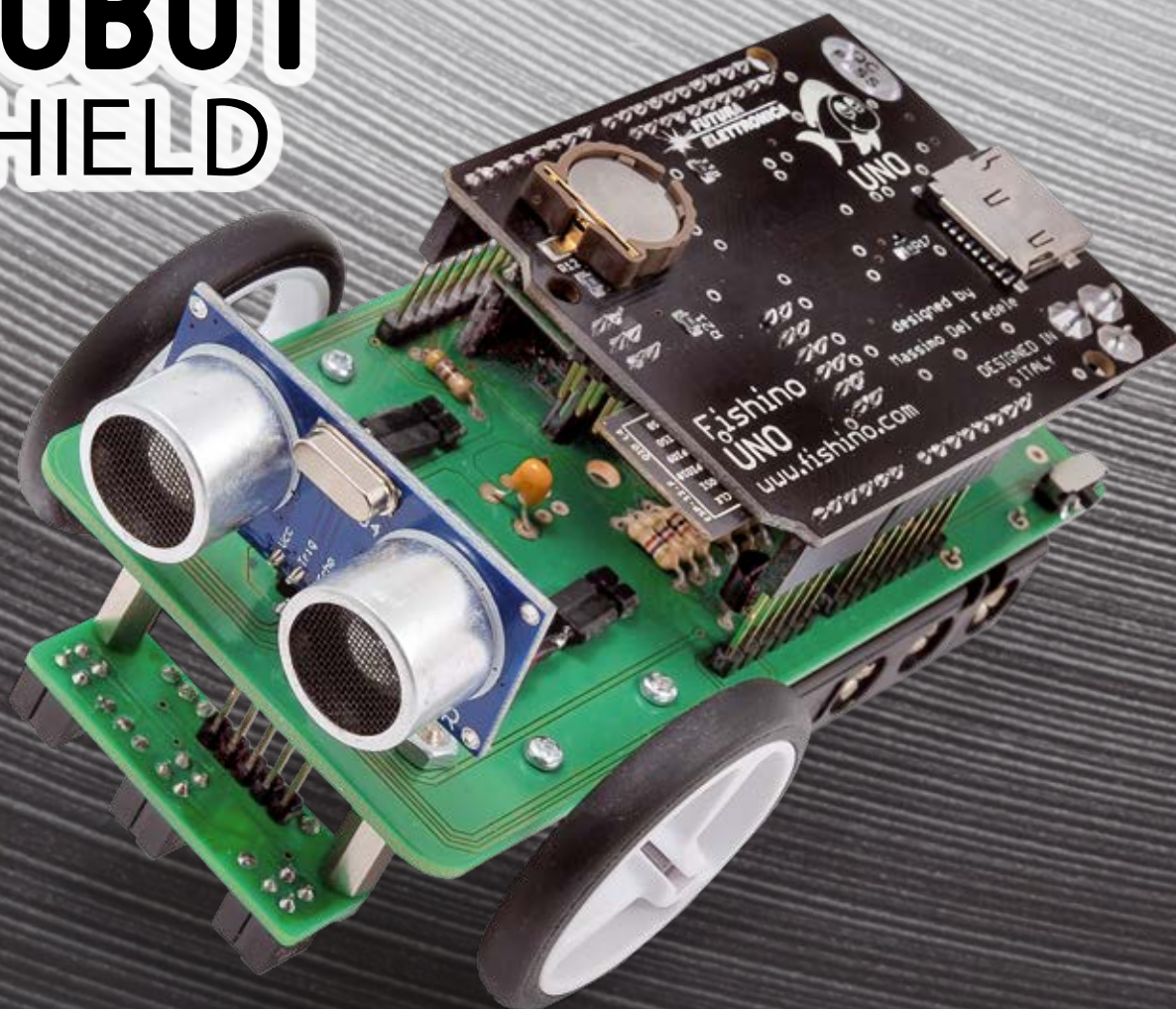
per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità, mentre il materiale elencato di seguito è disponibile presso Futura Elettronica. La board Fishino UNO (cod. FISHINOUNO) è in vendita a Euro 36,00, l'OCTOPUS shield 16 I/O per Fishino/Arduino (cod. OCTOPUS) costa Euro 24,00 (cadauno). Il SUB MICRO SERVO (cod. SERVO206) è disponibile a Euro 7,50 (cadauno), il display LCD 16x2 con retroilluminazione blu (cod. LCD1602ABLU) è in vendita a Euro 8,70 e l'interfaccia I²C per display LCD 16x2 (cod. LCDI2CINTERFACE) costa Euro 6,00. Tutti i prezzi si intendono IVA compresa.

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

EASY ROBOT SHIELD

Piattaforma per sviluppare un piccolo robot su ruote controllabile tramite uno smartphone collegato in WiFi.

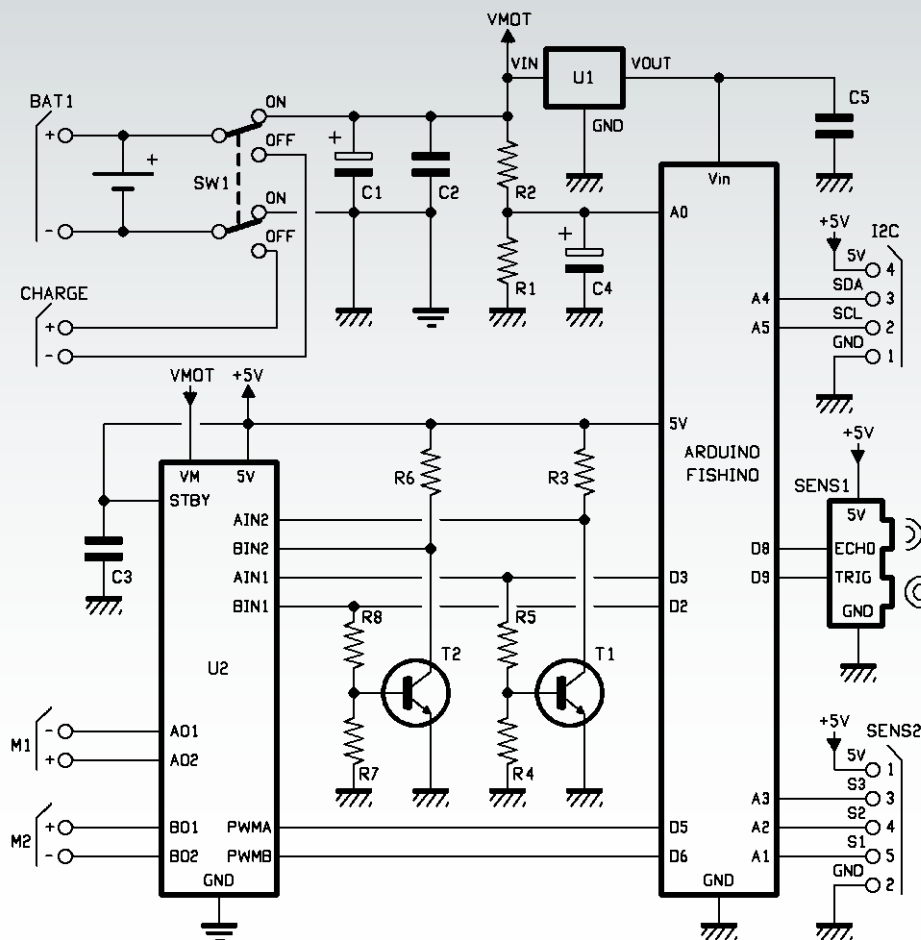


dell'Ing. MIRCO SEGATELLO

Non è la prima volta che presentiamo un robot su ruote basato su Arduino, ma questa volta, oltre a proporvi una nuova variante di facile utilizzo (non a caso utilizziamo il termine Easy Robot...) più che un robot vogliamo introdurvi uno shield molto particolare, perché, come vedremo nel corso dell'articolo, costituirà sia la parte elettrica che meccanica: uno shield in verità anomalo, perché ha una particolare sagoma e dimensioni adatte a fare da supporto elettrico e meccanico all'intero robot, compresa una scheda Arduino, che sarà il

cervello. Questo modo di procedere consente di abbattere i costi di produzione e semplificare notevolmente la sua realizzazione, permettendo anche ai meno esperti di cimentarsi con la robotica. Il valore aggiunto in questo progetto è la piena compatibilità con le schede Arduino, ma anche con schede di terze parti come le nostre Fishino. Su quest'ultima si concentrerà il nostro lavoro, giacché proporremo un'applicazione pratica proprio con questo tipo di scheda, che, lo ricordiamo, oltre a svolgere le normali funzioni di una sche-

[schema elettrico **SHIELD**]



da Arduino è equipaggiata con un modulo WiFi (più un RTC e SD Card) quindi può interagire direttamente in modalità wireless con altri dispositivi, come ad esempio uno smartphone o un tablet.

SCHEMA ELETTRICO

Analizziamo ora lo schema elettrico, per capire come funzionano i diversi blocchi funzionali e i moduli componenti utilizzati. Premettiamo che per semplificare lo schema ci siamo affidati, per alcune sezioni, a moduli già pronti reperibili in commercio come il radar a ultrasuoni. L'insieme consta di due schede, ossia lo shield che fa da supporto a tutti i componenti, i moduli e la scheda Fishino, più il circuito del

sensore infrarosso a riflessione a
 tre stadi che fornisce al robot la
 capacità di inseguimento delle
 linee.

L'alimentazione dell'intero robot è ottenuta grazie all'utilizzo di un pacco di quattro pile tipo ministilo da 1,5 volt, in grado di fornire una tensione compresa tra 4,5 e 6,5 V, a seconda del livello di carica. Nulla vieta (anzi è caldamente consigliabile) di utilizzare delle batterie ricaricabili NiMH che, pur fornendo una tensione leggermente più bassa, consentono di essere riutilizzate previa la loro ricarica.

L'alimentazione a batterie è la scelta elettiva per un robot che debba muoversi in libertà. La tensione fornita dalle quattro batterie è ottimale per alimenta-

re, senza alcuno stress, i piccoli motoriduttori ma risulta insufficiente per alimentare correttamente una scheda Arduino o Fishino, pertanto utilizziamo un boost converter per innalzare la tensione ad un valore di circa 7V, indipendentemente dal livello di carica delle batterie. Si tratta di un piccolissimo convertitore DC/DC di tipo step-up, in grado di elevare una tensione continua in ingresso, compresa tra 1,8 e 8V, in una tensione regolabile di 9V, impostando una tensione in uscita del valore di 9V, saremo sicuri di alimentare correttamente la scheda Arduino, rispettando il vincolo imposto dalle caratteristiche del modulo.

Per la lettura del livello di tensione delle batterie è stato previsto un partitore di tensione, in grado di fornire a Fishino una tensione dimezzata rispetto a quella

Tabella 1 - Disposizione pin dello shield.

pin	nome	funzione
D0	RX	RX Arduino
D1	TX	TX Arduino
D2	DIRB	Direzione motore B
D3	DIRA	Direzione motore A
D4	---	usato da Fishino
D5	PWMA	PWM motore A
D6	PWMB	PWM motore B
D7	---	usato da Fishino
D8	ECHO_PIN	Sens. ultrasuoni echo pin
D9	TRIG_PIN	Sens. ultrasuono Trigger pin
D10	---	usato da Fishino
D11	---	usato da Fishino
D12	---	usato da Fishino
D13	---	usato da Fishino
A0	VBAT	Misura tensione batteria
A1	S1	Sensore di linea 1
A2	S2	Sensore di linea 2
A3	S3	Sensore di linea 3
A4	---	A disposizione per ampliamenti
A5	---	A disposizione per ampliamenti

fornita dalle batterie; a livello software convertirò il livello di tensione in un valore numerico in percentuale, utile a rilevare lo stato di carica della batteria. Il driver impiegato per controllare i motori è fornito dalla Futura Elettronica e identificato dal codice 7300-TB6612FNG; tale driver è in grado di gestire sino a due motori contemporaneamente, con correnti sino a 1A e tensione di alimentazione da 4,5 a 13,5 Vcc. Il driver e quindi i motori, sono alimentati direttamente dalla batterie, così da svincolarli dalla limitazione della massima corrente di uscita fornita dal convertitore di tensione, che invece alimenterà solo la scheda di controllo.

Per gestire il modulo driver servono sei linee di controllo, che ridurremo a quattro grazie all'utilizzo di due BJT utilizzati per invertire il segnale su due linee; in questo modo, per ciascun motore disporremo della linea per il PWM, che controllerà la potenza sul motore, ed una linea digitale per impostare il verso di rotazione.

I SENSORI

Per quanto riguarda la sensoristica abbiamo previsto un sensore a ultrasuoni per identificare la distanza alla quale si trovano eventuali ostacoli di fronte al

Tabella 2 - Elenco materiale.

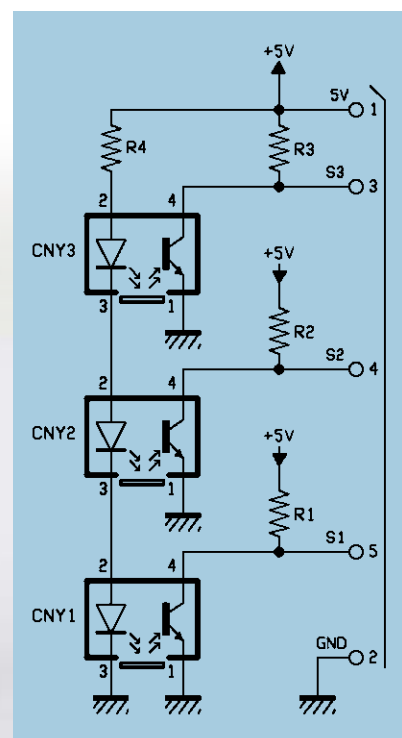
Componente	Codice
Micro motoriduttore in metallo 85 RPM	1686-MMG150
Coppia di supporti per motoriduttori MMG298	7300-MMGSUPP
Coppia di ruote pololu 40x7mm per motoriduttore mmg	8218-RP407
Ball caster con sfera in metallo da 1,2 cm	7300-PBCASTER12
Convertitore DC/DC uscita regolabile da 2,5 a 9,5V	7300-ABR2595
Sensore ultrasuoni 2=450 cm	2846-MISDIST04
Portabatteria 4 ministilo (AAA) con terminali a saldare (L x L x A) 52 x 49 x 13 mm	BH441P

robot. Si tratta di un economico modulo SONAR (radar a ultrasuoni codice 2846-MISDIST04) che è in grado di eseguire misure di distanza comprese tra 2 e 450 cm; per il suo controllo sono necessarie due linee digitali, una per il segnale di trigger ed una per il segnale echo, la cui durata è proporzionale alla distanza dell'oggetto; a tale riguardo va detto che l'uscita (echo) fornisce un impulso quando al sensore viene richiesto attraverso un impulso logico a livello alto della durata di 10 μ s sul trigger: il sensore trasmette otto impulsi a 40 kHz, che riflessi tornano alla capsula ricevente e sul pin di uscita echo viene fornito un impulso la cui durata rappresenta il ritardo e, di riflesso, la distanza. Ad esempio un impulso da 150 μ s corrisponde a 2 cm e uno da 25 ms a 4,5 m. Se non rileva alcun ostacolo entro i 4,5 m, il SONAR produce un impulso di durata di

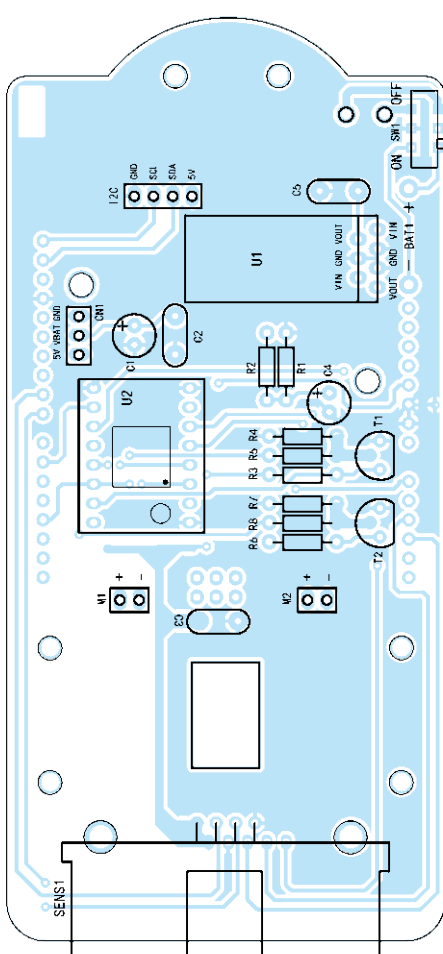
circa 38 ms. In generale la distanza è pari alla durata dell'impulso in μ s diviso 58.

Il rilevamento è valido frontalmente o con un'angolazione di $\pm 15^\circ$ rispetto alla perpendicolare al sensore.

Oltre al SONAR abbiamo previsto tre sensori di linea ad infrarossi basati sull'affermato sensore a riflessione CNY-70 (codice 7300-CNY70) che, nella configurazione da noi usata, forniscono in uscita una tensione inversamente proporzionale al livello di riflessione del suolo in cui si trova ad operare il robot: sfondi chiari forniranno tensioni basse e viceversa sfondi scuri forniranno valori alti. Questa è



[schema elettrico **SENSORE IR**]



Elenco Componenti:

R1, R2, R4, R7: 10 kohm
 R3, R5, R6: 4,7 kohm
 R8: 4,7 kohm
 C1: 47 μ F 63 VL elettrolitico
 C2, C3: 100 nF ceramico
 C4: 1 μ F 63 VL elettrolitico
 C5: 100 nF ceramico
 T1, T2: BC547
 U1: Modulo Boost DC Converter
 U2: Driver motori (TB6612FNG)
 SW1: Deviatore slitta 2 vie
 SENS1: Sensore ultrasuoni (MISDIST04)
 SENS2: Modulo CNY70 (FT1289)
 M1, M2: Micro Motoriduttore 6V (MMG150)

Varie:

- Strip maschio 2 vie 90° (2 pz.)
- Strip maschio 4 vie (2 pz.)
- Strip femmina 2 vie (2 pz.)
- Strip femmina 4 vie 90° (3 pz.)
- Strip femmina 4 vie (2 pz.)
- Strip maschio 6 vie 18 mm
- Strip maschio 8 vie 18 mm (2 pz.)
- Strip maschio 10 vie 18 mm (2 pz.)
- Porta Batteria 4xAAA da CS
- Ball Caster (PBCASTER12)
- Ruota 40x7 mm (2 pz.)
- Distanziale M/F 15mm (2 pz.)
- Vite 10 mm 3 MA (2 pz.)
- Dado 3 MA (4 pz.)
- Circuito stampato S1288 (58 x 123 mm)

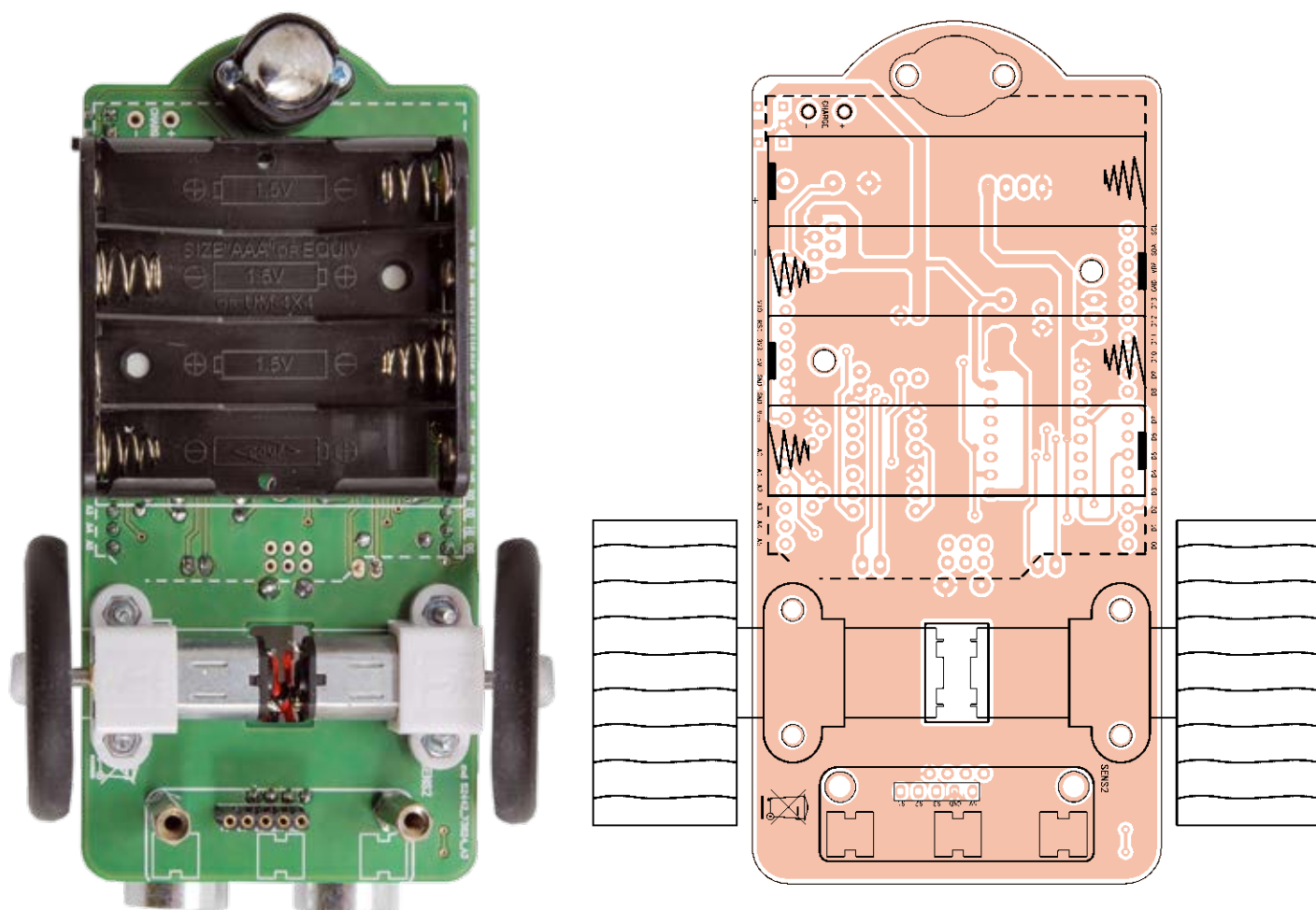
la classica funzionalità richiesta per realizzare robot inseguitori di linea o anche semplicemente per identificare il tipo di pavimento in cui si trova a operare il robot. Lo shield è stato pensato per essere utilizzato in abbinamento ad una scheda Fishino, pertanto i pin a disposizione per il nostro shield sono solo quelli non utilizzati dalla scheda di controllo. Alla fine solo i pin A4 e A5 non vengono usati e lasciati disponibili in un opportuno connettore, a disposizione per essere usati come generici pin digitali, come ingressi analogici oppure per comunicazioni I²C. Per completezza nella **Tabella 1**

riportiamo l'elenco dei pin utilizzati e la loro funzione.

REALIZZAZIONE PRATICA

La realizzazione del robot non comporta difficoltà, perché i componenti da saldare sono pochi, mentre i componenti SMD sono tutti forniti sotto forma di breakout board per renderne facile l'utilizzo. Una volta ottenuti i PCB (quello grande è lo shield e il piccolo il triplo sensore di linea) iniziate la saldatura dai componenti a più basso profilo proseguendo mano a mano con quelli più alti, aiutandovi con le immagini riportate in queste pagine e i piani di montaggio. Per

ultimi fissate il convertitore di tensione e il driver per i motori, direttamente oppure dotandoli di connettori strip, a seconda dell'altezza finale che vorrete ottenere; ricordate che sopra lo shield sarà posta la scheda Arduino rovesciata, ragion per cui è probabile che sia necessario montare gli elettrolitici sdraiati. Provvedete a tagliare con cura i reofori dei componenti sulla parte inferiore che sarà utilizzata per fissare il porta batteria. Terminata la saldatura di tutti i componenti, potete passare all'installazione dei componenti meccanici, infatti questo shield svolge anche la funzione di



chassis rendendo la realizzazione molto più semplice. Saldate due piccoli filetti sui contatti dei motori, meglio se terminati con un piccolo connettore ricavato da uno strip femmina; durante le prime prove sarà semplice invertire la polarità dei motori se non dovessero girare nel modo opportuno. Fissate i motori con il loro supporto negli appositi alloggiamenti avendo cura di allineare i mozzì delle ruote allo stampato. Fissate il porta batteria con due piccole viti e prevedete il collegamento elettrico, filo positivo e negativo di alimentazione. Il piccolo stampato sarà elettricamente e meccanicamente connesso

allo shield tramite una coppia di connettori strip maschio sullo stampato dei sensori e femmina sullo shield, per quest'ultimi esistono due altezze commerciali, 12 e 18mm; se usate quelli da 18 mm avrete un maggior spazio per i componenti saldati sullo shield. Completate il secondo stampato (quello più piccolo) con i tre sensori CNY70 solo dopo aver saldato le resistenze, fate molta attenzione all'orientamento dei componenti. I motori vanno fissati allo shield tramite due piccoli supporti (codice 7300-MMGSUPP). Installate le ruote sul mozzo dei motori e fissate il ball caster con

gli adeguati spessori in modo da garantire che il robot rimanga orizzontale, a questo punto avrete la precisa altezza da terra dello shield e quindi potete provvedere all'installazione dello stampato dei sensori di linea. La distanza



Fig. 1 - Creazione dell'interfaccia tramite l'app RoboRemo.

Listato 1

```
// Ricezione pacchetto dati dallo smartphone ed estrapolazione valori numerici
if (packetSize)
{
    int len = Udp.read(cmd, 255);
    if (len > 0) cmd[len] = 0;

    isConnected=1;

    if(cmd[0]=='s' && cmd[1]=='p' && cmd[2]=='e' && cmd[3]=='e' && cmd[4]=='d' && cmd[5]==' ') {
        speed = 0;
        if (cmd[6]=='-') {
            for(int i=7; cmd[i]!='\n'; i++) {
                speed = speed*10 + cmd[i]-'0';
            }
            speed = -speed;
        }
        else {
            for(int i=6; cmd[i]!='\n'; i++) {
                speed = speed*10 + cmd[i]-'0';
            }
        }
        //Serial.println(speed);
    }

    // Parse incoming steer data
    if(cmd[0]=='s' && cmd[1]=='t' && cmd[2]=='e' && cmd[3]=='e' && cmd[4]=='r' && cmd[5]==' ') {
        steer = 0;
        if (cmd[6]=='-') {
            for(int i=7; cmd[i]!='\n'; i++) {
                steer = steer*10 + cmd[i]-'0';
            }
            steer = -steer;
        }
        else {
            for(int i=6; cmd[i]!='\n'; i++) {
                steer = steer*10 + cmd[i]-'0';
            }
        }
    }

    // do command
    turn(speed, steer);
}
```

za finale dovrà essere di 2 o 3 millimetri dal suolo, a tale scopo serviranno due distanziatori da 15mm in metallo o plastica secondo disponibilità.

Il sensore ad ultrasuoni può essere saldato direttamente sullo stampato garantendo un profilo finale molto basso, oppure potete utilizzare un connettore strip femmina, così da garantire il recupero del componente all'occorrenza.

Il robot si presta a diversi utilizzi sia didattici che hobbistici, dal classico inseguire linea sino ad esercitazioni di coding, la compatibilità con la scheda Fishino ci ha però permesso di sperimentare nuove applicazioni, proprio grazie alla connettività

a disposizione. Il modulo WiFi interno a Fishino permette di connettersi ad una rete WiFi esistente e scambiare dati con essa,

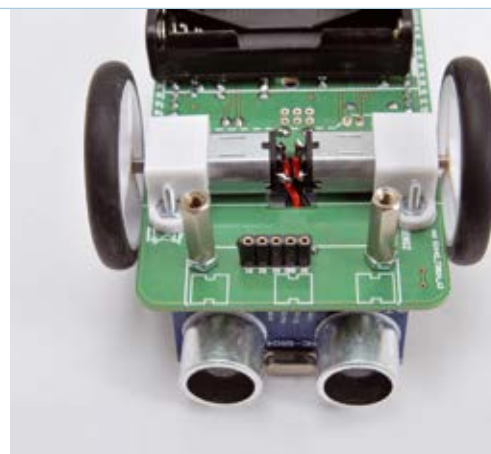
[piano di montaggio **SENSORE IR**]

Elenco Componenti:

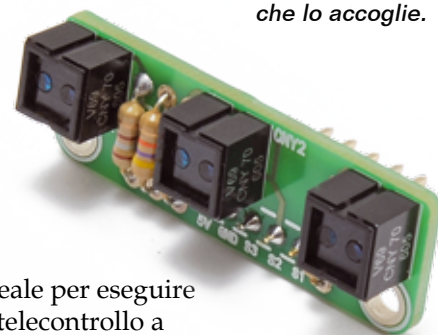
R1 ÷ R3: 22 kohm
 R4: 180 ohm
 CNY1: CNY70
 CNY2: CNY70
 CNY3: CNY70

Varie:

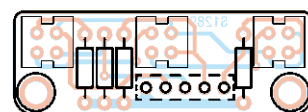
- Strip maschio 5 vie 18mm
- Circuito stampato S1289 (39 x 14 mm)



Il circuito stampato del triplo sensore a riflessione va montato dotandolo di pin-strip e fissato tramite colonnine esagonali al lato superiore dello shield, dove trova posto lo strip femmina che lo accoglie.



l'ideale per eseguire un telecomando a distanza di ultima generazione. Abbiamo già apprezzato in passato l'utilizzo dell'app RoboRemo (descritta nell'articolo Smart Home System con Fishino pubblicato nel fascicolo n° 211) per scambiare dati con sistemi connessi in rete WiFi (e non solo) e ci è sembrato interessante riuti-



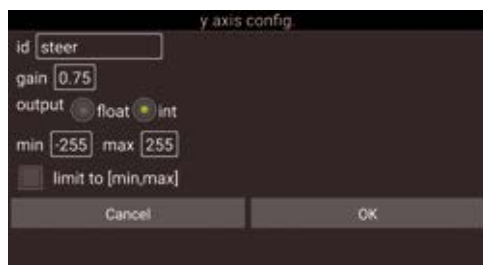


Fig. 2 - Impostazione widget accelerometer.



Fig. 3 - Impostazione widget slider.

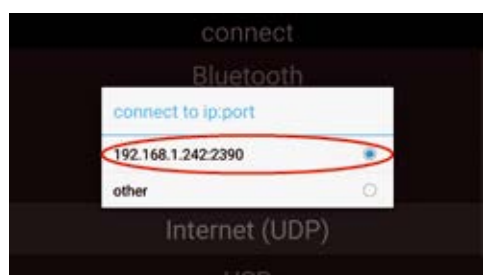


Fig. 4 - Connessione a EasyRobot.

lizzarla anche in questo progetto, vediamo come.

L'idea è di controllare a distanza il nostro robot utilizzando il nostro smartphone come telecomando costruendo un'interfaccia personalizzata e personalizzabile proprio con l'app RoboRemo. Tramite uno slider potremmo impostare la velocità di avanzamento del robot, mentre inclinando lo smartphone imposteremo la sterzata, il robot può anche girare su se stesso semplicemente impostando a zero la velocità ed agendo solo sullo sterzo. Per prima cosa programmeremo Fishino affinché possa comunicare tramite pacchetti UDP con dispositivi connessi alla stessa rete. In assenza di una struttura di rete sarà sufficiente programmare Fishino in modalità SoftAP (software enabled access point) al quale il vostro smartphone si

connetterà indipendentemente dall'esistenza oppure no di una rete WiFi.

La versione free di RoboRemo non permette di utilizzare molti widget, ma a noi basta il control-

lo della velocità e della direzione; quest'ultimo sarà implementato utilizzando l'accelerometro interno dello smartphone, cosicché sarà sufficiente tenere tra le mani lo smartphone (in posizione oriz-

Tabella 3 - Elenco delle funzioni del file *funzioni.ino*.

Funzione	Parametri	Descrizione
void turn(int power, int turn)	power da -255 a +255 turn da -255 a +255	controllo del movimento del robot (avanti e indietro) sterzando nella misura indicata da turn
void forward(int power)	power da 0 a 255	fa avanzare il robot con potenza indicata dal parametro power
void reverse(int power)	power da 0 a 255	fa indietreggiare il robot con potenza indicata dal parametro power
void stop()		ferma il robot
int distance()		visualizza la distanza misurata con il sensore ad ultrasuoni in cm
float VBat()		Visualizza la tensione della batteria in volt

Tabella 4 - Impostazioni parametri interfaccia.

Widget	Parametro	Valore
Slider	set label	SPEED
	set id	speed
	min, max	-255 e +255
	send when moved	on
	set autoreturn	on
	set autoreturn value	mid
Accelerometer	configurare asse Y	smartphone orizzontale
	set id	steer
	gain	0.75
	output	int
	min, 255	-255 e +255
	set repat periodo	200msec
Text Filed	set id	VBAT
	set text	Vbat

Listato 2

```
// Verifica dello stato della connessione ed invio del valore di carica della batteria
if (millis() > lasttime+1000)
{
    //Verifica ogni secondo se manca segnale
    if (isConnect==0)
        turn(0,0);
    isConnect=0;

    //send battery level
    char tmp[4];
    int BatLevel=100*(VBat()-VbatMIN)/(VbatMAX-VbatMIN);
    BatLevel = constrain(BatLevel, 0, 100);
    sprintf(tmp,"%d",BatLevel);
    String VbatString(tmp);
    sendData = "VBAT " + VbatString + "%\n";

    Udp.beginPacket(Udp.remoteIP(), localPort);
    Udp.print(sendData);
    Udp.endPacket();
    lasttime=millis();
}
```

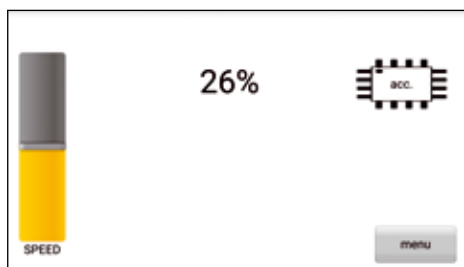


Fig. 5 - Interfaccia di comando conclusa.



Fig. 6 - Controlli del robot.



Fig. 7 - Easyrobot in funzione in una piccola pista approntata per testarne le funzionalità.

zontale) ed inclinandolo comandare la sterzata del robot.

Dallo smartphone saranno inviate essenzialmente due stringhe contenenti, rispettivamente, il valore della velocità (sia in avanti che retromarcia) che quello corrispondente alla sterzata richiesta; dal lato Fishino sarà sufficiente interpretare le due stringhe ricevute e comandare di conseguenza il robot. Non riportiamo la parte relativa alla connessione alla rete WiFi, perché già ampiamente descritta in articoli passati; riportiamo invece, nel **Listato 1**, la porzione di codice che si occupa di estrapolare i dati dalle stringhe inviate da RoboRemo. I valori di velocità e sterzata sono poi utilizzati dalla funzione *turn* per comandare correttamente i motori affinché eseguano il comando richiesto.

Pur non avendo previsto una specifica libreria, abbiamo realizzato un file chiamato *function.ino*, che contiene tutte le funzioni necessarie al funzionamento del robot, in modo che il listato principale rimanga pulito e snello e non vi dobbiate preoccupare di

scrivere complicate istruzioni. In questo file sono contenute le istruzioni a basso livello necessarie ad attivare tutte le funzionalità del robot.

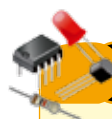
Per l'utilizzo dei sensori di linea non abbiamo previsto alcuna funzione in quanto l'istruzione *analogRead()* è sufficiente per leggere correttamente i valori, eventualmente eseguite alcune misure su diverse superfici per tarare adeguatamente la loro lettura. Una seconda parte del programma si occupa, ad intervalli di un secondo, di inviare allo smartphone lo stato della batteria e di controllare lo stato della connessione; a riguardo considerate che in mancanza di essa per più di un secondo, il robot viene fermato (vedere il **Listato 2**). Prima di caricare il firmware su Fishino dovrete configurare i parametri di accesso alla vostra rete modificando le righe `#define MY_SSID` e `#define MY_PASS` in:

```
#define MY_SSID "MY_SSID"
#define MY_PASS "MY_PASS"
```

Se volete realizzare delle sfide

con i vostri amici o semplicemente volete usare più robot contemporaneamente, è necessario che ciascuno disponga di un indirizzo IP differente; attenti a non andare in conflitto con altri dispositivi connessi alla rete. La riga per definire l'IP address è del tipo:

```
#define IPADDR 192, 168, 1, 242
```



per il MATERIALE

Tutto il materiale utilizzato in questo progetto è disponibile presso Futura Elettronica. Lo shield Easyrobot per Arduino-Fishino (cod. EASYROBOT) è in vendita a Euro 74,00, il sensore IR per robotica (cod. FT1289K) è disponibile a Euro 7,50. Il misuratore di distanza ad ultrasuoni (cod. MISOIST04) costa Euro 6,00 e la board Fishino UNO (cod. FISHINOUNO) è disponibile a Euro 36,00. Tutti i prezzi si intendono IVA compresa.

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

Altri due parametri possono essere impostati, relativamente ai livelli minimo e massimo della tensione ai capi della batteria, questi saranno usati per ricavare il livello di carica della batteria in valore percentuale; si tratta di:

```
const float VbatMIN = 4.0;
//tensione a batteria scarica pari a 0%
const float VbatMAX = 6.5;
//tensione a batteria carica pari a 100%
```

Il passo successivo consiste nell'installare sul vostro smartphone l'applicazione RoboRemo disponibile gratuitamente nel Play Store. Una volta avviata l'app entrate nella modalità *edit ui* ed inserite uno *slider*, un *accelerometer* ed un *text field*, dopodiché settate i loro parametri come indicato nella **Tabella 4** e mostrato nelle **Fig. 2**, **Fig. 3** e **Fig. 4**. Impostate quindi la proprietà *autorotate* su *lock* per evitare che l'interfaccia ruoti quando ruotate lo smartphone per far sterzare il robot.

Per far funzionare il tutto è necessario accendere il robot ed attendere che si connetta alla

rete: di solito sono sufficienti circa 5 secondi, ma per sicurezza potete lasciare connessa la scheda Fishino al Personal Computer ed attendere i messaggi di debug per verificare se le operazioni vanno a buon fine.

Una volta programmata l'app RoboRemo potete connettervi al robot, con la funzionalità *Connetti*, specificando l'indirizzo IP e la porta indicati nel firmware. L'interfaccia di comando è illustrata nella **Fig. 5** e comprende i controlli visibili nella **Fig. 6**.

CONCLUSIONI

Con la componentistica proposta in questo articolo, il robot ha un movimento molto fluido e preciso, adattissimo ad esempio per un uso didattico; inoltre l'assorbimento di corrente e il consumo di potenza sono assai modesti, il che permette qualche ora di funzionamento anche utilizzando delle batterie stilo. Nulla vieta di sostituire i motori con altri dello stesso formato, ma con diverso numero di giri, per ottenere delle prestazioni più spinte, fermo restando che l'autonomia di

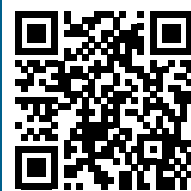
**Guarda EasyRobot
in funzione su YouTube!**



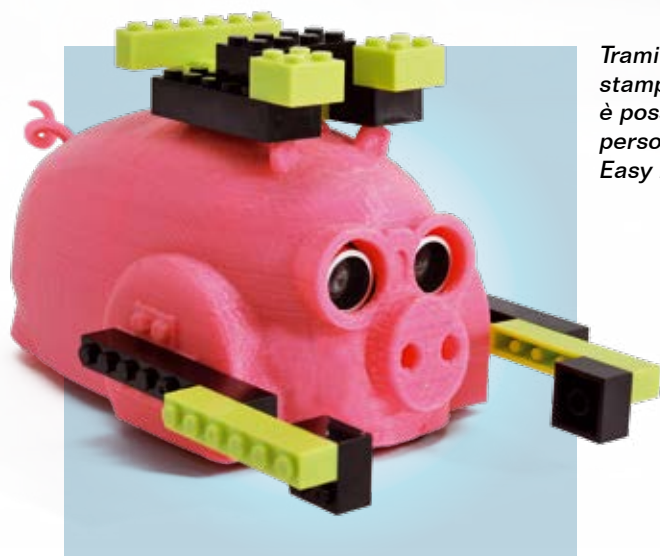
**Vuoi vedere
il nostro
EasyRobot in
azione? Nulla
di più semplice:
collegati alla
pagina**

<https://youtu.be/lxJm-Z5cSeY>.

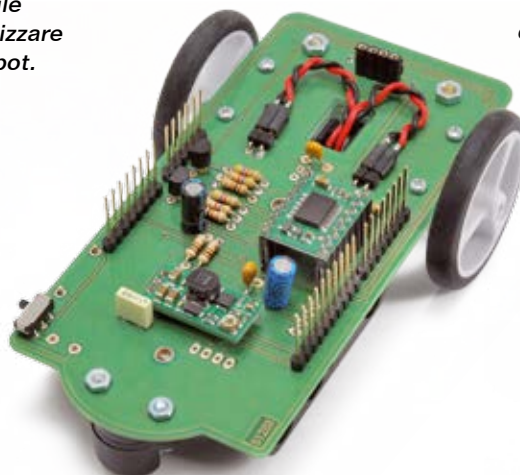
**Dal nostro canale YouTube
www.youtube.com/user/ElettronicaIN
potrai anche vedere i video
di moltissimi altri nostri progetti
proposti in passato.**



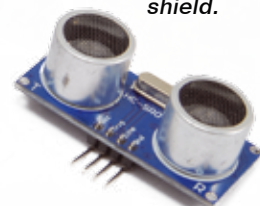
esercizio si ridurrà e che all'occorrenza occorrerà impiegare una batteria con maggiore corrente di spunto. Per darvi un'idea delle capacità del nostro Easy Robot, potete guardare la **Fig. 7**, che vi mostra il nostro piccolo robot in azione su un tracciato che l'autore ha realizzato appositamente per testarlo. ■



Tramite una stampante 3D è possibile personalizzare Easy Robot.



Il sensore che rileva la distanza è dotato di un pin-strip a quattro contatti (+, -, dell'alimentazione, oltre a trigger ed echo) che permette di innestarlo nello strip-femmina previsto sul lato superiore dello shield.



Quasi un anno fa abbiamo presentato Fishino UNO, una board Arduino-compatibile nata per l'accesso a Internet e al mondo dell'IOT senza bisogno di shield dedicati. Il riscontro favorevole ci ha spinto a realizzare due ulteriori unità, alimentabili a batteria ricaricabile e quindi utilizzabili in assenza di alimentazione di rete: Fishino Mega (rivista 204), per chi necessita più pin e memoria per le proprie applicazioni, ed il Fishino Guppy (rivista 206), il "pesce" più piccolo del nostro acquario, per applicazioni davvero miniaturizzate.

Che altro potevamo aggiungere? Guardandoci attorno e sviluppando, ci siamo resi conto che le limitazioni imposte dagli 8 bit dei processori Atmega 328p e Atmega 2560 iniziavano a farsi sentire, sia in termini di spazio per le applicazioni, sia in termini di prestazioni. Abbiamo quindi deciso (già da un po', a dire il vero, ma l'impegno sia dal lato hardware che dal lato software ha comportato lunghi tempi di sviluppo) di passare a un processore a 32 bit. Sì, ma quale? La scelta tra ciò che offre il mercato è stata

difficile.

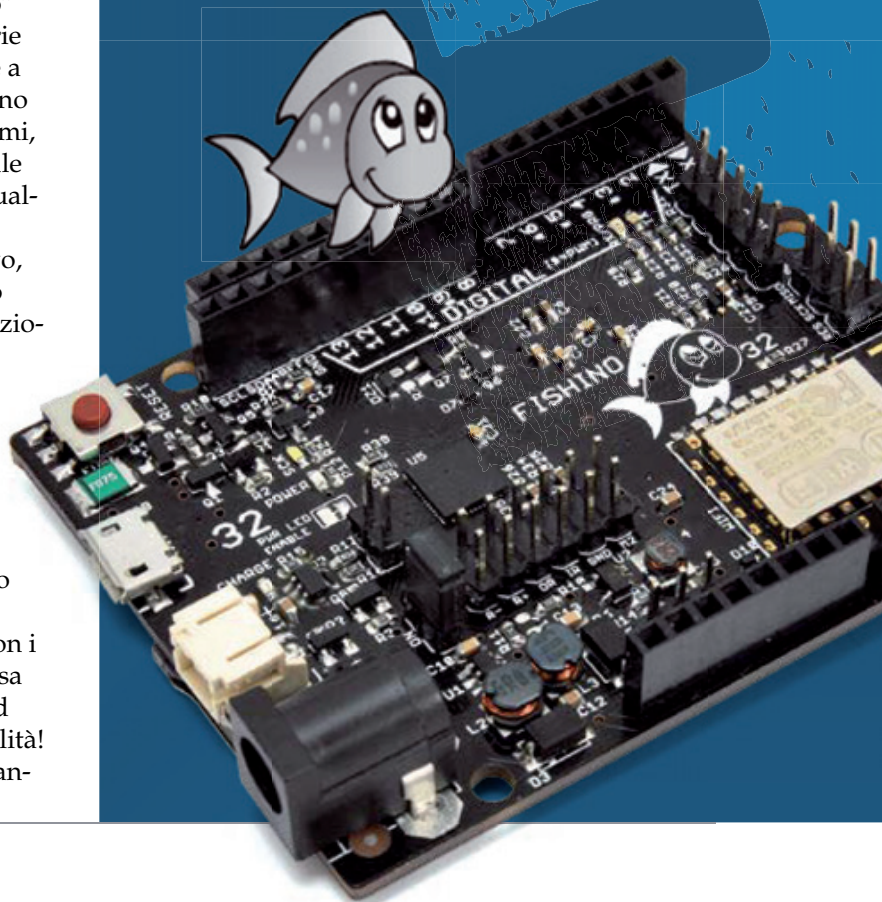
Per ragioni che andremo a spiegare a breve, siamo andati un po' controcorrente, abbandonando gli Atmel (e quindi gli ARM, nel campo dei 32 bit) in favore dei PIC della Microchip, nella versione a 32 bit, in architettura MIPS.

Inizialmente la scelta era caduta sulle "nuovissime" MCU PIC32MZ, la serie più prestante, ma in seguito, a causa dell'immaturità hardware e soprattutto dell'assenza attuale di uno stack USB veramente open source, abbiamo "ripiegato" sulla collaudata serie PIC32MX che, oltre a non soffrire di alcuno dei suddetti problemi, ha il non trascurabile pregio di costare qualcosa in meno.

È un ripiego relativo, visto che il modello utilizzato ha prestazioni di tutto rispetto, ovvero un clock a 120 MHz, una Flash Memory da 528 kByte ed una memoria RAM da 128 kByte: un abisso rispetto a quanto eravamo abituati con i modelli ad 8 bit, cosa che apre a nuove ed interessanti possibilità! Avendo fatto il "gran-

FISHINO A 32 BIT!

di MASSIMO DEL FEDELE



La nostra scheda di prototipazione guadagna il processore a 32 bit e punta in alto, verso applicazioni impegnative che possono sfruttare a pieno il supporto WiFi, l'RTC e l'SD-Card che distinguono Fishino dalla comune Arduino UNO. Prima puntata.



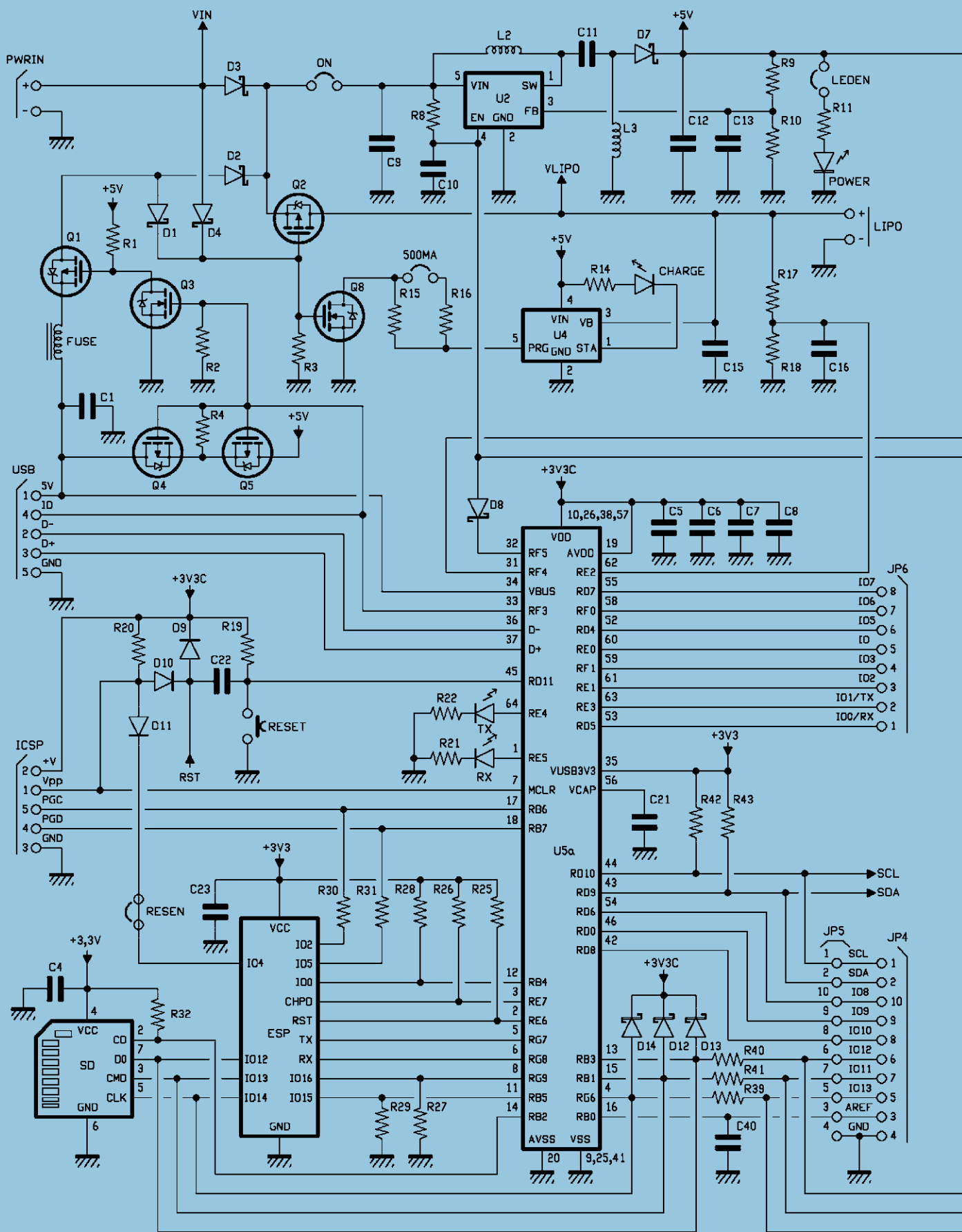
de balzo" nel mondo dei 32 bit, ci siamo poi chiesti se era possibile aggiungere qualcos'altro alla nostra scheda capostipite, che avevamo già ipotizzato nel formato collaudato e molto amato del Fishino UNO. Il PIC ha un'interfaccia USB incorporata (nativa), cosa che ci fa risparmiare lo spazio del convertitore; abbiamo inoltre deciso di sopprimere l'RTC esterno, visto che il controller utilizzato ne ha uno interno, dotato di molte possibilità aggiuntive rispetto al DS1307 utilizzato nel Fishino UNO, come per esempio la capacità di risvegliare il controller dallo standby ad orari o intervalli di tempo prefissati.

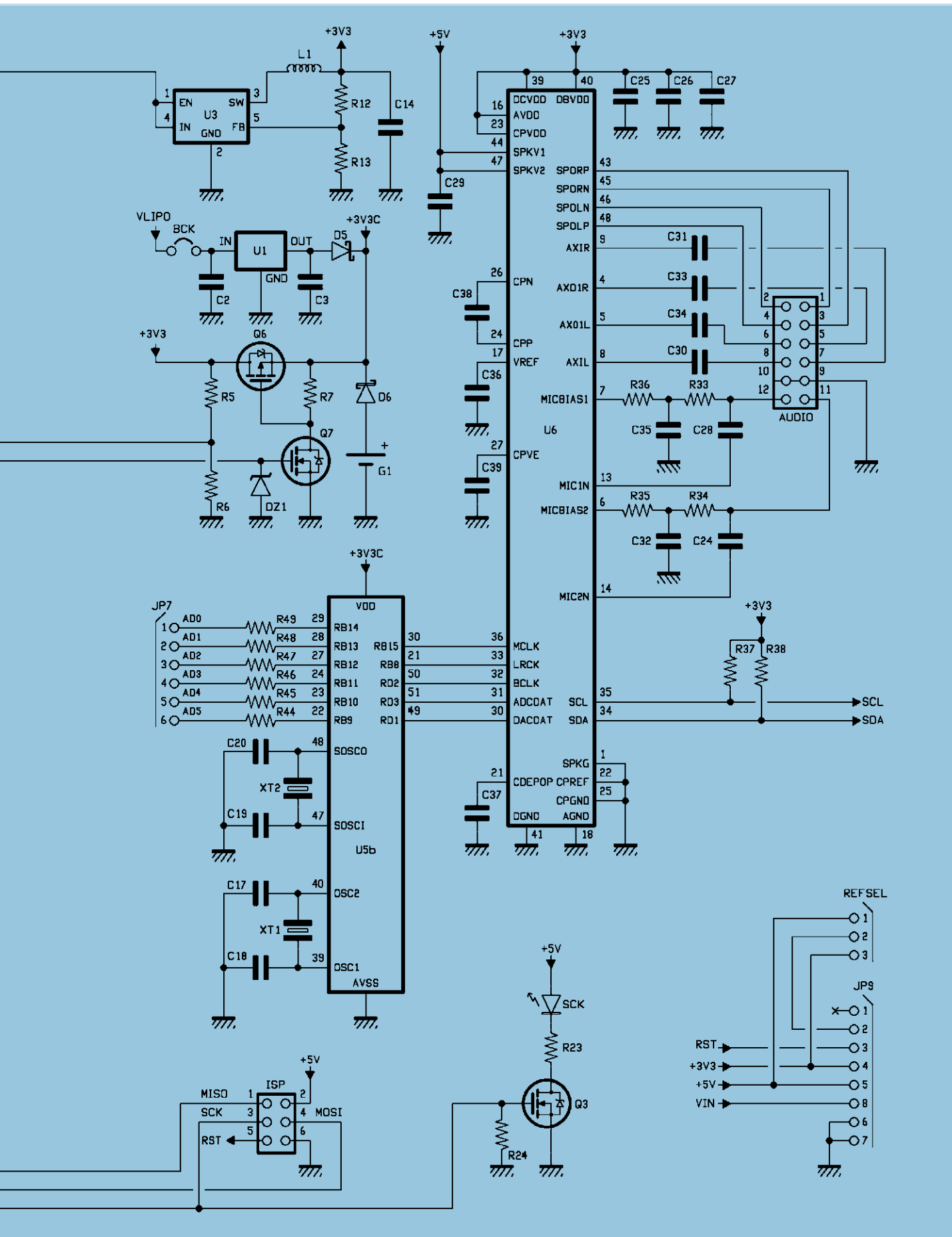
Come utilizzare, quindi, lo spazio aggiuntivo? Per prima cosa abbiamo inserito l'ormai collaudatissimo alimentatore SEPIC, già presente nel Mega e nel Guppy, rendendo quindi possibile un'alimentazione a batteria. Che altro potevamo inserire? Beh, un bel codec audio! Il codec scelto è un collaudatissimo ALC5631Q, le cui caratteristiche complete elencheremo nella sezione a esso dedicata; al momento vi basti sapere che include

un DAC (convertitore digitale/analogico) stereo, un ADC (convertitore analogico/digitale) stereo, un amplificatore in classe D in grado di pilotare direttamente una coppia di altoparlanti, ingressi microfoni, equalizzatore, effetti audio quali pseudostereo e stereo spaziale ...e molto altro. Con questo componente è, per esempio, possibile realizzare, senza bisogno di hardware aggiuntivo, un riproduttore audio, un feedback vocale per i propri progetti (che dire, ad esempio, di far leggere la temperatura di una sonda a voce dal nostro Fishino ?), un registratore audio, un controllo vocale, eccetera. È persino possibile sfruttare il codec per realizzare un sintetizzatore polifonico, cosa che interesserà gli appassionati di musica.

Ecco quindi le caratteristiche del nostro Fishino 32, capostipite di una serie a 32 bit di cui abbiamo già pronto il secondo modello, di caratteristiche inferiori ma con dimensioni estremamente ridotte, che presenteremo a breve:

- processore PIC32MX470F512;
- 512 kB di ROM;
- 128 kB di RAM;
- clock a 120 MHz;





- interfaccia USB nativa, sia Device che Host;
- RTC incorporato;
- supporto schede microSD;
- modulo WiFi;
- codec audio;
- alimentazione a batteria, plug e/o via connettore USB, da 3 volt circa fino a 20 volt;
- ricarica automatica di una batteria LiPo, se presente l'alimentazione esterna;
- possibilità di spegnere via software tutta la scheda, lasciando alimentato solo il processore in standby;
- possibilità di risveglio dallo standby tramite pin esterno o in tempi prefissati, da RTC;
- formato standard Arduino UNO;
- funzionamento interno a 3,3 volt;
- pin DIGITALI 5V-tolerant, quindi non c'è pericolo di bruciare la scheda con shield a 5 volt.

Abbiamo poi aggiunto una "chicca": grazie a un bootloader appositamente studiato, non è più necessario effettuare alcun collegamento esterno per aggiornare il firmware del modulo WiFi; con una pressione prolungata del tasto reset la scheda si predisporrà automaticamente in modalità aggiornamento, sfruttando il connettore USB. Qualcuno sarà probabilmente preoccupato dalla difficoltà di programmare questa nuova scheda, ma possiamo rassicurarlo; un po' grazie al lavoro del team di Arduino che ha reso l'IDE "estensibile" in modo estremamente semplice, un po' per il lavoro del team di Chipkit che abbiamo parzialmente sfruttato (anche se con parecchie modifiche), un po' grazie ad un buon numero di librerie software da noi scritte e/o adattate, la scheda

si programma esattamente come un normalissimo Arduino UNO, tramite l'IDE e con la sintassi cui siamo abituati.

È sufficiente eseguire una piccola procedura nell'IDE stesso, che permette di attivare gli strumenti di sviluppo per le MCU PIC ed il gioco è fatto. Per i componenti aggiuntivi abbiamo predisposto una serie di librerie che permetteranno di sfruttarli appieno; alcune di queste (soprattutto la libreria audio) sono in fase di sviluppo e quindi verranno aggiornate frequentemente. Sperando di aver stimolato il vostro "appetito" di maker, passiamo all'esame dello schema elettrico.

SCHEMA ELETTRICO

Partiamo dallo stadio di alimentazione, che risulta decisamente più complesso rispetto a quello dei modelli precedenti, e questo per le seguenti ragioni:

- la necessità di funzionare sia come host che come device USB;
- la tripla alimentazione;
- la possibilità di spegnimento software di tutto lo stadio di alimentazione;
- la possibilità di attivare esclusivamente l'alimentazione del controller in standby.

Iniziamo dalla sezione di controllo dell'alimentazione USB; questa sezione dev'essere in grado di fornire alimentazione al Fishino dal connettore USB se connesso come device, mentre deve fornire un'adeguata alimentazione a un device eventualmente connesso al Fishino, che in questo caso si comporta da host USB.

Per poter eseguire la commutazione ci viene in aiuto il piedino che fa capo alla linea U_{SBID} , collegata sia al connettore USB che all'apposita porta del controller.

I cavetti microUSB sono costituiti in modo da avere ad un estremo il pin corrispondente all' U_{SBID} connesso a massa (tipo A), mentre all'altro estremo questo pin è scollegato (tipo B); l'apparecchiatura cui è connessa l'estremità di tipo A assume il ruolo iniziale di host e contemporaneamente fornisce l'alimentazione all'altra apparecchiatura (connessa, questa, all'estremo di tipo B) che assume il ruolo iniziale di device. Abbiamo parlato di ruoli iniziali per un motivo ben preciso: in una connessione OTG i due device connessi possono cambiare ruolo; la specifica garantisce soltanto che la linea verrà alimentata sempre dal device connesso all'estremità A.

Torniamo quindi al nostro schema; vediamo che alla linea U_{SBID} è connessa una resistenza di valore elevato (R2, 100 kohm) verso massa, il cui scopo è soltanto quello di non lasciare fluttuante la linea in caso di assenza di alimentazione, ed una resistenza, R4, di valore decisamente inferiore, (10 kohm) connessa tra i due source dei MOSFET Q4 e Q5. Come si evince dallo schema, su questi source, grazie ai diodi interni ai MOSFET, è sempre presente una tensione positiva (5V meno la caduta diretta sui diodi) proveniente o dalla presa USB o dall'alimentazione a 5V della scheda Fishino. Abbiamo quindi la certezza che la R4 sia connessa ai +5V in qualsiasi condizione di funzionamento. Se la linea U_{SBID} è scollegata da massa (connessione tipo B) questa tensione positiva, attraverso R4, porta in conduzione il MOSFET a canale N Q3, il quale a sua volta manda in conduzione il MOSFET a canale P Q1; contemporaneamente, sempre attraverso R4 i due MOSFET Q4 e Q5 visti precedentemente vengono posti in interdizione e,

essendo collegati in antiserie, non permettono il passaggio dell'alimentazione dai 5V di Fishino al connettore USB.

Sintetizzando, a U_{SBID} fluttuante, quindi quando inseriamo un connettore di tipo B, il fishino NON fornisce alimentazione al connettore USB che, per contro, fornisce alimentazione al Fishino attraverso il MOSFET Q1. Otteniamo quindi la prima parte dello scopo desiderato, ovvero che con un connettore di tipo B Fishino ricava l'alimentazione dalla linea USB comportandosi da device vero e proprio.

Se inseriamo, per contro, un connettore di tipo A, la linea U_{SBID} viene posta a massa, cosa che porta in conduzione Q4 e Q5 e contemporaneamente, sempre attraverso Q3, in interdizione Q1; l'alimentazione fluisce quindi dal Fishino verso il connettore USB e non rientra "dalla porta di servizio" essendo Q1 interdetto.

È da notare, ovviamente, che nel secondo caso (connessione di

tipo A) per poter fornire i 5V alla porta USB il Fishino dev'essere alimentato in qualche altro modo, ovvero dalla batteria o dal plug VIN, come vedremo in seguito.

Proseguendo verso il basso del circuito notiamo la serie di diodi D1+D4 e D2+D3 rispettivamente, i primi due dei quali sono per piccole correnti, mentre i secondi sono diodi di potenza. Vediamo inoltre il MOSFET Q2, al quale giunge l'alimentazione della batteria.

Anche qui il funzionamento è abbastanza sottile; i diodi di potenza (D2 e D3), insieme al MOSFET Q1, costituiscono una porta OR di potenza, che confluisce nella linea V_{OR} dello schema. Il funzionamento dei diodi è abbastanza semplice: la tensione maggiore tra V_{IN} e V_{USB} viene trasmessa su V_{OR} , mentre l'altra, a causa dell'interdizione del diodo corrispondente, che si trova polarizzata inversamente, viene isolata. Ci si potrebbe chiedere perché sulla linea della batteria non abbiamo

inserito semplicemente un diodo di potenza come per le altre due, cosa che effettivamente avrebbe semplificato il circuito, realizzando una porta OR di potenza con tre diodi. Questo è fattibile, ma c'è il solito problema della caduta di tensione sui diodi che, se per tensioni elevate è praticamente ininfluenza, su una tensione ridotta come quella della batteria porta a perdite considerevoli: intuitivamente, si può dire che una caduta di tensione di 0,5 volt su 10 volt corrisponde al 5%, mentre una caduta di 0,5 volt su 3,6 volt è pari a circa il 14%, cosa che si riflette sull'assorbimento in corrente come vedremo di seguito, e proprio dove servirebbe una maggior efficienza, ovvero nell'alimentazione a batteria.

Il mosfet Q2 non ha questo problema, visto che non presenta una caduta di tensione fissa come un diodo ma una resistenza di conduzione, particolarmente bassa nel modello selezionato; per contro, un MOSFET non ha il

CARATTERISTICHE TECNICHE

- **Alimentazione:**
 - 5 V tramite connettore USB
 - 5 V sull'ingresso +5V
 - da 3,5 a 20 V sull'ingresso Vin
 - da 3,5 a 20 V sul plug di alimentazione
 - batteria ricaricabile agli ioni di litio, a cella singola da 3,7 V, sul connettore BAT
 - caricabatteria incluso
- **Livelli digitali:**
 - 3,3 V (5 V tollerati sugli ingressi digitali eccetto i due aggiuntivi caratteristici Fishino)
- **Controller:**
 - PIC32MX470F512H
- **Clock:**
 - 120 MHz, riducibile via software
- **Memoria:**
 - 512 kByte Flash
 - 128 kByte RAM
 - EEPROM emulata via software
- **Porte disponibili:**
 - 14 digitali di cui 5 in PWM
 - 6 analogiche
 - 2 digitali a 3,3 V
- **Interfacce disponibili:**
 - 1 x SPI esterna condivisa con WiFi ed SD
 - 1 x SPI/I²S interna dedicata al codec
 - 1 x I²C
- **Moduli aggiuntivi sulla scheda:**
 - WiFi
 - Connettore per schede microSD
 - RTC incorporato nel processore
 - Codec Audio ad alta definizione
 - Controllo software completo dell'alimentazione



Il PIC32MX470F512

Il microcontrollore PIC utilizzato nella nostra board appartiene alla famiglia PIC32MX, basata sull'architettura MIPS, che si distingue subito dalla classe di microcontrollori che equipaggiano Arduino, i quali sono Atmel AVR, basati su un core ARM.

Lo schema a blocchi che trovate in questo riquadro mostra l'architettura interna, a partire da un core MK4, in grado di funzionare fino a 120 MHz di clock, con una capacità di elaborazione di ben 198 DMIPS. Il core è dotato di capacità di moltiplicazione/divisione hardware a 32 bit, di un sistema per il debugging del codice e di altre notevoli caratteristiche, oltre ad essere contornato da una ricca dotazione di periferiche incorporate, una generosa dotazione di memoria Flash e RAM. Cosa non trascurabile per lo sviluppo di board di sviluppo e prototipazione, la MCU Microchip da noi impiegata in Fishino 32 offre la possibilità di "spostare" le connessioni alle periferiche incorporate su vari I/O, permettendo quindi una notevole flessibilità nella progettazione.

Il modello da noi scelto è al top delle prestazioni ed è dotato, inoltre, di un modulo USB OTG, che ne permette un

facile interfacciamento sia con device che con host USB esterni.

Il controller è inoltre dotato di un completo sistema di power management, che permette di spegnere le periferiche non utilizzate, ridurre la velocità di clock per limitare i consumi e "ibernare" tutto il sistema, mantenendo acceso lo stretto indispensabile per poter reagire agli eventi esterni. Risulta quindi possibile, tramite software, "calibrare" il consumo della scheda a seconda delle necessità, fino all'arresto pressoché completo, allo scopo di aumentare fino al limite desiderato l'autonomia, nel funzionamento a batteria.



pregio del diodo di condurre solo quando serve, per cui occorre un sistema per mandarlo in conduzione o interdirlo a seconda delle necessità. Per questo utilizziamo i due diodi per piccole correnti D1 e D4; il funzionamento del tutto è il seguente:

- se non è presente un'alimentazione su V_{IN} e nemmeno attraverso V_{USB} , la resistenza R3 porta a massa il gate di Q1 che quindi conduce e permette il passaggio della tensione della batteria verso V_{OR} ;
- se invece è presente una delle due (V_{IN} o V_{USB} , o entrambe) uno dei diodi D1 e D4 conduce, polarizzando positivamente Q2 che passa in interdizione scollegando quindi la batteria.

In breve, se abbiamo un'alimentazione esterna, che proviene dal

plug V_{IN} o dal connettore USB, la batteria viene scollegata; se invece manca l'alimentazione esterna entra in funzione la batteria.

Si può notare che la linea su cui confluiscono i diodi D1 e D4 ha un nome, BATOFF, perché questo segnale, che è a livello alto se è presente un'alimentazione esterna, verrà utilizzato anche in un'altra sezione del circuito. Vediamo ora il gruppo di componenti che fanno capo ai MOSFET Q6 e Q7; questi servono per scollegare la linea +3V3, ovvero la linea principale a +3,3 V, dalla linea +3V3Core, destinata ad alimentare solo il controller. A cosa servono? Semplicemente ad evitare che un'alimentazione destinata al solo controller, attivata quando tutto il resto è spento, e che ha lo scopo di mantenere attivo l'RTC interno ed altre

funzionalità minime, confluisca nei circuiti che si vogliono tenere spenti.

Normalmente, se è presente tensione sui +3,3V, questa confluisce attraverso il diodo interno a Q6, e anche attraverso il Q6 stesso se questo è in conduzione, sull'alimentazione +3V3Core del controller; è questo il funzionamento normale. Se noi invece togliamo l'alimentazione sui +3,3V, e contemporaneamente portiamo a livello basso la linea OFF, attraverso Q7 portiamo il Q6 ad interdizione e quindi impediamo ai +3V3Core di andare ad alimentare il resto della scheda. La linea OFF viene comandata dal controller, come vedremo in seguito, durante lo spegnimento programmato.

Si possono poi notare la pila al litio di mantenimento G1 e il diodo D6, che servono a fornire tensione al controller quando manca qualsiasi altra fonte in grado di alimentarlo. Per questo occorre fare attenzione: se togliamo completamente l'alimentazione al Fishino, questa pila finisce per alimentare tutta la scheda, a meno di non "prendere provvedimenti" attraverso il software portando a livello basso la linea OFF. Quindi, a differenza delle schede ad 8 bit, dove la pila serviva solo per la tensione di mantenimento dell'RTC, e la cosa veniva gestita in modo automatico, qui è necessario un intervento del software a supporto, altrimenti la pila si scaricherebbe in pochi minuti. Consigliamo quindi di non utilizzarla e lasciare il mantenimento alla batteria LiPo, a meno che non si sappia come gestire il risparmio energetico della scheda via software. Per ultima notiamo la linea PWRGOOD, connessa ad un partitore con la massa; questa fornisce un segnale positivo ad

un I/O del controller che indica la presenza di un'alimentazione adeguata sulla scheda, e può venire utilizzato, appunto, per gestire la mancanza di questa e le funzioni di standby. Come anzidetto, si tratta di una gestione piuttosto complessa, per la quale è in preparazione un'apposita libreria software; per il momento, a meno che non sappiate come gestire la cosa, consigliamo di non utilizzare le funzioni di spegnimento software.

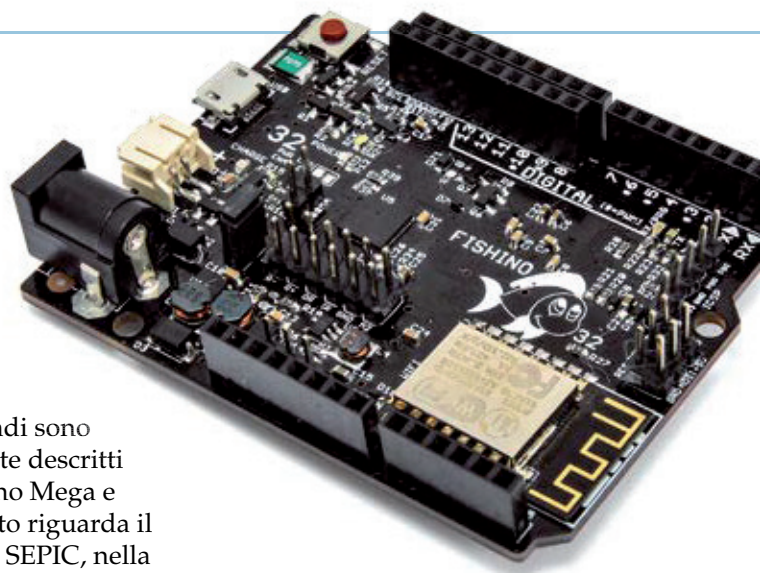
L'ultima parte della sezione in esame è quella che ruota attorno al regolatore lineare U1, un comune regolatore a bassa corrente (max 100 mA circa) e bassissimo consumo, destinato a fornire la tensione di mantenimento al controller tramite la batteria LiPo esterna. Questo ha lo stesso effetto della pila al litio G1, con l'indubbio vantaggio di utilizzare una batteria ricaricabile in grado di fornire un'autonomia estremamente lunga anche in presenza di consumi non proprio trascurabili. Questa sezione è disattivabile tagliando il ponticello BCK, normalmente connesso. Anche qui valgono (anche se in modo meno pressante) i discorsi sul risparmio energetico fatti prima: se spegniamo via software la scheda, il controller prenderà alimentazione dalla batteria LiPo, che quindi tenderà a scaricarsi; è compito del software fare in modo che questo venga posto in standby o comunque in una delle modalità a basso consumo disponibili.

Passiamo adesso alla sezione dello stadio switching, che fa capo agli integrati U2 (SX1308), un convertitore asincrono step-up, utilizzato qui in modalità SEPIC, in grado di fornire in uscita 5V con tensioni in ingresso variabili da circa 3, fino ad oltre 20 volt, ed all'integrato U3, che

è un convertitore sincrono step-down, utilizzato per ricavare i 3,3 volt per l'alimentazione della maggior parte del circuito. Entrambi questi stadi sono già stati ampiamente descritti per le schede Fishino Mega e Guppy e, per quanto riguarda il funzionamento del SEPIC, nella scheda Torpedo, pubblicata sul numero 204 di Elettronica In. Le uniche differenze consistono nella presenza del jumper ON, che permette l'inserimento di un interruttore di spegnimento in grado di togliere l'alimentazione a tutta la scheda (fatto salvo quella di mantenimento eventualmente abilitata, come spiegato nel paragrafo precedente), e la linea OFF, già vista prima, in grado di spegnere, se portata a massa, il convertitore switching. Questa linea è sotto diretto controllo del processore, attraverso un I/O, e permette quindi sia lo spegnimento completo della scheda via software, sia l'isolamento della sezione di alimentazione del solo controller dal resto del circuito, come visto nel punto precedente.

Riassumendo, ponendo la linea OFF a massa viene tolta tutta l'alimentazione alla scheda; ovviamente questa linea deve essere mantenuta a massa dal controller, che quindi deve rimanere operativo attraverso una delle due sorgenti di mantenimento (LiPo o pila al litio); in assenza di queste, portando la linea OFF a massa otterremo solo un reset della scheda ed il conseguente riavvio del tutto.

Quanto al caricabatteria, fa capo all'onnipresente MCP73831, un completo regolatore di carica integrato per LiPo a singola cella;



è lo stesso della Fishino Mega, quindi rimandiamo al relativo articolo per i dettagli. L'unico punto degno di nota è costituito dal MOSFET Q9, un canale N, utilizzato per abilitare o meno la funzione di carica della batteria. Se ricordiamo la descrizione del primo stadio (controllo alimentazioni), la linea BATOFF è a livello alto quando è presente l'alimentazione esterna; in questo caso il MOSFET Q9 va in conduzione, portando a massa la resistenza R15 (ed eventualmente la R16 se abbiamo selezionato la carica a 500 mA) attivando quindi la carica. In assenza di tensioni esterne (alimentazione a batteria) la linea BATOFF va a livello basso, disattivando la carica.

Notate le resistenze R17 ed R18, che costituiscono un partitore di tensione in grado di fornire metà della tensione della batteria LiPo, misurabile attraverso una linea del controller per poter segnalare lo stato di carica della stessa. Abbiamo utilizzato un partitore perché la tensione della batteria LiPo, a piena carica, supera i 3,3 V accettati dagli ingressi analogici del controller; occorre tenerne quindi conto nella misura.

STADIO CONTROLLER

Arriviamo finalmente al PIC32MX470F512, del quale al momento ci limiteremo ad

esaminare alcune connessioni, lasciando il resto agli altri stadi per i dettagli. Notiamo innanzitutto due quarzi, uno da 20 MHz e l'altro da 32.768 kHz; servono perché il primo fornisce il clock principale, attraverso un PLL interno, che lo moltiplica fino a 120 MHz. Si tratta di un circuito non proprio a basso consumo, quindi da evitare nelle condizioni di risparmio energetico; per contro, il quarzo da 32.768 kHz serve esclusivamente per il real time controller (RTC) interno alla PIC, che è un circuito studiato appositamente per avere un bassissimo assorbimento e che può quindi essere lasciato in funzione anche in condizioni di standby.

Tornando al primo quarzo, accettando una precisione minore sulla frequenza questo avrebbe potuto essere rimosso; la PIC ha infatti un oscillatore RC piuttosto preciso al suo interno in grado di farne quasi completamente le veci. "Quasi", perché l'oscillatore interno ha due difetti:

- non ha la precisione del quarzo (questo ha un'importanza relativa);
- non è in grado di portare la PIC alla sua massima frequenza di 120 MHz, a causa di vincoli del PLL interno, che sarebbe quindi limitata a 96 MHz.

Il primo punto ha importanza solo se utilizziamo la scheda per misure precise di tempi brevi, ad esempio come frequenzimetro o periodometro; per i tempi lunghi basterebbe l'altro quarzo.

Il secondo punto è vincolante se vogliamo "spremere" il controller al massimo delle sue possibilità, e quindi l'abbiamo inserito!

Si notino, tra le svariate connessioni sul controller, le tre linee viste in precedenza, ovvero la OFF, che gli permette di togliere

l'alimentazione principale, la PWRGOOD, che consente il controllo dello stato della medesima, e la linea MBAT che permette la misura della tensione della batteria.

Diamo adesso uno sguardo alla sezione reset, interessante perché ci ha permesso di utilizzare un solo pulsante di reset per svariate funzioni, in abbinamento con un apposito bootloader sul controller. Notiamo subito tre linee che fanno capo allo stadio di reset:

- MCLR;
- REMOTERESET;
- LONGRESET.

La linea MCLR è quella di reset vero e proprio del controller; portandola a massa il medesimo viene riavviato.

Notiamo subito che questo può avvenire sia premendo il pulsante di reset, che attraverso il condensatore C22 invia un breve impulso negativo alla linea MCLR, sia portando a massa la linea REMOTERESET, proveniente dal modulo WiFi e che consente quindi a questo di resettare il controller.

La linea LONGRESET, per contro, viene a trovarsi a massa per tutto il tempo in cui il pulsante di reset viene mantenuto premuto, consentendo quindi al bootloader di conoscere le nostre intenzioni.

In sintesi:

- una breve pressione del pulsante reset (inferiore ai 2 secondi) o un segnale negativo sulla linea REMOTERESET riavviano la scheda, come d'abitudine;
- mantenendo premuto il pulsante di reset per un tempo tra 2 e 4 secondi viene avviato il bootloader (cosa evidenziata dal lampeggio del led arancione), mettendo quindi la scheda in modalità caricamento sketch;

- mantenendo premuto il pulsante per un tempo superiore ai 4 secondi si avvia la procedura di upgrade del firmware del modulo WiFi, cosa evidenziata dal lampeggio del LED blu.

È comunque possibile, una volta entrati nelle 2 modalità del bootloader, riavviare semplicemente la scheda ed eseguire lo sketch precaricato ripremendo brevemente il pulsante di reset. Per quanto riguarda la sezione led, si tratta semplicemente di due led connessi ad altrettante porte del controller, utilizzabili per qualsiasi compito (indicati sulla scheda come RX/PRG e TX, rispettivamente) e del solito LED connesso alla porta 13 che viene utilizzato nei test della scheda (programma BLINK), questo attraverso il MOSFET Q9 per non caricare l'uscita D13, e che tra l'altro riflette lo stato della linea di clock (SCK) dell'interfaccia SPI, cosa che si nota utilizzando la scheda microSD o il modulo WiFi.

Passiamo adesso alla sezione WiFi, simile a quella delle Fishino precedenti, perché il modulo ha sempre un firmware da noi realizzato, in grado di comunicare ad alta velocità tramite la porta SPI e non la porta seriale.

In questa scheda è stato eliminato il connettore ESPCONN al quale eravamo abituati nelle board precedenti, ed al quale facevano capo alcuni I/O e pin di controllo del modulo sfruttati sia per programmarlo che, per alcuni di essi, come I/O aggiuntivi. Questo perché nel Fishino32 abbiamo già liberato alcuni degli I/O che venivano usati per comunicare con il modulo e con la scheda SD, che erano il 4, 7 e 10; abbiamo inoltre separato il bus I²C dai due ingressi analogici A4 e A5, ottenendo quindi 3 porte

digitali e due porte analogiche in più a disposizione.

Non solo, ma due porte sul connettore ICSP sono sfruttabili come porte digitali aggiuntive (ne parleremo in dettaglio descrivendo il software), quindi abbiamo preferito sfruttare le connessioni con il modulo ESP in altro modo. Infatti, tutte le linee dell'ESP (salvo l'ingresso analogico) sono ora sotto controllo del processore principale, cosa che ci permette di effettuare le seguenti operazioni direttamente da software, senza impegnare alcun I/O:

- resettare il modulo da software;
- spegnere il modulo da software (cosa che in precedenza richiedeva l'uso di un I/O del Fishino);
- caricare il firmware sul modulo senza dover effettuare collegamenti strani né caricare sketch particolari;
- ridirigere l'output seriale del modulo sulla seriale USB o su qualsiasi altra seriale, sia software che hardware, disponibile sul Fishino.

Per il resto, nulla cambia rispetto alle schede precedenti; la comunicazione avviene attraverso la porta SPI, che fa capo anche alla scheda SD ed ai pin digitali D11, D12 e D13 sul connettore esterno. Visto che la PIC utilizzata ha due interfacce SPI incorporate, verrebbe da chiedersi perché non è stata dedicata una di queste ai moduli inclusi (WiFi e SD) liberando così anche gli I/O 11, 12 e 13 completamente; questo è stato fatto perché la seconda interfaccia SPI è dedicata, in modalità I²S, al codec audio che quando è in funzione la impegna a tempo pieno. In un primo tempo avevamo scelto quella strada, salvo accorgerci in seguito che risultava impossibile, ad esempio,

Poiché una trattazione completa di tutte le librerie disponibili richiederebbe troppo spazio, la rimandiamo ad articoli successivi; al momento vi proponiamo uno sketch dimostrativo, ma perfettamente funzionante e piuttosto piacevole, di un audio player.

Per poterlo utilizzare è necessario seguire i seguenti passi :

- Preparare una scheda microSD con al suo interno alcuni files in formato .wav. Attualmente il player non supporta il formato .mp3 (in preparazione!), quindi eventuali .mp3 vanno convertiti in .wav utilizzando, ad esempio, il programma open source Audacity, reperibile in rete. Nella conversione occorre scegliere il formato Microsoft WAV, contenente audio PCM non compresso, indifferentemente a 8, 16, 24 o 32 bit. Per un'ottima qualità consigliamo 16 o ancora meglio 24 bit.
- Una volta preparata la scheda con i file musicali, inseritela nel lettore sul Fishino, collegate una coppia di altoparlanti alle uscite L+ e L- (altoparlante sinistro) e R+ ed R- (altoparlante destro) sul connettore AUDIO, caricate lo sketch *FishinoPlayerDemo* e... godetevi un po' di musica!

Il player esegue tutti i brani in formato .wav presenti sulla scheda in sequenza, ripetendoli all'infinito.

Una nota: l'amplificatore audio sulla scheda dispone di una potenza non trascurabile (ovviamente rapportata alle dimensioni della scheda!), che può arrivare a 2,3W per canale. Questa potenza è eccessiva per l'alimentazione tramite USB, e rischia di resettare la scheda per insufficienza di corrente. In questo caso, le soluzioni sono tre:

- utilizzare un solo altoparlante;
- ridurre il volume audio;
- utilizzare un'alimentazione esterna, anche tramite la batteria LiPo; quest'ultima può tranquillamente essere usata insieme all'USB e provvederà a fornire i picchi di corrente necessari all'audio quando l'USB non ce la fa.

Un ultimissima ma importante nota: fate molta attenzione alla polarità della batteria LiPo, sulla scheda è indicata chiaramente e, nel caso la vostra batteria abbia una diversa polarità occorre invertire il connettore.

Per motivi di efficienza l'ingresso di alimentazione a batteria non è protetto da inversioni di polarità, quindi collegando la LiPo in marriera errata danneggerete irrimediabilmente la scheda!

riprodurre un file audio letto da scheda SD, visto che durante la riproduzione l'interfaccia era totalmente impegnata con il codec senza possibilità di accedere alla SD. Va detto che nelle schede in formato Arduino UNO i 3 I/O suddetti sono comunque dedicati alla porta SPI, e comunque se necessario abbiamo a disposizione i 2 I/O sulla porta ICSP di cui parlavamo prima, ed eventualmente tutti gli I/O analogici che sono comunque utilizzabili anche in digitale. L'interfaccia verso la microSD è identica a quella delle schede già viste; sfrutta l'interfaccia SPI, come il modulo WiFi, mentre anche qui per la selezione

della scheda viene utilizzato un I/O interno, lasciando libero quindi l'I/O D4 che prima era necessario per questo scopo. L'unica grossa differenza rispetto alle schede precedenti è l'assenza degli stadi di conversione livelli (da 5 a 3,3 V) visto che il PIC utilizza i 3,3 V come alimentazione, rendendo quindi possibile (e più veloce!) un collegamento diretto. Unico accorgimento da tenere, nell'esecuzione di sketch esistenti: il pin di selezione scheda (CS) non è più il 4 (o il 10 in alcune versioni di Arduino) ma un pin interno denominato SDCS; è sufficiente quindi sostituire tutti i riferimenti all'I/O 4 con l'I/O

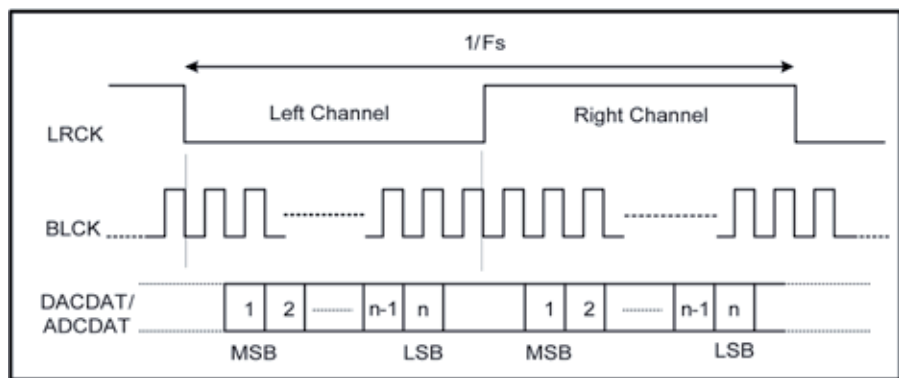


Fig. 1 - Segnali dell'SPI correlati.

SDCS (scritto proprio così, visto che può cambiare fisicamente da scheda a scheda).

STADIO CODEC AUDIO

Ed eccoci arrivati alla succosa novità presente nel Fishino32: il codec audio! Si basa su un chip, l'ALC5631Q, utilizzato in molti prodotti consumer (tablet, riproduttori audio, eccetera) e quindi dotato di prestazioni notevolissime pur conservando un prezzo più che accettabile. Il codec ha le seguenti caratteristiche:

- un convertitore D/A (DAC) stereo a 24 bit con un rapporto segnale/disturbo di 100 dB ed una distorsione totale a -90 dB;
- un convertitore A/D (ADC) stereo a 24 bit con un rapporto segnale/disturbo di 93 dB ed una distorsione totale a -88 dB;
- un amplificatore stereo in classe D, in grado di erogare 650 mW per canale se alimentato a 3,3 V su un'impedenza di 8 Ohm, e fino a 2,8 W per canale se alimentato (come nel nostro caso) a 5 volt e su un'impedenza di 4 ohm;
- un amplificatore per cuffie stereo, in grado di erogare 45 mW su 16 ohm, che purtroppo non utilizzeremo nella nostra scheda per motivi di spazio;
- 3 ingressi analogici differenziali o un ingresso analogico stereo (noi utilizziamo quest'ultimo);
- due ingressi microfonici analogici preamplificati, sia in modalità differenziale (da noi non usata) sia in modalità single-ended;
- un ingresso per microfono stereo digitale (anche questo, non utilizzato sulla nostra scheda sempre per motivi di spazio);
- un'uscita analogica di linea, alla quale è possibile connettere un amplificatore esterno;
- un completo sistema di power management che permette di spegnere i singoli componenti del codec se non utilizzati;
- un equalizzatore digitale a 6 bande audio;
- effetti di pseudo-stereo e di stereo spaziale;
- circuiti di rimozione dei disturbi di accensione e spegnimento negli altoparlanti;
- gestione automatica di livelli (ALC) sia in riproduzione che in registrazione;
- interfaccia standard I²S ad alta velocità (24 bit/da 8 a 192 kHz sul DAC in riproduzione e 24 bit/da 8 a 96 kHz sull'ADC in registrazione).

Come si può notare, una serie di caratteristiche impressionanti, che permetteranno di realizzare, senza alcun hardware aggiuntivo che non sia un altoparlante o un microfono, moltissimi progetti che coinvolgono l'audio digitale; per esempio:

- player audio (WAV e MP3);
- registratore audio;
- riconoscimento vocale;
- sintesi musicale;
- sintesi vocale.

Come vedrete in seguito, abbiamo già implementato un player per files .wav, ed è in realizzazione un completo lettore MP3; per la registrazione e le altre caratteristiche scriveremo le opportune librerie software in seguito.

Come già accennato parlando del modulo WiFi, il codec sfrutta un'interfaccia SPI dedicata, utilizzata in modalità I²S, per la gestione della quale abbiamo scritto un'apposita libreria software.

Il codec ha inoltre bisogno di una porta I²C (non necessariamente dedicata solo a lui, ed infatti abbiamo utilizzato la porta I²C standard del Fishino) per controllare tutte le varie funzioni disponibili. Anche qui abbiamo creato un'apposita libreria in grado di controllare il funzionamento del codec; tramite questa si può, per esempio, decidere se registrare o riprodurre, modificare il volume audio, regolare l'equalizzatore, impostare l'ALC, eccetera.

Come potrete ben immaginare i controlli sono numerosissimi, quindi la libreria verrà estesa man mano che se ne presenterà la necessità.

Sullo schema elettrico c'è pochissimo da aggiungere; come potete vedere, oltre al codec, qualche condensatore di disaccoppiamento sulle alimentazioni e sugli I/O, un condensatore per un convertitore interno destinato ad aumentare la tensione per l'amplificatore delle cuffie (C38) ed un paio di resistenze per la tensione di bias (polarizzazione) dei microfoni, non c'è altro!

Dicevamo dell'interfaccia I²S: questa è un'interfaccia basata sull'SPI prevista espressamente per

la trasmissione di flussi audio stereo ad alta velocità; a differenza della SPI, ed a causa del fatto che i segnali che vi transitano devono essere in tempo reale, è praticamente impossibile utilizzarla in contemporanea su più apparecchiature.

L'interfaccia utilizza gli usuali pin SDI (MISO), SDO (MOSI) e SCK (clock) per i dati, mentre il pin di selezione viene utilizzato in modo differente per selezionare il canale audio tra i 2 trasmettitori; è inoltre presente una linea addizionale di clock usata come riferimento.

Quindi abbiamo, nominando i pin dal lato master, ovvero dal punto di vista del controller:

- MISO o SDI > ADCDAT, ovvero i dati in ingresso dal convertitore A/D al controller;
- MOSI o SDO > DACDAT, ovvero i dati in uscita dal convertitore D/A verso la PIC;
- SCK > BLCK, ovvero il clock per i singoli bit trasmessi;
- SS > LRCK, ovvero il clock per differenziare canale sinistro e destro.
- MCLK > Clock di riferimento, abitualmente quattro volte l'SCK.

I segnali che transitano sono visibili nella **Fig. 1**, dove manca il clock di riferimento (MCLK) che è comunque un clock fisso di frequenza quadrupla rispetto al BLCK.

Lasciamo per il momento la descrizione del codec, che riprenderemo quando parleremo delle librerie software.

CONNETTORI D'ESPANSIONE

In questa sezione sono raggruppati tutti i connettori che portano i segnali verso l'esterno della scheda, quindi gli header laterali, il connettore ISP (che su Fishino32 non è utilizzato per pro-

grammare la scheda, ma inserito per compatibilità con gli shield esistenti), il connettore ICSP (In Circuit Serial Programming) che serve per programmare la scheda senza passare per il bootloader (ad esempio per programmare il bootloader stesso), e sul quale sono disponibili due I/O digitali aggiuntivi, ed il connettore USB. Come accennato parlando delle caratteristiche della scheda, la PIC ha molti, ma non tutti, i pin I/O 5V tolerant, ovvero in grado di accettare 5V in ingresso anche se il controller è alimentato a 3,3 V senza danneggiarsi. Dove possibile abbiamo sfruttato quei pin portandoli all'esterno come I/O digitali; purtroppo in alcuni casi questo non è stato possibile, ed abbiamo quindi inserito una protezione in ingresso tramite resistenze e diodi; è questo il caso dei segnali sulla porta SPI (I/O da 11 a 13) che, essendo necessari anche ai moduli interni, avremmo dovuto limitare comunque. La protezione consiste in una resistenza da 220 Ohm in serie all'ingresso ed un diodo in grado di scaricare l'eccedenza sulla linea a 3,3 V, limitando quindi la tensione in ingresso a tale valore. Sui pin analogici, che non sono 5V-tolerant (proprio a causa della circuiteria analogica interna al controller) abbiamo inserito solamente delle resistenze in serie per limitare un eventuale sovracorrente nel caso vi si inseriscano per errore dei valori superiori ai 3,3 V.

Attenzione, questo non significa che è possibile inserire 5 volt su quel connettore! Le resistenze offrono un minimo grado di protezione, che dovrebbe proteggere il controller per brevi periodi se per errore vi si inseriscono i 5 Volt, ma non bisogna farne conto ed utilizzarli stabilmente con quei valori di tensione!

L'integrato dispone infatti di diodi di protezione al suo interno, che fanno le veci di quelli da noi inseriti esternamente sugli ingressi D11..D13; questi diodi sono però in grado di assorbire una corrente molto limitata e per brevi periodi di tempo, quindi non ci si deve fare assolutamente conto.

Nella sezione header trovate anche un connettore denominato REFSEL, che serve per selezionare la tensione che verrà connessa al pin VREF; è infatti possibile farvi giungere i 5V, nel caso si utilizzino shield "normali" funzionanti a quella tensione, oppure i 3,3 V destinati a shield a bassa tensione come quelli utilizzati nell'Arduino DUE.

CONCLUSIONI

Bene, con questo abbiamo concluso, almeno per questa prima puntata.

Termineremo il discorso nella prossima, affrontando la realizzazione della nostra Fishino 32 e la messa in esercizio, scoprendo e utilizzando i tool di Arduino e librerie specifiche importabili grazie alle funzioni di acquisizione di hardware di terze parti implementate nelle più recenti versioni dell'IDE.



per il MATERIALE

La board Fishino32 (cod. FISHINO32) viene fornita montata e collaudata. Può essere acquistata presso Futura Elettronica al prezzo di Euro 59,00. Il prezzo si intende 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>

Fishino

Design With Simplicity

I TUOI PROGETTI DIVENTANO WIRELESS

FISHINO UNO

- Alimentazione:
 - da 7 a 12 Vdc (tramite plug)
 - 5 Vdc (tramite porta USB)
- Compatibile al 100% con Arduino UNO
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Modulo RTC con batteria di mantenimento
- Sezione di alimentazione a 3,3 V potenziata
- Compatibile con shield e schede millefori
- Peso 25 g
- Dimensioni 76,5 x 53,5 x 14 mm



cod. FISHINOUNO
€ 36,⁰⁰

cod. FISHINOMEGA
€ 49,⁹⁰



FISHINO MEGA

- Alimentazione:
 - 3,6 Vdc (tramite batteria esterna al litio)
 - da 3,5 a 20 Vdc (tramite plug)
 - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile al 100% con Arduino MEGA
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Modulo RTC con batteria di mantenimento
- Stadio di alimentazione switching (5 Vdc e 3,3 Vdc)
- Compatibile con shield e schede millefori
- Peso 37 g
- Dimensioni 101,5 x 53,5 x 15 mm



FISHINO32

- Controller PIC32 bit
- Alimentazione:
 - 3,6 Vdc (tramite batteria esterna al litio)
 - da 3,5 a 20 Vdc (tramite plug o sull'ingresso Vin)
 - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile con Arduino UNO
- Modulo WiFi
- Modulo RTC con batteria di mantenimento
- Clock 120 MHz, riducibile via software
- Interfaccia per scheda microSD
- Codec audio stereo
- Peso 29 g
- Dimensioni 76,5 x 53,5 x 14 mm

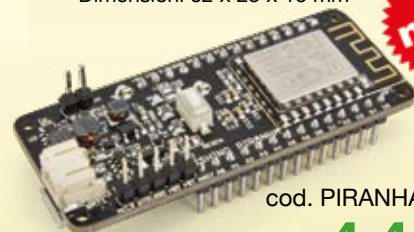
cod. FISHINO32
€ 59,⁰⁰



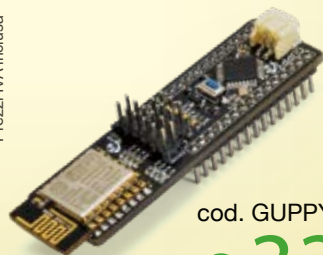
Fishino UNO, Fishino MEGA, Fishino GUPPY, Fishino32 e Fishino PIRANHA sono schede di sviluppo Arduino-compatibile dotate di WiFi e lettore di schede microSD. Grazie ad un completo pacchetto di librerie (scaricabile dal sito www.fishino.it), si ha la possibilità di gestire tutti i propri progetti dal web in maniera semplice e veloce!

FISHINO PIRANHA

- Alimentazione:
 - 3,7 Vdc (tramite batteria esterna al litio)
 - da 3,5 a 20 Vdc (tramite plug)
 - da 3,5 a 20 Volt sull'ingresso Vin
 - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile al 100% con Arduino MKR1000
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Peso 10 g
- Dimensioni 62 x 25 x 16 mm



cod. PIRANHA
€ 44,⁹⁰



cod. GUPPY
€ 33,⁹⁰

FISHINO GUPPY

- Alimentazione:
 - 3,6 Vdc (tramite batteria esterna al litio)
 - da 6,5 a 20 Vdc (tramite plug)
 - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile al 100% con Arduino NANO
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Peso 10 g
- Dimensioni 75 x 20 x 18,5 mm

Prezzi IVA inclusa

**FUTURA
ELETTRONICA®**

Futura Group srl
Via Adige, 11 • 21013 Gallarate (VA)
Tel. 0331/799775

Caratteristiche tecniche
e vendita on-line su:

www.futurashop.it

FISHINO SALE A QUOTA 32!



La nostra scheda di prototipazione guadagna il processore a 32 bit e ora punta in alto, verso applicazioni impegnative che possono sfruttare a pieno il supporto WiFi, l'RTC e l'SD-Card che distinguono Fishino dalla comune Arduino UNO. Seconda e ultima puntata.

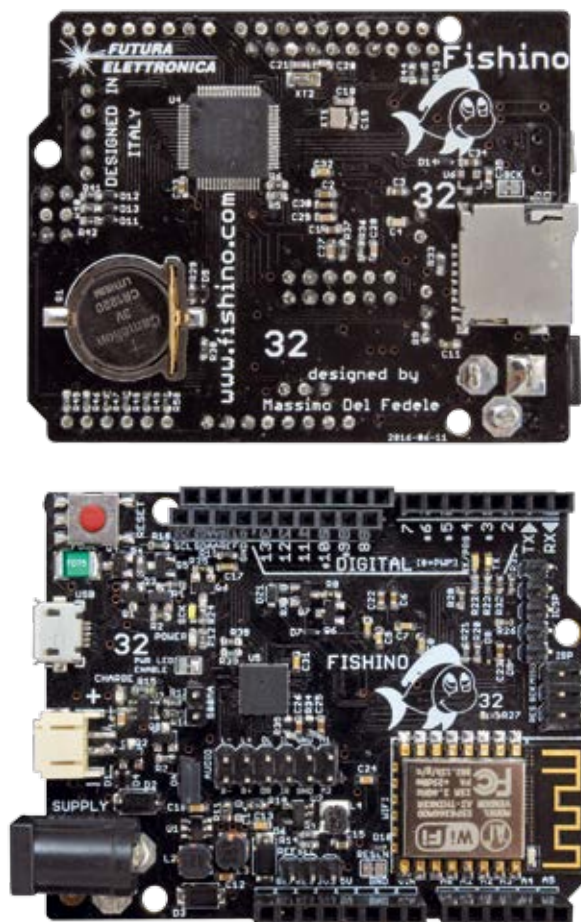
di MASSIMO DEL FEDELE

Nel fascicolo di ottobre avete conosciuto la nostra nuova e potente scheda Arduino-like, che spicca per la disponibilità on-board di transceiver WiFi ed RTC e che ora si distingue ancor più da Arduino per il processore utilizzato; infatti è stata dotata di MCU a 32 bit della Microchip. Nella scorsa puntata vi abbiamo descritto lo schema elettrico rimar-

candone le caratteristiche e i vantaggi rispetto alle nostre Fishino basate su microcontrollore ATmega; ora è giunto il momento di passare alla costruzione e alla messa in esercizio.

REALIZZAZIONE PRATICA

Fishino32 è una scheda realizzata completamente con componenti SMD, alcuni di dimensioni particolarmente



Elenco Componenti:

C1: 100 μ F ceramico (1206)	(0603)
C2: 1 μ F ceramico (0603)	C40: 100 nF ceramico (0603)
C3: 1 μ F ceramico (0603)	D1: RB521S
C4÷C8: 100 nF ceramico (0603)	D2: SS34SMA
C9: 22 μ F 25 VL ceramico (0603)	D3: SS34SMA
C10: 100 nF ceramico (0603)	D4÷D7: RB521S
C11: 4,7 μ F 25 VL ceramico (0603)	D7: SS34SMA
C12: 22 μ F ceramico (0603)	D8÷D14: RB521S
C13: 360 pF ceramico (0603)	DZ1: MM3Z3V3
C14: 10 μ F ceramico (0603)	FUSE: MF-MSMF050-2 500mA
C15: 4,7 μ F ceramico (0603)	Q1: FDN340P
C16: 100 nF ceramico (0603)	Q2: NTR4171P
C17÷C20: 22 pF ceramico (0603)	Q3: 2N7002
C21, C22: 1 μ F ceramico (0603)	Q4÷Q5: FDN340P
C23: 4,7 μ F ceramico (0603)	Q6: NTR4171P
C24÷C28: 1 μ F ceramico (0603)	Q7÷Q9: 2N7002
C29: 22 μ F ceramico (0603)	R1, R2: 100 kohm (0603)
C30÷C31: 1 μ F ceramico (0603)	R3÷R10: 10 kohm (0603)
C32: 4,7 μ F ceramico (0603)	R6÷R8: 220 kohm (0603)
C33, C34: 1 μ F ceramico (0603)	R9: 97,6 kohm (0603)
C35, C36: 4,7 μ F ceramico (0603)	R10: 13,3 kohm (0603)
C37: 1 μ F ceramico (0603)	R11: 1 kohm (0603)
C38, C39: 2,2 μ F ceramico	R12: 475 kohm (0603)
	R13: 105 kohm (0603)
	R14: 470 ohm (0603)
	R15: 10 kohm (0603)
	R16: 2,7 kohm (0603)

ridotte, e popolata su entrambi i lati.

Pur utilizzando un PCB a due facce, quindi teoricamente realizzabile con metodi artigianali, ha molte piste particolarmente sottili e moltissime vie (connessioni tra le due facce) che ne rendono la realizzazione casalinga alquanto difficile.

Il montaggio non risulta particolarmente complesso, salvo per un componente, il codec audio ALC5631Q, che, essendo in package QFN ha i piedini sotto il corpo (ma anche lungo il bordo) ed è praticamente impossibile da saldare con un comune saldatore a stilo, ma richiede la stazione ad aria calda e del flussante co-

sparso sulle piazzole, dopo aver sciolto su queste ultime un velo di stagno. L'impresa è fattibile, a patto di disporre dell'attrezzatura adatta e di un PCB realizzato in modo professionale e non casalingo.

Suggeriamo quindi ai meno esperti di acquistare la scheda già pronta all'uso, visto che a nostro avviso il tempo ed il costo dei materiali per una realizzazione casalinga non vengono compensati dal risparmio che se ne potrebbe ottenere. Sul nostro sito web www.elettronica.in mettiamo comunque a disposizione tutti i file necessari per poterlo fare, come d'abitudine; poi decidete voi.

Per tutte le fasi del montaggio, seguite i disegni nella pagina accanto.

MESSA IN FUNZIONE

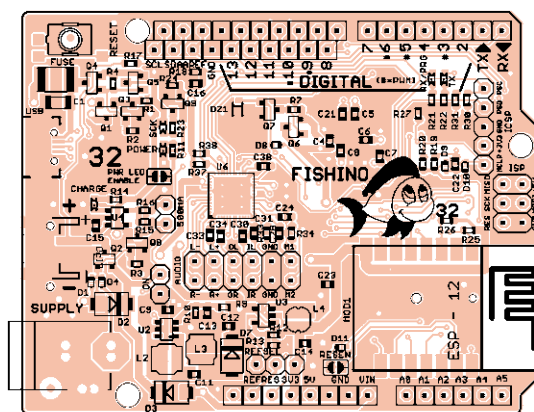
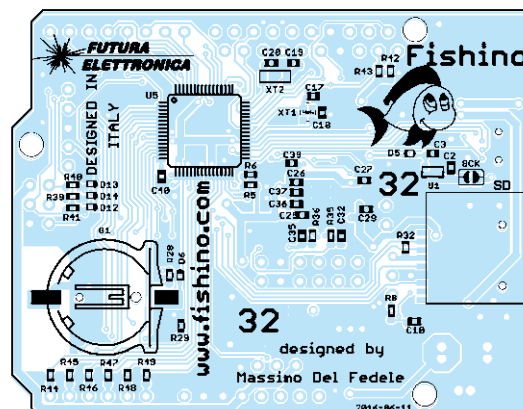
Una volta completata la vostra Fishino 32, dovete procedere alla sua programmazione, che, come abbiamo detto nella prima puntata, si può effettuare tranquillamente tramite l'IDE di Arduino, avendo l'accortezza di installare il package da noi fornito sul sito www.fishino.it contenente tutti i programmi aggiuntivi necessari (programmi di terze parti). L'installazione del package, grazie al lavoro del team di Arduino, è decisamente semplice: è sufficiente aprire l'IDE, selezio-

R17÷R18: 1 Mohm (0603)
 R19: 1 kohm (0603)
 R20: 10 kohm (0603)
 R21, R22: 330 ohm (0603)
 R23: 470 ohm (0603)
 R24: 220 kohm (0603)
 R25÷R26: 10 kohm (0603)
 R26, R28, R29: 10 kohm (0603)
 R27, R30, R31: 3,3 kohm (0603)
 R32: 10 kohm (0603)
 R33, R34: 680 ohm (0603)
 R35, R36: 2,2 kohm (0603)
 R37, R38: 10 kohm (0603)
 R39÷R41: 220 ohm (0603)
 R42, R43: 10 kohm (0603)
 R44÷R49: 220 ohm (0603)
 U1: XC6206P33
 U2: SX1308
 U3: LC3406
 U4: MCP73831
 U5: PIC32MX470F512HI/MR
 U6: ALC5631Q
 XT1: 16MHz
 XT2: 32.768 kHz
 TX: LED blu (0603)
 RX/PRG: LED giallo (0603)
 SCK: LED bianco (0603)

POWER: LED verde (0603)
 CHARGE: LED rosso (0603)
 L1: 2.2 μ H
 L2: 6.8 μ H
 L3: 6.8 μ H
 USB: Connettore micro-USB
 SD: Connettore micro-SD
 MOD1: Modulo ESP12
 con ESP8266
 LIPO: Connettore JST
 2 vie 2.54 mm
 G1: Porta batterie $\varnothing$ 12 mm
 per CR1220
 SUPPLY: Plug alimentazione
 RESET: Microswitch

Varie:

- Strip femmina 3 vie (2 pz.)
- Strip femmina 6 vie
- Strip femmina 8 vie (2 pz.)
- Strip femmina 10 vie (2 pz.)
- Strip maschio 2 vie (2 pz.)
- Strip maschio 5 vie
- Strip maschio 6 vie (2 pz.)
- Jumper (2 pz.)
- Circuito stampato S1291



nare dal menu *File* il sottomenu *Impostazioni* e, nella riga intitolata *Url aggiuntive per il Gestore schede*, inserire il seguente percorso: www.fishino.it/arduinoide/package_fishino_index.json. Come si vede nella **Fig. 1** è possibile inserire più percorsi, facendo clic sul pulsante , tramite il quale si aprirà la finestra in cui confermare o cercare nuovi percorsi. Una volta inserito il percorso di Fishino è sufficiente entrare nel menu *Strumenti* e impartire il comando *Scheda>Gestore schede...*, tramite il quale si avrà accesso al pannello di gestione delle schede aggiuntive; verso la fine troverete il pacchetto *fishino_pic32 by Fishino* (**Fig. 2**).

Sulla destra potete notare il pulsante *Installa*, insieme a una casella per selezionare il numero di versione. Al momento della stesura di questo articolo il numero di versione è 1.0.1, ma il pacchetto è in continuo aggiornamento. Fate clic sul pulsante e verrà avviato lo scaricamento dei pacchetti, che potrà durare anche parecchi minuti; non preoccupatevi, perché è un'operazione da eseguire una sola volta all'inizio (o per aggiornare la scheda, nel qual caso verranno scaricati solo gli elementi modificati). Una volta terminata l'installazione, la schermata si presenterà come in **Fig. 3**. Come si nota, il

pulsante *Installa* è stato ora sostituito da *Rimuovi* ed è apparso l'elenco delle schede disponibili; non preoccupatevi se sarà diverso da quanto appare nell'immagine: stiamo studiando ulteriori schede!

Una volta completata questa operazione, nel menu di selezione della scheda troverete la nostra Fishino32.

Adesso la vostra Fishino 32 è quasi operativa; mancano solo i driver USB, che per alcuni sistemi operativi sono da installare (non è il caso di Linux, nel quale sono inclusi, e nemmeno di MacOS X).

Per Windows occorre procedere come spiegato di seguito: i driver

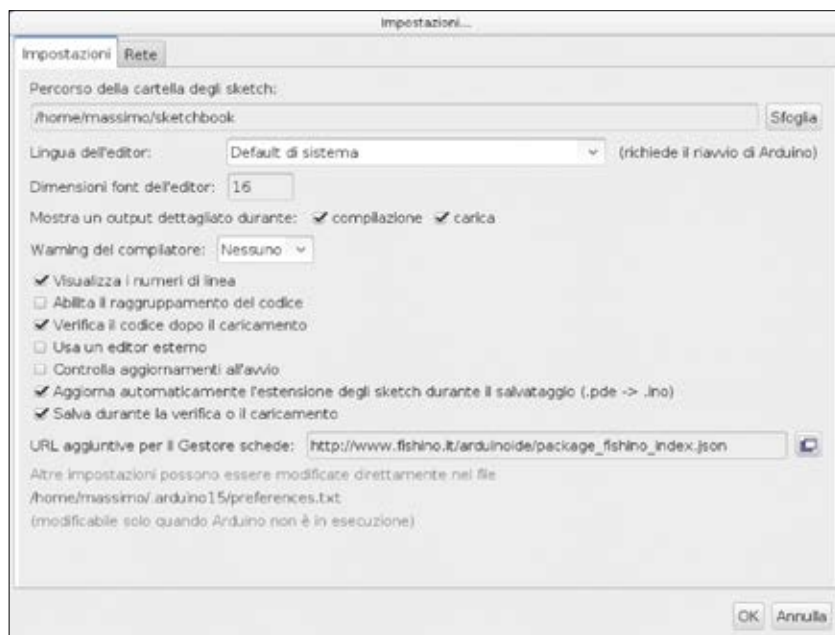


Fig. 1 - Definizione del percorso di ricerca della scheda.

venivano scaricati automaticamente insieme al pacchetto software contenente le descrizioni delle nuove schede che abbiamo appena installato; purtroppo non vengono installati automaticamente nel sistema operativo, per motivi tecnici che in questa sede sarebbe lungo esporre.

Per l'installazione è sufficiente, una volta collegato Fishino al PC, seguire le istruzioni che appariranno sullo schermo, cercando i driver manualmente nel percorso `C:\Users\nomeutente\AppData\Local\Arduino15\packages\fishino\tools\pic32-driver-windows\1.0.0\chipKIT Drivers`, dove

al posto di *nomeutente* va scritto il nome dell'utente da cui state installando i driver.

Una volta selezionato il file `Stk500v2.inf` il driver verrà installato ed apparirà la relativa porta COM virtuale nell'IDE. Una volta installati (se necessario) i driver, il sistema è pronto all'uso.

Adesso occorre selezionare la porta corretta nell'IDE, che sarà una COMxx su Windows o una TTYACMxx se lavorate con il sistema operativo Linux.

ESECUZIONE DEL PRIMO SKETCH

Eccoci pronti per il collaudo della nostra scheda! Qui troviamo subito una piccola differenza, che può sembrare una scomodità rispetto ad Arduino e alle schede Fishino ad 8 bit, ma che in realtà spesso si traduce in un vantaggio: la scheda attualmente non dispone di un autoreset che la mette automaticamente in modalità caricamento sketch; occorre quindi premere il pulsante di reset, tenendolo premuto per circa 2 secondi, finché non lampeggia il LED arancione di programmazione, ma non più di 4 secondi circa, dopodiché viene impostata la modalità di aggiornamento del firmware del modulo WiFi, condizione che sarebbe evidenziata dal lampeggio del LED blu.

Se ci è scappato il tempo, poco male: rilasciamo il tasto reset e ricominciamo.

Vi chiederete il perché di questo modo di funzionamento; ebbene, deriva dal fatto che, disponendo la scheda di un USB nativo, questo è legato al funzionamento del controller stesso, come nel caso delle schede Arduino Leonardo. Se il controller si blocca (basta uno sketch che si blocchi) la porta USB non risponde più ai comandi, nemmeno a quelli di reset. Abbiamo preferito quindi



Fig. 2 - Ricerca dei pacchetti d'installazione di terze parti.

Caricamento del bootloader

al momento un sistema manuale che eviti, ad esempio, la famigerata 'manovra di emergenza' necessaria in alcuni casi sulle schede Arduino, ovvero la pressione del tasto reset "un istante" dopo aver lanciato la programmazione dall'IDE.

Non escludiamo comunque di inserire, in una prossima versione del bootloader, un sistema di reset automatico, anche se i vincoli di cui abbiamo parlato restano. Torniamo quindi al nostro sketch; carichiamo l'esempio BLINK dall'IDE, quello che fa lampeggiare il led13 (che corrisponde al led bianco sulla scheda), teniamo premuto il tasto reset per circa 2 secondi e, quando inizia a lampeggiare il LED arancione, premiamo il pulsante di caricamento sketch dell'IDE.

Dopo pochi secondi (dipende dalla velocità del PC, dalla complessità dello sketch, dal numero di librerie utilizzate, eccetera...) il caricamento sarà completato, cosa evidenziata dallo spegnimento del led arancione e dal contemporaneo lampeggio del LED bianco.

Ecco fatto: scheda collaudata e funzionante! Ed ora?

AGGIORNIAMO IL MODULO WIFI

Se avete seguito gli articoli sulle schede Fishino ad 8 bit ricordere sicuramente che la procedura di aggiornamento firmware richiedeva alcune connessioni volanti tra gli I/O della scheda e la porta ESPCONN, il caricamento preventivo di uno sketch "neutro", ad esempio il blink, eccetera.

Nulla di tutto ciò è più necessario sul Fishino32: la scheda viene posta in modalità aggiornamento firmware utilizzando semplicemente il pulsante di reset!

La procedura è la seguente:

- premere e mantenere premuto

Se avete acquistato la scheda montata, questa è già dotata di bootloader sul controller, quindi potreste tranquillamente saltare questa parte di articolo. Diciamo "potreste" perché, come tutto il software, anche il bootloader può avere delle evoluzioni, anche se meno frequenti rispetto a librerie e firmware, quindi è sempre utile sapere come fare per riprogrammarlo.

La programmazione del bootloader diviene poi indispensabile se avete realizzato la scheda seguendo le nostre istruzioni.

Se qualcuno di voi ha già caricato il bootloader su un Arduino (o un Fishino) ad 8 bit saprà sicuramente che occorre un programmatore esterno; la cosa non si può fare attraverso la porta USB.

Per le schede ad 8 bit era sufficiente utilizzare un altro Arduino programmato per quello scopo e connesso opportunamente, seguendo facili istruzioni reperibili in rete, al connettore ISP, quello da 3+3 poli.

Per il Fishino32 le cose cambiano anche se, grazie soprattutto al lavoro della comunità, non di molto.

È sempre possibile utilizzare un Arduino (o un Fishino) ad 8 bit come programmatore, utilizzando lo schema di **Fig. A**.

Il programmatore va connesso non più alla porta ISP ma alla ICSP, ovvero al connettore con 5 pins in linea, denominati MCLR, +3V3, GND, PGD e PGC. Come si evince dallo schema, sulle porte PGD e PGC vanno inserite due resistenze di pullup connesse NON ai 5V del programmatore ma ai 3,3 Volt della scheda da programmare, che DEVE quindi essere alimentata durante la programmazione.

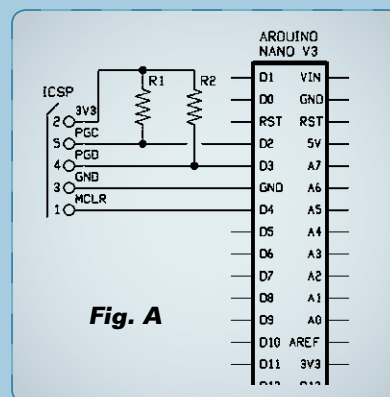


Fig. A

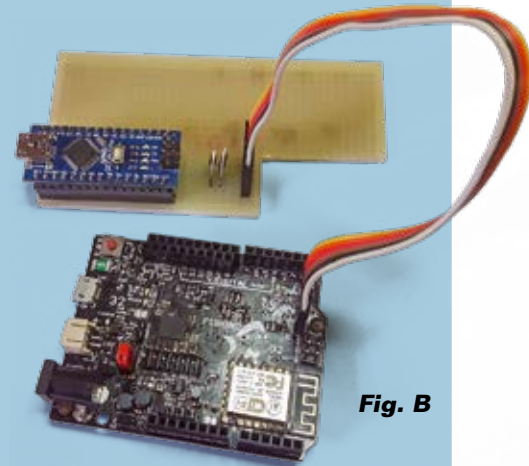


Fig. B

Il nostro programmatore sfrutta queste due resistenze come un traslatore di livello fittizio, in modo da non superare i 3.3 Volt ammissibili sulle due entrate del Fishino32; un livello basso si ottiene portando l'uscita a massa, un livello alto si ottiene lasciandola fluttuante, in modo che assuma i 3,3 Volt provenienti dal connettore stesso.

Il programmatore può essere tranquillamente realizzato su una breadboard o anche, data la semplicità, tramite fili volanti, visto che verrà usato pochissimo (**Fig. B**).

Una volta preparato e connesso il circuito, caricate l'apposito sketch sull'Arduino utilizzato come programmatore, scaricabile al seguente indirizzo:

https://github.com/sergev/pic32prog/blob/master/bitbang/ICSP_v1E.ino

Ora è sufficiente lanciare l'IDE (la quale ovviamente dev'essere dotata dei componenti aggiuntivi scaricati come da articolo), selezionare come programmatore "ARDUINO BITBANG" che appare in fondo alla lista relativa, selezionare come porta USB quella del programmatore, selezionare la scheda corretta (FISHINO32 in questo caso) e cliccare su "Scrivi il bootloader". Se è tutto a posto, l'IDE mostrerà alcuni messaggi di progresso e poi darà la conferma dell'avvenuto caricamento del bootloader.

Scollegando il programmatore vedrete quindi il led arancione lampeggiare, segno della modalità di caricamento sketch attiva, visto che qualsiasi eventuale sketch precedente sarà stato cancellato dalla procedura.

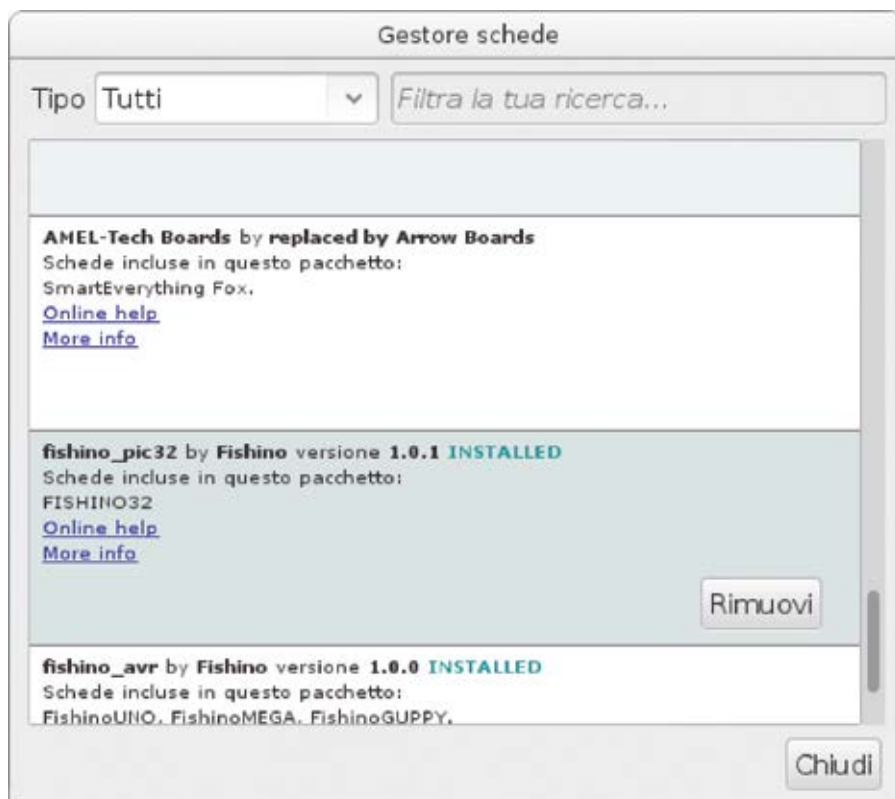


Fig. 3 - Installazione terminata.

il tasto reset finché non inizia a lampeggiare il LED blu; notate che occorrono 4 secondi, trascorsi i primi due dei quali

inizierà a lampeggiare il LED arancione, a indicare la modalità di caricamento sketch, poi (dopo altri 2 secondi) inizierà a

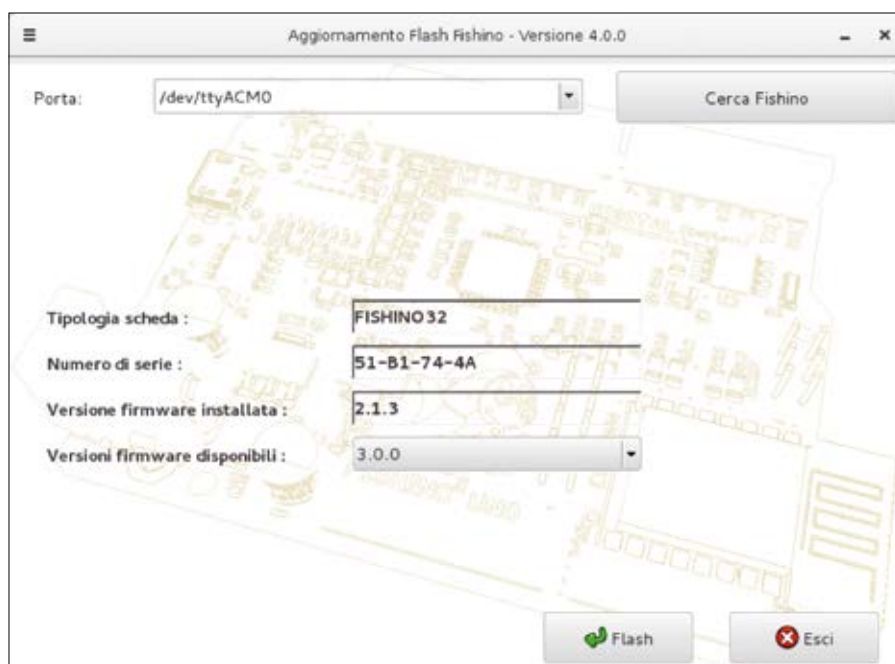


Fig. 4 - Schermata di FishinoFlasher.

lampeggiare il LED blu;

- lanciare il programma FishinoFlasher (scaricabile da <http://fishino.it/>), dopo aver verificato di avere una connessione Internet funzionante sul proprio computer, visto che il firmware verrà scaricato dalla rete wireless e quindi dal web.

Una volta avviato, il programma FishinoFlasher presenterà con la schermata di **Fig. 4**; nella schermata si possono notare alcuni elementi, tra i quali la tipologia della scheda rilevata (Fishino32), il numero di serie del modulo WiFi e la versione attualmente installata del firmware (in questo caso è la 2.1.3).

Nella casella sottostante è possibile scegliere una nuova versione da installare; viene proposta l'ultima versione disponibile (la 3.0.0, in questo caso), ma nulla vieta di installare una versione precedente, anche se normalmente non è consigliato.

Fate attenzione al fatto che ad un cambio di numero di versione "major" (la prima cifra a sinistra, che in questo caso è 3) corrisponde una non retrocompatibilità con le librerie precedenti; in pratica, una libreria installata e funzionante con la versione 2.5.5 funzionerà anche con un firmware di versione 2.5.7 e 2.9.9, per esempio, anche se non ne sfrutterà le migliorie aggiuntive, ma non è vero il contrario, ovvero una libreria fatta per la versione 2.5.7 richiederà un firmware di versione pari almeno a 2.5.7). Per contro, quella libreria non funzionerà con un firmware 3.0.0. In sintesi, una libreria creata per un firmware avente un numero di versione major X funzionerà con tutti i firmware successivi aventi lo stesso numero di versione major, ma non con i precedenti né con i successivi

aventi numero di versione major differente.

Detto ciò, scegliamo la nostra versione e facciamo clic sul pulsante Flash; dopo pochi istanti inizierà l'aggiornamento vero e proprio (Fig. 5). Al termine della procedura apparirà una finestra di notifica dell'avvenuto aggiornamento.

A questo punto è sufficiente fare clic su Ok, chiudere il programma FishinoFlasher e premere il tasto di reset sulla scheda per uscire dalla modalità aggiornamento.



LIBRERIE SOFTWARE DISPONIBILI

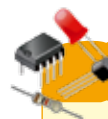
Possiamo dividere le librerie in due categorie logiche:

- librerie di sistema;
- librerie utente.

Si tratta di una divisione abbastanza fittizia, visto che bene o male tutte le librerie potrebbero essere considerate parzialmente in entrambi i settori; la vera differenza sta nel dove vengono collocate sull'hard disk e nelle modalità di aggiornamento. Le librerie di sistema sono collocate nella cartella arduino15 (che può trovarsi in posizioni differenti a seconda del sistema operativo) e vengono gestite ed aggiornate dal gestore di schede integrato nell'IDE. A questa categoria appartengono, ad esempio, la libreria SD, la AudioCodec,

la I2S, eccetera. Ad ogni aggiornamento disponibile l'IDE, ne segnalerà la presenza e darà la possibilità di scaricare automaticamente gli aggiornamenti. La libreria Fishino, per contro, insieme alla FishinoWebServer ed altre eventuali, appartiene alla seconda categoria per una nostra precisa scelta; questa libreria infatti dipende non dalla scheda in sé ma dalla versione di firmware del modulo WiFi, e va quindi aggiornata in base a questo. Inserirla tra le librerie di sistema avrebbe obbligato l'utente ad aggiornarla, e quindi ad aggiornare il firmware, ad ogni aggiornamento delle librerie di sistema, cosa non strettamente necessaria. Questa libreria va quindi scaricata direttamente dal sito www.fishino.it, sezione download, come per le precedenti schede; in quella sezione troverete anche i programmi di esempio, tra i quali il FishinoHomeAuto. Le librerie della sezione download sono fornite come files .zip da decomprimere nella cartella "libraries" presente dentro la cartella contenente gli sketch.

Una volta eseguita questa procedura, è possibile provare tutti gli esempi relativi al modulo wifi come per le precedenti schede. Una nota riguardo a questi ultimi: il loro funzionamento è identico, sulla Fishino32, a quello delle schede precedenti; noterete solo un notevole incremento delle prestazioni e, ovviamente, un'occupazione di memoria decisamente lontana dai limiti della scheda! Ora che avete tra le mani tutta la potenza di Fishino 32, siete pronti a sviluppare.



per il MATERIALE

La board Fishino32 (cod. FISHINO32) viene fornita montata e collaudata. Può essere acquistata presso Futura Elettronica al prezzo di Euro 59,00. Il prezzo si intende 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>

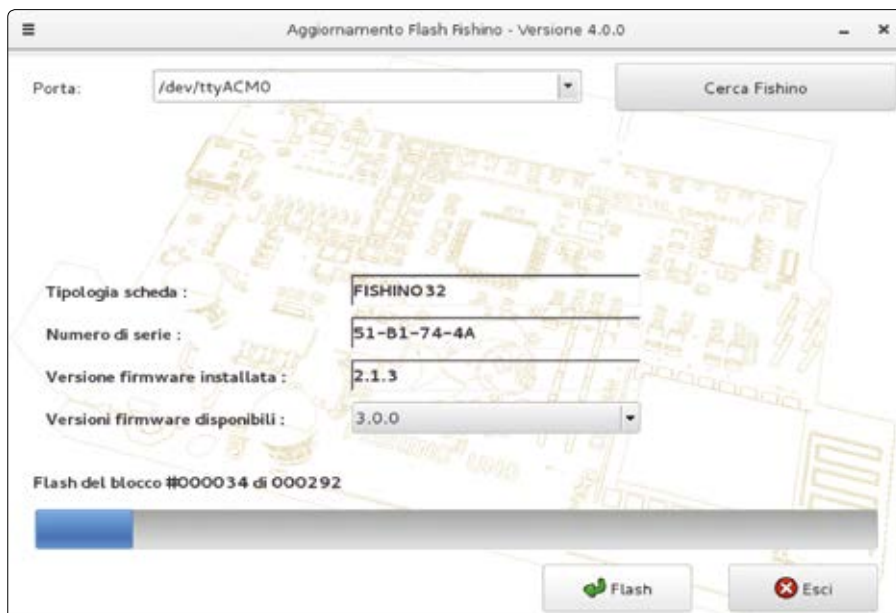


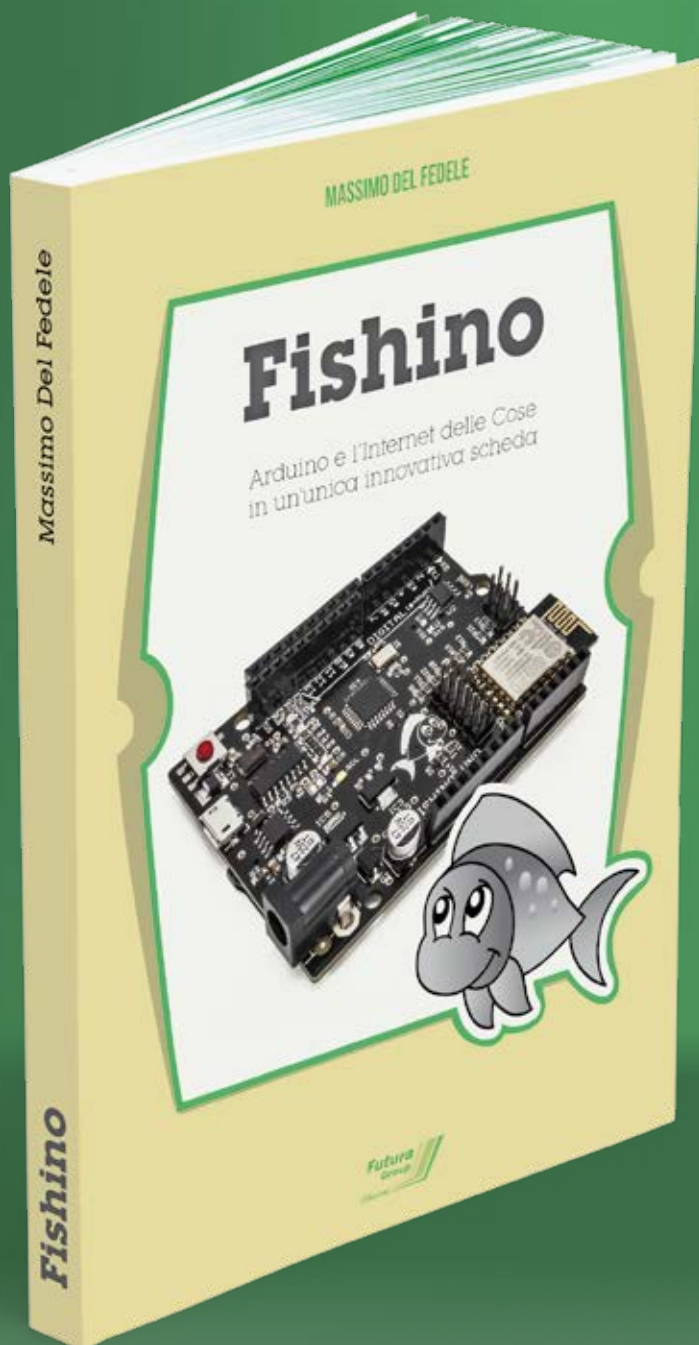
Fig. 5 - Aggiornamento del firmware.

GUIDA COMPLETA a Fishino

Arduino e l'Internet delle Cose in un'unica innovativa scheda

€ 15,00

Cod. FISHBOOK



Questo libro è rivolto sia a chi vuole muovere i primi passi nel mondo dei microcontrollori sia a chi li ha già utilizzati e desidera approfondire l'argomento.

Pur essendo focalizzato sulla famiglia di schede Fishino, dotate di connettività wireless, il libro spazia dalle conoscenze basilari di elettronica ed informatica fino ai concetti avanzati di programmazione orientata ai sistemi embedded, risultando quindi un ottimo strumento didattico anche applicato ad altre tipologie di schede. Il testo vi condurrà gradualmente in questo affascinante mondo, fino a rendervi in grado di sviluppare progetti anche complessi.

240 PAGINE

23 CAPITOLI

Acquistalo su www.futurashop.it

Futura Group srl - Via Adige, 11 - 21013 Gallarate (VA)
Tel: 0331-752668

Futura
Group
Edizioni

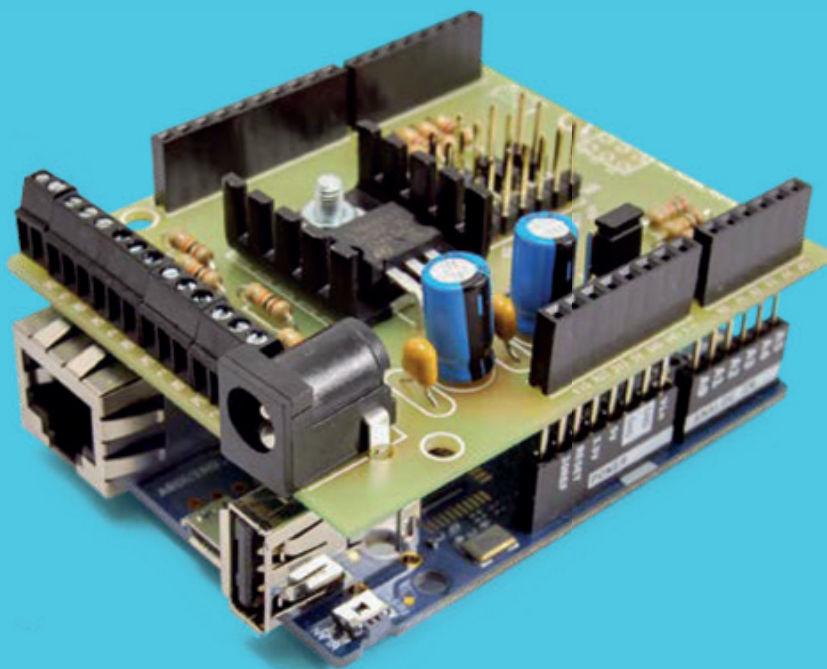
Centralina per l'irrigazione del giardino
controllata via web e smartphone
grazie ad Arduino Yún.

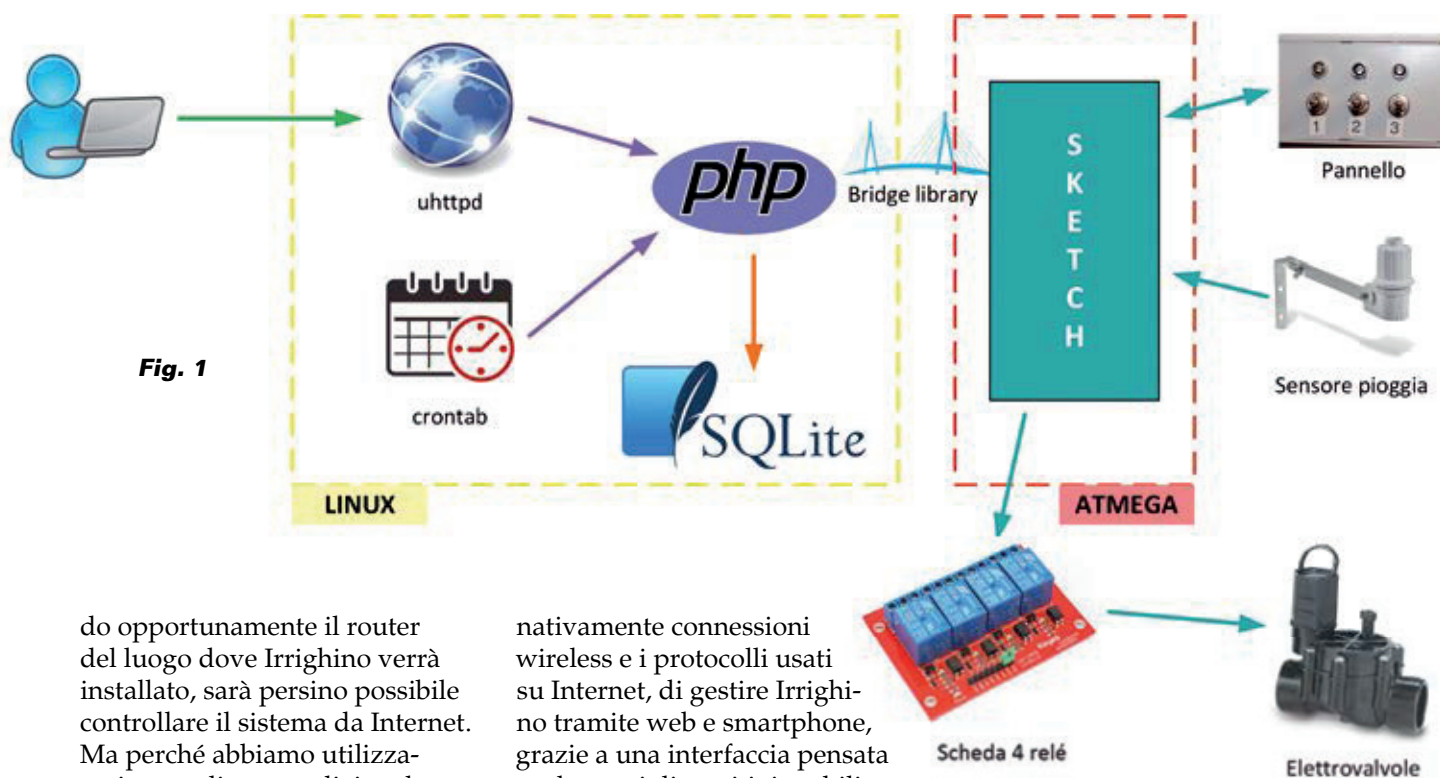
dell'ing. LUCA DENTELLA



irrlghlno

Con l'arrivo dell'estate, le nostre piante e i nostri giardini iniziano a richiedere un'irrigazione frequente e costante per poter rimanere floridi e verdi come li desideriamo. Ricordarsi di annaffiare piante e prati e riuscire ad essere a casa sempre agli stessi orari non è facile, quindi ci siamo chiesti se invece di dedicare quotidianamente del tempo per bagnare piante e fiori, non potessimo sfruttare la nostra passione per l'elettronica e realizzare un sistema automatico che assolva a tale compito. Da questa considerazione è nato il progetto che vi presentiamo in questo articolo, che consiste in una centralina di irrigazione basata su Arduino Yún e una scheda a relé interfacciabile ad essa. La centralina è in grado di gestire un numero configurabile di uscite, un sensore per la pioggia ed è programmabile tramite un'interfaccia web raggiungibile da qualsiasi PC o smartphone collegato alla rete domestica. Configurando





do opportunamente il router del luogo dove Irrighino verrà installato, sarà persino possibile controllare il sistema da Internet. Ma perché abbiamo utilizzato, invece di una tradizionale Arduino, una Yún? La risposta è che ci permette, in virtù del suo hardware che supporta

nativamente connessioni wireless e i protocolli usati su Internet, di gestire Irrighino tramite web e smartphone, grazie a una interfaccia pensata anche per i dispositivi mobili.

IL PROGETTO

Arduino Yún è il cuore della

centralina. Il progetto sfrutta appieno la caratteristica fondamentale di questo Arduino, cioè la possibilità di eseguire con-



Il colosso di Mountain View ha annunciato di essere quasi pronta a immettere sul mercato il suo Google Home: il dispositivo, a forma di cilindro con un tronco conico, può essere gestito da remoto tramite comandi vocali ed essere personalizzato con una vasta gamma di colori. Sarà in vendita entro la fine dell'anno e permetterà la gestione "smart" dell'abitazione, ma anche di chiedere e ottenere informazioni, esattamente come si fa con un motore di ricerca. L'interazione con gli

Google Home: il domestico 2.0

utenti si svilupperà grazie alla nuova App Google Assistant (evoluzione di Google Now e presto integrato nei dispositivi smart, anche indossabili e nelle automobili). Il dispositivo di Google può essere considerato un "domestico 2.0" o più semplicemente un affidabile assistente virtuale. Il dispositivo domotico, oltre a accendere le luci e a dialogare con altri dispositivi domotici e gli elettrodomestici smart, si interfacerà tramite rete locale o Internet per svolgere al meglio le attività ad esso demandate: dall'impostazione della sveglia alla lista della spesa, dal controllo del contenuto del frigorifero alla ricerca di un posto su un aereo o su un treno o di un posto al ristorante. Insomma, un vero maggiordomo 2.0.

temporaneamente una distribuzione Linux (grazie al *System on Chip* Atheros) e degli sketch (sul microcontrollore ATmega32u4 di cui è dotata). Al sistema operativo Linux è dato il compito di gestire l'interfaccia grafica, la memorizzazione della programmazione e l'attivazione delle uscite agli orari stabiliti, mentre lo sketch in esecuzione nel microcontrollore si occupa dell'interfacciamento verso l'hardware esterno: il sensore di pioggia, il pannello di comando a deviatori (opzionale) e i relé che attivano le elettrovalvole.

Nella **Fig. 1** è rappresentato lo schema logico di funzionamento: l'interfaccia web, sviluppata utilizzando librerie *javascript* secondo il paradigma AJAX (vedi riquadro), riceve dati e invia comandi ad alcune pagine scritte in linguaggio php ed eseguite dal Web Server uhttpd.

Le pagine php a loro volta comunicano con lo sketch tramite la libreria **Bridge** (che esegue il bridging tra l'hardware Arduino basato su ATmega e il SoC Atheros) mentre la programmazione e gli eventi sono memorizzati in un database SQLite. Alcuni script php sono eseguiti in maniera periodica dal **crontab** (lo schedatore) del sistema operativo Linux e si occupano di attivare la programmazione e di fare una pulizia periodica di eventi e log. Lo sketch ha il compito di gestire il pannello di controllo, di leggere lo stato del sensore pioggia e di attivare le uscite (elettrovalvole, pompe...) tramite dei relé esterni.

SCHEMA ELETTRICO

Affinché Irrighino sia in grado di gestire il nostro impianto di irrigazione, è necessario collegarlo ad alcuni elementi esterni. Per rendere più semplici tali collega-

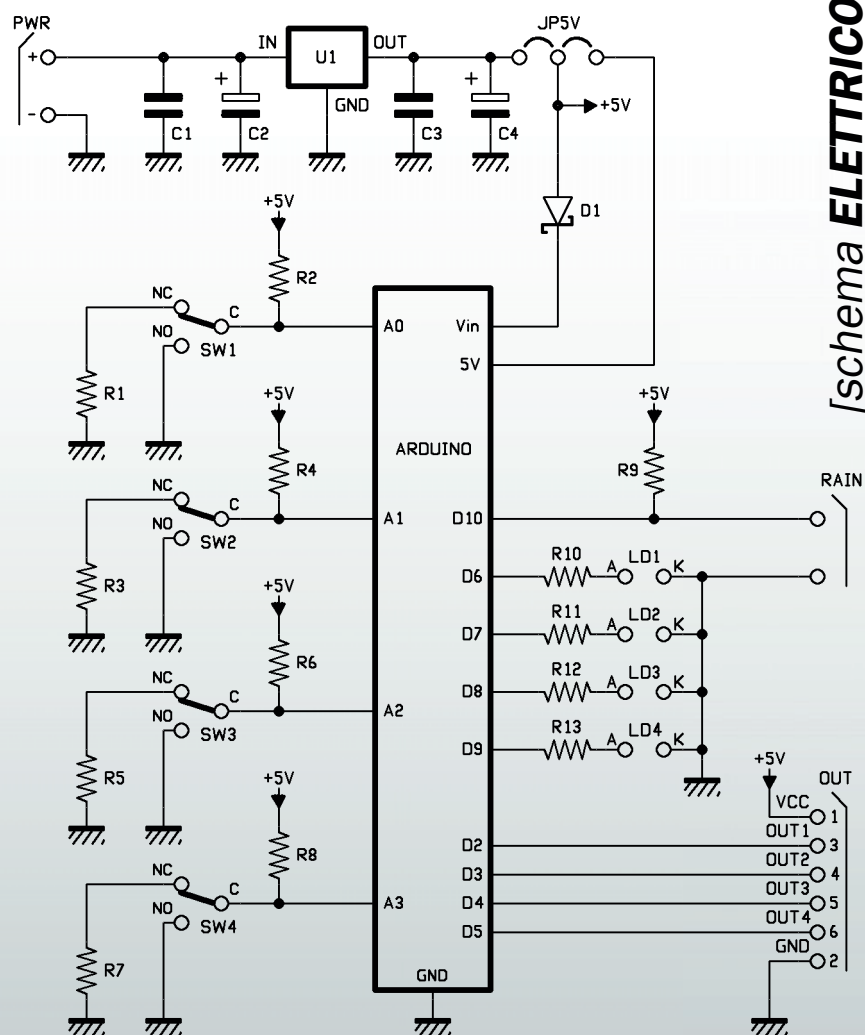
menti, abbiamo preparato uno shield che consente di gestire un impianto con un massimo di quattro elettrovalvole, un sensore di pioggia e un pannello di comando esterno con quattro interruttori e quattro LED di

stato. Lo schema di tale shield è composto da un regolatore di tensione, indispensabile ad alimentare Arduino (tramite il contatto Vin) da un certo numero di uscite per LED e uscite per la gestione delle elettrovalvole,

```

192.168.1.14 - PuTTY
root@Arduino:~# opkg update
Downloading http://downloads.arduino.cc/openwrt-yun/1/packages/Packages.gz.
Updated list of available packages in /var/opkg-lists/attitude_adjustment.
Downloading http://downloads.arduino.cc/openwrt-yun/1/packages/Packages.sig.
Signature check passed.
root@Arduino:~# opkg install php5
Installing php5 (5.4.5-3) to root...
Downloading http://downloads.arduino.cc/openwrt-yun/1/packages/php5_5.4.5-3_ar71xx.ipk.
Installing libxml2 (2.7.8-2) to root...
Downloading http://downloads.arduino.cc/openwrt-yun/1/packages/libxml2_2.7.8-2_ar71xx.ipk.
Configuring libxml2.
Configuring php5.
root@Arduino:~#
  
```

Fig. 2



[schema ELETTRICO]

Linino, OpenWrt-Yun o LininoIO

Dopo le note vicende che hanno portato alla divisione tra Arduino.cc e Arduino.org, entrambe le società hanno sviluppato e portato avanti una propria distribuzione Linux per le schede Arduino Yún.

A seconda quindi di chi è il produttore della scheda che avete a disposizione, è possibile che vi troviate con una delle due distribuzioni attuali: OpenWrt-Yún se Arduino.cc, LininoIO se Arduino.org. Fortunatamente è possibile installare entrambe su qualsiasi Arduino Yun. Abbiamo testato Irrighino su una scheda che esegue OpenWrt-Yun, se quindi avete qualche difficoltà nel provare questo progetto, potete installare tale versione seguendo le istruzioni al seguente indirizzo: <https://www.arduino.cc/en/Tutorial/YunUBootReflash>.

Sul versante Genuino, è da poco in commercio un nuovo shield ufficiale per la Yún: il Genuino Yún Shield. Questo shield consente di aggiungere le funzionalità proprie dello Yún (in particolare la possibilità di eseguire un sistema operativo Linux) a qualsiasi scheda Arduino e Genuino. Irrighino è pienamente compatibile anche con questo shield, accoppiato ad una scheda Genuino Uno o Leonardo.

nonché da quattro ingressi per i deviatori del pannello di comando, che sono opzionali.

PWR è il connettore jack di alimentazione, cui va fornita una tensione continua di valore compreso fra 8 e 15 volt, che viene filtrata dai condensatori C1 e C2 e stabilizzata a 5 volt dal regolatore integrato U1. Il piedino OUT di quest'ultimo fornisce i propri 5 V riferiti a GND al lato sinistro del ponticello JP5V, che serve a decidere la fonte di alimentazione del sistema: se vogliamo alimentare Arduino Yún attraverso lo shield bisogna chiudere il jumper tra il centrale e il contatto di sinistra (in questo caso i 5V giungono dal diodo Schottky D1), mentre se vogliamo alimentare il tutto tramite la Yún, quest'ultima va alimentata con un proprio alimentatore e il jumper va

chiuso a destra. In quest'ultimo caso non bisogna alimentare il jack PWR dello shield, perché lo shield stesso viene alimentato dal 5V di Arduino, sempre tramite D1. Indipendentemente da quale sia la fonte di alimentazione, la linea dei 5 volt alimenta anche i resistori di pull-up R2, R4, R6, R8 degli ingressi destinati a deviatori ed R9, relativa all'ingresso riservato al sensore di pioggia RAIN. Nelle prossime pagine spiegheremo l'utilizzo di quest'ultimo.

Le uscite fanno capo a OUT1, OUT2, OUT3 e OUT4 (rispettivamente D2, 3, 4, 5 di Arduino) e forniscono il livello logico alto quando devono essere aperte le elettrovalvole; sono pensate per comandare schede a relé, che accettano segnali TTL e alimentano in proprio le bobine dei relé.

PREPARAZIONE DI ARDUINO

Per poter utilizzare Irrighino, è necessario per prima cosa installare alcuni pacchetti software nel sistema operativo Linux.

Utilizzando un client SSH (se siete sotto Windows potete utilizzare PuTTY, su Linux e MacOS esiste il comando `ssh`) colleghiamoci in console ad Arduino Yún e inseriamo le credenziali di accesso (per impostazione predefinita, **root** è l'utente e **arduino** la password). Assicuriamoci inoltre che lo Yún sia connesso ad Internet.

Per prima cosa eseguiamo il comando per aggiornare la lista dei pacchetti disponibili per l'installazione, che è:

```
opkg update
```

opkg è il *package manager* della nostra distribuzione Linux; tramite questo comando è possibile installare o disinstallare applicazioni e programmi. Per installare l'interprete php (versione 5) possiamo quindi digitare il comando `opkg install php5`. Il gestore pacchetti scaricherà l'ultima versione disponibile –insieme ad eventuali prerequisiti– e la installerà nel sistema (Fig. 2).

Utilizziamo sempre **opkg** per installare tutti gli altri pacchetti necessari ad irrighino:

Fig. 3 - Cancellazione del carattere #.

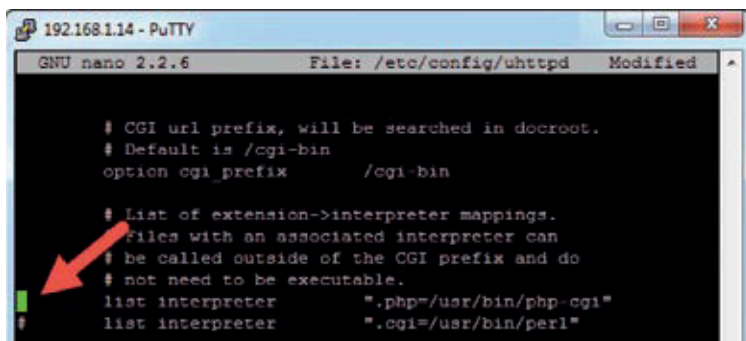
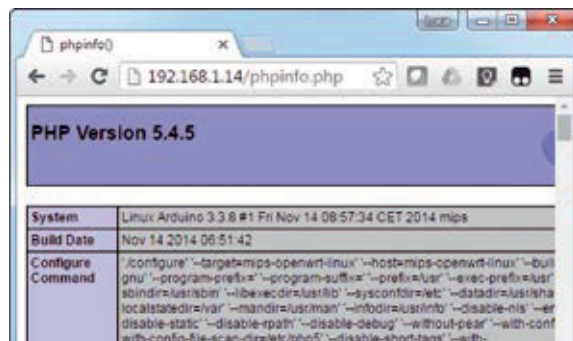
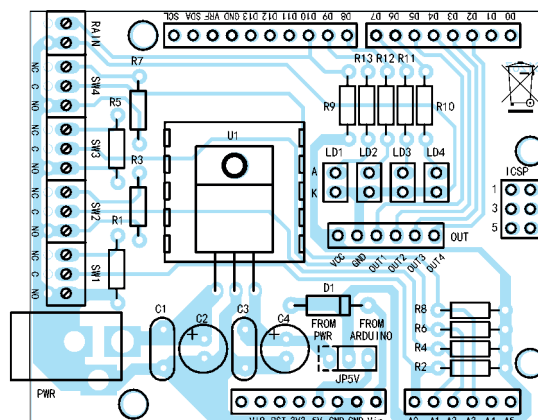
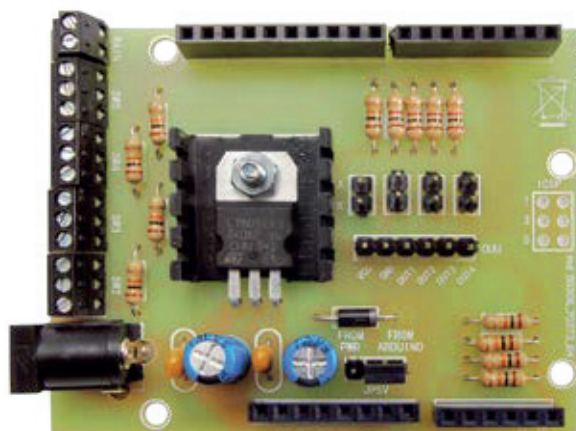


Fig. 4 - Pagina informazioni dell'interprete php.





Elenco Componenti:

C1, C3: 100 nF ceramico
C2: 100 μ F 25 VL
eletrolitico
C4: 100 μ F 25 VL
eletrolitico
R1÷R9: 10 kohm
R10÷R13: 1 kohm

U1: 7805
D1: 1N5819

Varie:
- Plug alimentazione
- Morsetto 2 vie passo
2,54mm

- Morsetto 3 vie passo
2,54mm (4 pz.)
- Vite 10 mm 3 MA
- Dado 3 MA
- Dissipatore
- Strip M/F 6 vie
- Strip M/F 8 vie (2 pz.)

- Strip M/F 10 vie
- Strip M/F 2x3 vie
- Strip maschio 3 vie
- Strip maschio 2 vie (4 pz.)
- Strip maschio 6 vie
- Jumper
- Circuito stampato S1267

```
opkg install php5-cgi php5-cli
php5-mod-curl php5-mod-json php5-
mod-pdo php5-mod-pdo-sqlite zonein-
fo-core zoneinfo-europe
```

Attiviamo ora l'interprete php all'interno del server web **uhttpd** modificando il suo file di configurazione con il comando:

```
nano /etc/config/uhttpd
```

Dobbiamo cancellare il carattere di cancelletto (#) all'inizio della riga relativa all'estensione **.php** come illustrato dalla Fig. 3. Premiamo contemporaneamente CTRL e X, quindi digitiamo Y per confermare la modifica effettuata. Infine riavviamo il server web con il comando:

```
/etc/init.d/uhttpd restart
```

Per verificare che ora il nostro Yún sia in grado di eseguire pagine php, possiamo crearne una

di esempio digitando i seguenti comandi:

```
cd /www/
echo "<?=phpinfo(); ?&gt;" &gt;phpinfo.php</pre

```

Se ora apriamo un browser sul nostro computer e digitiamo nella barra degli indirizzi l'indirizzo IP di Arduino Yún seguito da **/phpinfo.php**, dovremmo vedere la pagina di informazioni dell'interprete php come in Fig. 4.

INSTALLAZIONE

Scarichiamo ora l'ultima versione del software dal repository Github. Collegiamoci all'indirizzo <https://github.com/lucadentella/irrhino> e clicchiamo sul pulsante **Download ZIP**: dopo qualche istante avremo sul nostro PC un file compresso che contiene tutto il necessario per installare Irrhino.

Prendiamo una scheda micro SD e inseriamola nel nostro PC; dobbiamo creare in essa una

AJAX

AJAX (Asynchronous JavaScript and XML) è una tecnica di sviluppo di siti e applicazioni web pensata per renderli più interattivi.

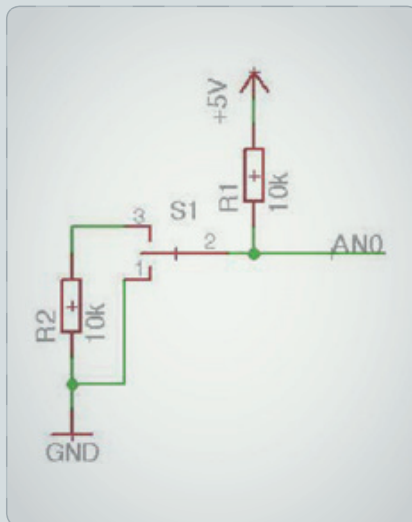
Prima dell'introduzione di tale tecnica, ad ogni azione dell'utente (ad esempio il click su un link o un pulsante) veniva ricaricata l'intera pagina del sito.

Ora invece, grazie alla possibilità di implementare chiamate asincrone in **background**, è possibile inviare o ricevere dati dal server web e aggiornare solo una porzione della pagina, rendendo l'esperienza utente molto più ricca e simile all'uso di applicazioni tradizionali.

L'interfaccia grafica di Irrhino fa ampio uso di chiamate asincrone; ad esempio all'inserimento di una nuova schedulazione corrisponde una chiamata in background che invia la schedulazione al server per memorizzarla nel database SQLite.

Tre posizioni, un solo pin

Per leggere lo stato di un interruttore a tre posizioni, è possibile usare un solo pin analogico di Arduino e due resistenze, collegandolo come mostrato nell'immagine in questo riquadro. Quando l'interruttore è in posizione centrale, il pin è collegato a 5V tramite R1 e il valore letto dall'istruzione **analogRead()** è vicino al massimo (1023). Quando l'interruttore è in posizione 3, il pin è collegato ad un partitore resistivo formato da R1 e R2; il valore letto sarà circa la metà (512). Infine quando l'interruttore è in posizione 1, il pin è collegato direttamente a massa e il valore letto è vicino a zero. Tale tecnica viene utilizzata nello shield utilizzato in questo progetto.



cartella chiamata **arduino** e, all'interno di quest'ultima, una cartella chiamata **www**. Copiamo ora la cartella **Website**, contenuta nell'archivio scaricato da Github, all'interno della cartella **www**; infine rinominiamo la cartella **Website** in **irrighino**. Se tutti i passaggi sono stati effettuati correttamente, il contenuto della scheda micro SD sarà quello di **Fig. 5** e potremo inserire la scheda di memoria nell'apposito slot del nostro Arduino. Sempre collegati ad Arduino via SSH, digitiamo il comando

`crontab -e` per modificare la configurazione dello schedulatore e, dopo aver premuto il tasto "i", inseriamo tre righe, come indicato in **Fig. 6**. Per salvare la configurazione premiamo ESC, quindi digitiamo **:wq** e premiamo INVIO. Le tre righe inserite attivano ogni minuto (*****) lo script che controlla se vi sono schedulazioni da attivare e una volta al giorno (alle 00:05 e 00:10) gli script che puliscono gli eventi e i log. Adesso ci resta solo un ultimo passo da fare: caricare lo sketch,

contenuto nella cartella **Sketch** all'interno dell'archivio di Github. Utilizzando quindi l'IDE di Arduino, apriamo lo sketch *irrighino.ino* ed effettuiamo l'upload sul nostro Yún.

COLLEGAMENTI

Per comandare le elettrovalvole è necessario utilizzare dei relé esterni funzionanti a 5V e comandati dai pin di output di Arduino. Per questo scopo, lo shield prevede il connettore OUT, che riporta positivo d'alimentazione 5V, massa e quattro pin digitali di Arduino per l'attivazione di altrettanti relé. Il connettore è pensato per pilotare il modulo **4 relé 5vdc** commercializzato da Futura Elettronica (www.futura-shop.it). Tale scheda è dotata di 4 relé con bobina a 5 V e scambio che consente di commutare carichi elettrici che funzionano con un massimo di 240Vca e assorbono fino a 10A (max); ciascun relé viene attivato portando l'ingresso di comando a livello logico alto (da 1,6 a 5 Vcc), quindi la scheda è ideale per attuare i segnali logici forniti dallo shield, anche perché ha un assorbimento, su ciascun input, di pochi milliamperere, compatibile anche con le logiche CMOS e CMOS HC. Gli ingressi della scheda 4 relé sono tutti optoisolati, così che laddove sia necessario è possibile separare galvanicamente la scheda dal circuito di comando; ogni optoisolatore ha l'anodo del diodo emettitore di ingresso (pin 1) collegato al rispettivo input tramite un resistore di limitazione della corrente e il catodo in comune con i catodi dei LED degli altri fotoaccoppiatori e connesso al contatto COM di ingresso. Se si desidera rinunciare all'isolamento, si devono ponticellare il contatto GND e il COM.

Per ciascun relé è previsto un

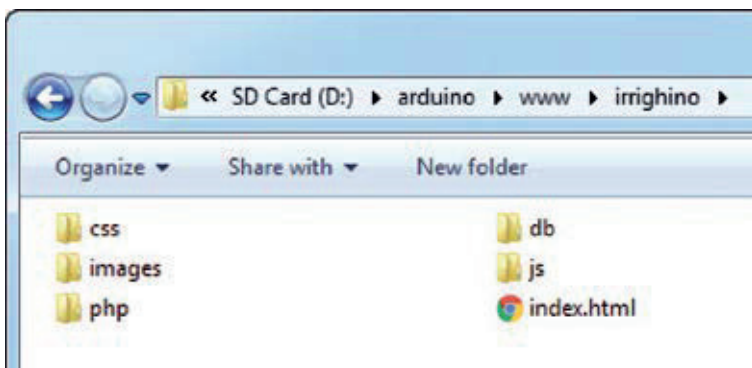


Fig. 5

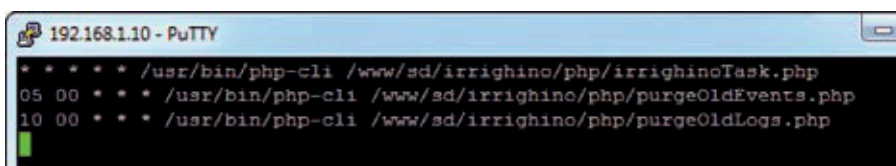
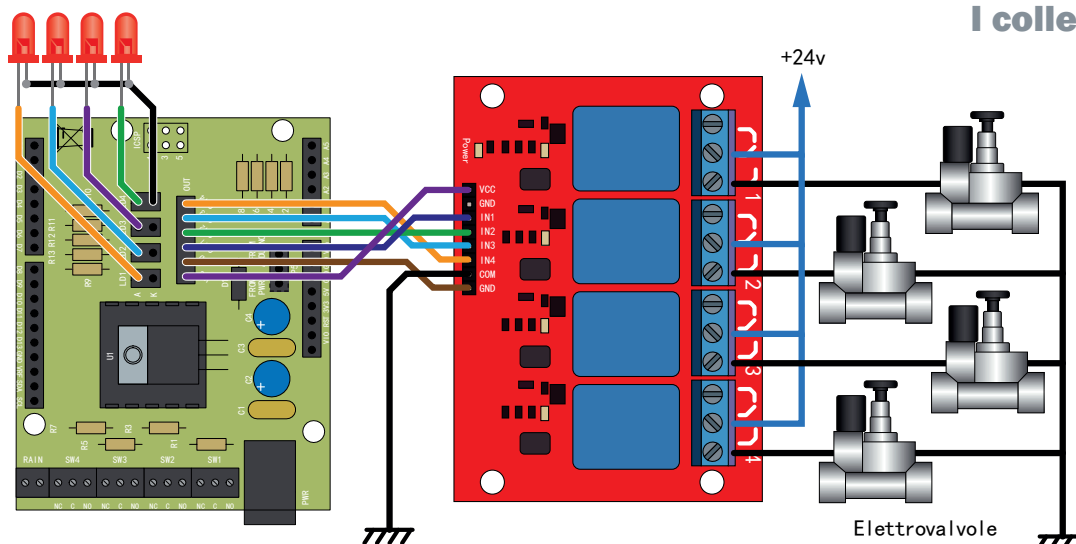


Fig. 6



LED di stato posto sul bordo della scheda.

Le elettrovalvole comunemente utilizzate per gli impianti di irrigazione dei giardini funzionano a 24V; sarà quindi necessario un trasformatore da rete con secondario che fornisca tale tensione e la corrente che serve ad alimentare il complesso delle elettrovalvole stesse. I capi del secondario saranno collegati uno direttamente al comune delle elettrovalvole e uno al polo comune dei relé, come indicato nel disegno in questa pagina (**I collegamenti**). Se disponete di una sola elettrovalvola, il collegamento è quello visibile in **Fig. 7**, dove trovate l'esempio di collegamento valido per un'elettrovalvola pilotata da una scheda relé singola.

Se le elettrovalvole sono interrate in pozzetti, per il loro collegamento elettrico utilizzate sempre connettori o scatole di derivazione a tenuta stagna (IP65). Il sensore di pioggia è solitamente composto da contatti interdigitati che vengono posti a terra e che quando piove, l'acqua unisce determinando un forte abbassamento della resistenza; collegato all'ingresso RAIN, in condizioni di asciutto lascia che il pin D10 di Arduino assuma l'1 logico, mentre quando è bagnato, la resistenza dell'acqua fa partitore con R9 e

Sopra, lo schema di cablaggio dello shield con la scheda a 4 relé che utilizziamo per comandare fino a quattro elettrovalvole; lo schema è valido a prescindere dalla tensione di funzionamento delle elettrovalvole, che può essere da 12 a 220 Vca.

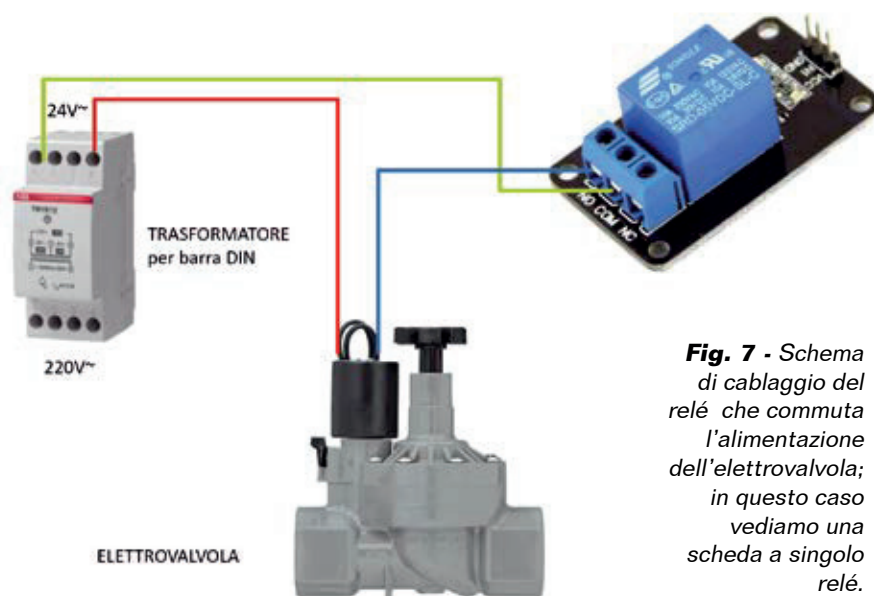


Fig. 7 - Schema di cablaggio del relé che commuta l'alimentazione dell'elettrovalvola; in questo caso vediamo una scheda a singolo relé.

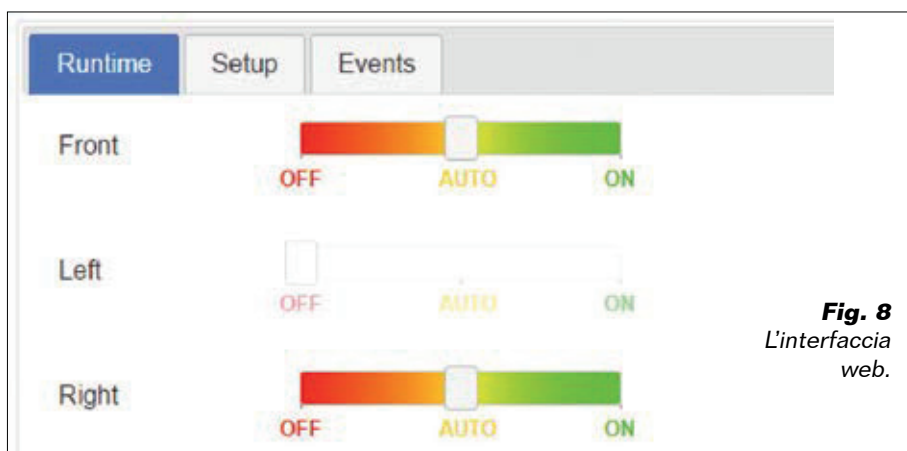


Fig. 8 L'interfaccia web.

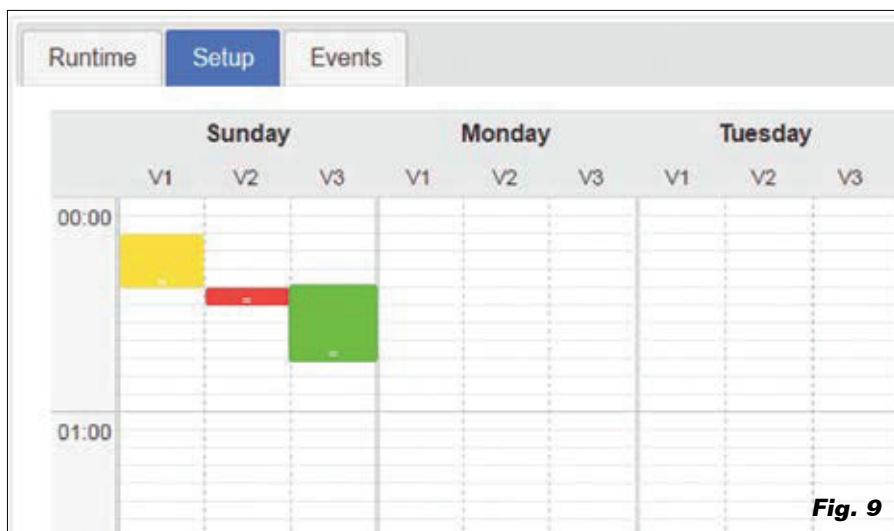


Fig. 9

```

30 // ----- Output configuration -----
31
32 define ("OUTPUTS_NUMBER", 2);
33
34 $outputs = array();
35 $outputs[0]["name"] = "Giardino";
36 $outputs[0]["baseColor"] = "#FFD800";
37 $outputs[0]["borderColor"] = "#E5BF00";
38
39 $outputs[1]["name"] = "Fiori balcone";
40 $outputs[1]["baseColor"] = "#FF0000";
41 $outputs[1]["borderColor"] = "#E00000";
42

```

Fig. 10

```

// outputs, digital PINs connected to the relays
#define OUT_1      2
#define OUT_2      3
#define OUT_3      4

// leds, digital PINs that directly drive the panel leds
#define LED_1      7
#define LED_2      6
#define LED_3      5

// switches, analog PINs that read the 3-way panel switches
#define SWITCH_1    0
#define SWITCH_2    1
#define SWITCH_3    2

// rain sensor, digital PIN connected to the rain sensor
// (basically a 2-way switch, normally closed)
#define RAIN_SENSOR 8

```

Fig. 11

D10 presenta una tensione tanto bassa da corrispondere allo zero logico.

Alcuni sensori funzionano, invece, grazie a un materiale poroso che quando viene bagnato si espande, spingendo la levetta di un microswitch e asciugandosi rilascia tale levetta; questi sono molto precisi e regolabili, perché con una vite si può allontanare o distanziare il microswitch in modo che venga premuto solo con una certa espansione del materiale poroso e quindi con una determinata quantità di acqua assorbita. Il sensore può, in ogni caso, essere collegato direttamente ad un pin digitale di Arduino tramite una resistenza di *pull-up*; se avete scelto di utilizzare lo shield proposto in questo articolo vi basta collegarlo alla morsettiera etichettata come RAIN.

Infine è possibile dotare Irrighino di un pannello di controllo fisico, per operare manualmente in locale senza bisogno di un computer: detto pannello prevede un interruttore a tre posizioni (ON, AUTO, OFF) stabili e un LED di stato per ogni uscita. La posizione dell'interruttore viene letta tramite un pin analogico di Arduino Yún, come spiegato nell'apposito riquadro, grazie al particolare collegamento di due resistenze che permettono di ottenere un valore di tensione per ciascuna posizione.

Ogni interruttore del pannello serve, dunque, a forzare manualmente una condizione all'uscita corrispondente e quindi nell'elettrovalvola corrispondente; la condizione assunta viene segnalata dal relativo LED. SW1 si riferisce e OUT1, il cui stato è segnalato da LD1, SW2 a OUT e LD2 e via di seguito.

Portando l'interruttore sulla posizione di ON, la relativa uscita viene attivata, mentre portandolo

sulla posizione di OFF l'uscita viene disattivata. Lasciandolo nella posizione intermedia (AUTO) l'uscita segue la programmazione impostata. Il pannello ha priorità rispetto all'interfaccia grafica, quindi impostando sul pannello un'uscita ad OFF essa risulterà *disabilitata* sull'interfaccia grafica e non sarà possibile modificarne lo stato.

Lo shield prevede già le morsettiere cui collegare i quattro interruttori e i pin cui connettere i LED: per questi ultimi dovete fare attenzione a collegare correttamente l'anodo e il catodo ai pin come indicato dalla serigrafia (l'anodo corrisponde normalmente al terminale più lungo).

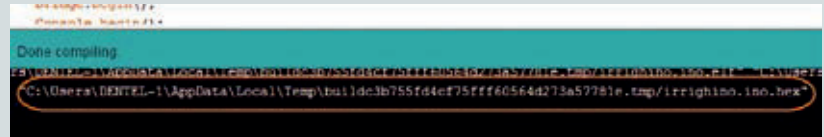
Lo shield è anche dotato di un regolatore di tensione (5V) che consente di alimentare tutti i componenti esterni in maniera autonoma rispetto ad Arduino. Tramite il jumper **JP5V** è possibile scegliere se utilizzare l'alimentazione proveniente da Arduino o un alimentatore esterno (9-12V corrente continua) collegato direttamente allo shield.

UTILIZZO DI IRRIGHINO

Vediamo adesso come si utilizza la nostra centralina di irrigazione nella modalità remota: da un PC aprite il browser Internet e nella barra degli indirizzi puntate l'indirizzo web http://ip_arduino_yun/sd/irrighino; dalla pagina che si aprirà potrete accedere alla schermata di gestione della nostra centralina.

L'interfaccia è divisa in tre tab: **Runtime**, da dove è possibile comandare manualmente le uscite, **Setup**, dalla quale è possibile programmare le attivazioni su base settimanale, ed **Events**, dove viene visualizzato l'elenco degli eventi (attivazioni, modifiche alle configurazioni...) più recenti. Ogni uscita può essere impostata

A volte capita che l'IDE di Arduino non sia in grado di rilevare in rete lo Yún; in tal caso possiamo caricare lo sketch collegando la scheda al PC tramite USB; oppure è possibile sfruttare la funzionalità di *upload via web*. Per prima cosa dobbiamo ottenere la versione compilata dello sketch: dal menu dell'IDE scegliamo **File > Preferences** e mettiamo una spunta a *Show verbose output during compilation*. Lanciamo quindi la compilazione dello sketch (pulsante **Verify**) e, una volta completato il processo, troviamo il percorso del file compilato in una delle ultime righe della finestra di log:

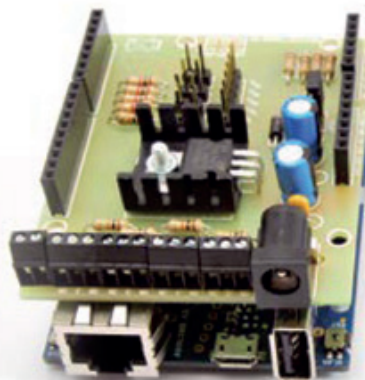


Collegiamoci tramite un browser ad Arduino Yún e, dopo aver inserito la password (**arduino** è la predefinita) carichiamo lo sketch tramite l'apposita funzione:



in tre stati: **ON** - l'uscita viene attivata; **OFF** - l'uscita viene disattivata o **AUTO** - l'uscita segue la programmazione impostata. Se l'uscita viene configurata tramite il pannello esterno, i relativi comandi sull'interfaccia web saranno disattivati: il pannello ha infatti priorità maggiore (vedi Fig. 8).

La fase di programmazione è molto semplice: gli eventi sono raccolti in una tabella nelle cui colonne sono rappresentati i giorni della settimana e le singole uscite (V1...). Ogni riga rappresenta un intervallo di 5 minuti: cliccando con il mouse su una cella è possibile definire un evento di attiva-



L'elettronica componente Irrighino: a sinistra Arduino Yún con su lo shield d'interfaccia con deviatori, LED e scheda a relé; a destra la scheda 4 relé da noi utilizzata per comandare le elettrovalvole.

Listato 1 – Il file config.h

```
// debug output (enable/disable)
#define DEBUG_MODE
#ifdef DEBUG_MODE
    #define DEBUG_PRINT(x)    Console.print(x)
    #define DEBUG_PRINTLN(x) Console.println(x)
#else
    #define DEBUG_PRINT(x)
    #define DEBUG_PRINTLN(x)
#endif

// outputs, digital PINs connected to the relays
#define OUT_1      2
#define OUT_2      3
#define OUT_3      4
#define OUT_4      5

// leds, digital PINs that directly drive the panel leds
#define LED_1      6
#define LED_2      7
#define LED_3      8
#define LED_4      9

// switches, analog PINs that read the 3-way panel switches
#define SWITCH_1    0
#define SWITCH_2    1
#define SWITCH_3    2
#define SWITCH_4    3

// rain sensor, digital PIN connected to the rain sensor
// (basically a 2-way switch, normally closed)
#define RAIN_SENSOR 10

// constants for switch positions
#define SWITCH_MANUAL_OFF 0
#define SWITCH_AUTO      1
#define SWITCH_MANUAL_ON  2

// constants for rain sensor status
#define NOT_RAINING       0
#define RAINING           1
```

zione. Tramite il mouse, l'evento può essere ridimensionato (per modificarne la durata) e spostato, anche su un'altra uscita. Cliccando su un evento, questo viene selezionato e attorno ad esso viene visualizzato un bordo più scuro (vedi Fig. 9) che ne evidenzia la selezione.

È possibile selezionare più eventi o deselectarli facendo nuovamente clic su di essi mentre si preme il tasto Shift.

Infine premendo il tasto CANC è possibile eliminare tutti gli eventi selezionati.

PERSONALIZZAZIONE DI IRRIGHINO

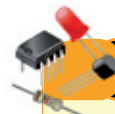
Sebbene il progetto che potete scaricare da Github sia configurato per funzionare *out of the box* insieme allo shield che abbiamo

realizzato, Irrighino è stato pensato per essere facilmente personalizzabile in modo da poterlo adattare ad ogni necessità. Aprendo il file **include.php**, presente all'interno della cartella php, è possibile modificare il numero di uscite, il loro nome e il colore assegnato, come illustrato in Fig. 10. È anche possibile modificare il comportamento in caso di pioggia (stop immediato delle schedulazioni in corso e/o nessuna nuova attivazione) o disabilitarlo completamente nel caso non sia previsto nel vostro impianto. Anche i collegamenti tra Arduino e i dispositivi esterni sono configurabili modificando lo sketch. Tutto il *mapping* tra i dispositivi e i pin cui sono collegati è stato definito in un apposito file, chiamato **config.h**, che riportiamo

nel **Listato 1**. È quindi possibile modificare i pin cui sono collegati relé, interruttori, LED e sensore di pioggia semplicemente indicando, sempre nel file **config.h**, i pin di Arduino desiderati (vedere Fig. 11). Ricordate, nel caso applichiate ad Arduino altri shield oltre quello di Irrighino, di verificare sempre -prima di assegnare gli I/O in **config.h**- che non siano già utilizzati dalle rispettive applicazioni.

Accertatevi sempre che il numero di uscite configurato nell'interfaccia web corrisponda al numero di output (OUT_x) definiti nello sketch e ricordatevi anche di aggiornare lo sketch principale. Infine, se non prevedete l'installazione di un pannello di controllo esterno è possibile inibire le funzioni associate al controllo manuale mettendo a **false** la costante **USE_SWITCHES**.

Bene, detto ciò abbiamo concluso e non possiamo che augurarvi buon lavoro! ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. La board Arduino Yún (cod. ARDUINOYUN) è in vendita a Euro 75,00. Lo shield Irrighino (cod. FT1267K) costa Euro 13,00; il modulo a 4 relé da 5V DC/10A (cod. RELAY4CH) è disponibile a Euro 13,00. Questi ultimi componenti si possono acquistare presso Futura Elettronica. Il firmware utilizzato per programmare Arduino Yún può essere scaricato dal sito della rivista.

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

FISH'N TWEET

Controlliamo le uscite di uno shield di espansione Arduino montato sulla nostra Fishino Uno, tramite Twitter.

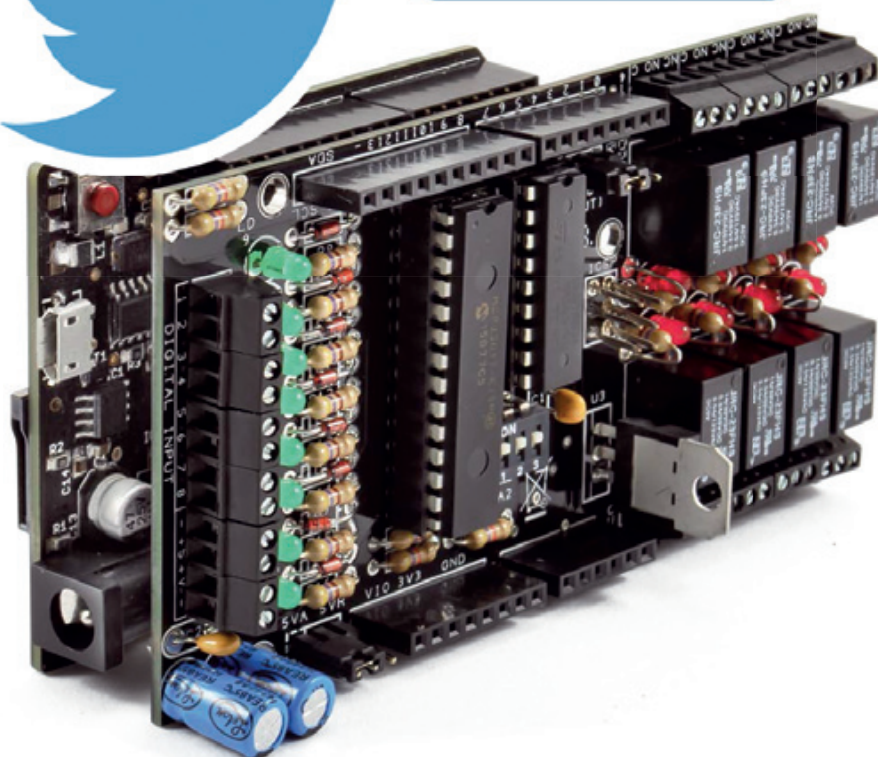


Ei Elettronica In
@elettronica_in

#fishino 24 ON

Reply Elimina Favorite Altro

21 ottobre 2015 12:36



di MASSIMO DEL FEDELE

Avete conosciuto Fishino, il nostro Arduino dotato di connettività WiFi ed RTC, nel fascicolo n° 199; ora è venuto il momento di vedere un'applicazione pratica della scheda, che permette, in modo molto economico e semplice, la gestione remota di apparecchiature elettriche quali lampadine, caldaie, condizionatori, antifurto, eccetera. L'unicità del sistema è costituita dal fatto di non richiedere apparecchiature di rete o server remoti esterni, ma solo una scheda Fishino UNO ed eventualmente uno o più shield per la gestione dei carichi applicati, che nel nostro caso

Fig. 1

Fig. 2



sono I/O Expander basati su MCP23017 (descritti nel fascicolo n° 181, cui rimandiamo per gli approfondimenti del caso). Questo progetto, non realizzabile con un semplice Arduino nemmeno se dotato di shield WiFi o Ethernet Shield, è stato reso possibile grazie alla capacità di Fishino di effettuare connessioni sicure con il protocollo https, disponibili solitamente su apparecchi di livello superiore e decisamente più costosi; restare in tema Arduino, attualmente per realizzare un progetto equivalente occorrono una YÙN e un software decisamente più complesso. Prima di entrare nel merito del progetto vediamo i vari modi per controllare apparecchiature a distanza.

Controllo via SMS

Vantaggi:

- il ricevitore può essere reso portatile ed indipendente da infrastrutture di rete;
- risposta pressoché istantanea;
- gestibile da qualsiasi cellulare, anche non recente;
- portata limitata solo dalla copertura radiomobile.

Svantaggi:

- richiede una SIM sull'apparecchio controllato, che solitamente scade se non usata;
- richiede un dispositivo GSM, non economicissimo, per la ricezione;
- richiede una copertura GSM, cosa non sempre scontata.

Controllo via moduli radio

Vantaggi:

- il ricevitore può essere reso portatile e indipendente da infrastrutture di rete;
- risposta pressoché istantanea;
- molto economico.

Svantaggi:

- portata molto limitata e fortemente influenzata dalla presenza di eventuali ostacoli.

Controllo via Internet

Vantaggi:

- può essere reso portatile al costo di un ricevitore GPRS;
- può essere effettuato pressoché da qualsiasi PC o telefono cellulare;
- molto economico, se presso la destinazione è presente un'infrastruttura di rete;
- può diventare bidirezionale senza costi aggiuntivi e senza particolare difficoltà.

Svantaggi:

- richiede la presenza di un'infrastruttura di rete in ricezione o un costo aggiuntivo per il GPRS;
- la stabilità dipende dalla connessione Internet;
- i tempi di accesso possono essere variabili a seconda della connessione e/o del metodo utilizzato;
- spesso richiede ricevitori complessi e/o un computer di appoggio;
- richiede un IP pubblico sull'apparecchio ricevente o l'uso di un servizio intermedio di terze parti;
- possono esserci problemi di sicurezza che richiedono accorgimenti particolari per evitare utilizzi non autorizzati.

Con questo progetto abbiamo cercato di unire i vantaggi del controllo via Internet, eliminando alcuni degli svantaggi di tale tecnologia, sfruttando un servizio ben noto a tutti: Twitter. In pratica, i dispositivi vengono azionati inviando un semplice tweet con un formato particolare, dall'utente preimpostato (o più utenti, con qualche piccola modifica al software). Con questo sistema risolviamo le seguenti problematiche del controllo via Internet:

- la stabilità della connessione; nel caso la connessione sia momentaneamente assente, il comando viene solo posticipato ed eseguito al suo ripristino;
- il ricevitore risulta largamente semplificato, dal lato sia hardware che software;
- non è richiesto alcun IP pubblico né un servizio di rete intermedio;
- la sicurezza è garantita dall'infrastruttura di Twitter, che lavora con connessioni sicure.

Come per tutte le cose, anche questo sistema non è esente da svantaggi, che sono principalmente la non immediatezza dell'attuazione (il tempo di propagazione di un tweet è di qualche secondo fino ad un paio di decine di secondi) ed alcune limitazioni (aggirabili) sull'invio di tweet duplicati.

Vediamo innanzitutto l'utilizzo, per poi passare alla descrizione del sistema.

Il controllo avviene tramite l'invio di un Tweet con il seguente formato:

#hashtag porta ON/OFF

ove '**#hashtag**' è un hashtag impostabile nel software (nel nostro caso abbiamo preimpostato **#fishino**), '**porta**' è un numero di porta da attivare (vedremo i dettagli in seguito) e '**ON**' o '**OFF**' sono i comandi di accensione/spengimento della medesima. Esempio:

#fishino 24 ON accende l'uscita 24
#fishino 10 OFF spegne l'uscita 10

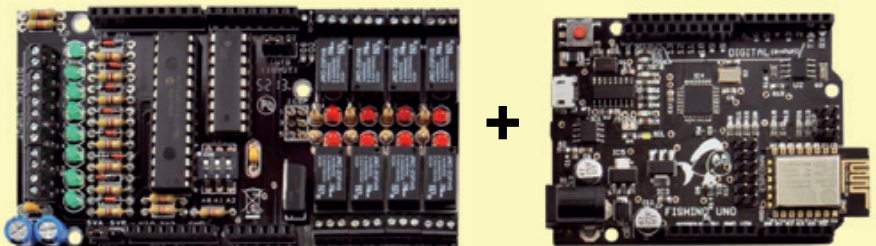
Come detto sopra, tra l'invio del tweet e l'esecuzione del comando passa un tempo stimabile in 5-20 secondi, dipendente dalla congestione della rete di Twitter. L'altro svantaggio di cui parliamo sopra è l'impossibilità di

Il progetto si propone di controllare utilizzatori elettrici o gestibili elettricamente tramite l'invio di tweet contenenti le istruzioni per governare i relé di schede di I/O applicabili a Fishino: si tratta di shield FT1079 della Futura Elettronica, sviluppati intorno all'I/O Expander MCP23017 della Microchip. Questo integrato dispone in totale di 16 pin (suddivisi a livello logico in 2 diverse porte A e B) configurabili individualmente come ingresso o uscita. Nello shield gli 8 pin della porta A sono impostati come uscite e gli 8 pin della porta B come ingressi. Le uscite pilotano gli ingressi di un line-driver ULN2803, progettato appositamente per pilotare carichi induttivi come le bobine dei relé (infatti al suo interno dispone di diversi stadi a transistor dotati di diodi di protezione della giunzione base-collettore). Gli ingressi sono invece collegati ad apposite sezioni circuitali che realizzano gli input digitali. Ogni ingresso dispone di un

diodo di protezione, una resistenza di pull-up e un LED che ne segnala lo stato. Comunque in questa applicazione non usiamo gli input.

La gestione dell'MCP23017 avviene da I²C Bus, realizzato con i piedini 12 (SCL) e 13 (SDA), i quali sono collegati ai relativi pin di Fishino. L'MCP23017 offre la possibilità di generare un interrupt a fronte della variazione di almeno un pin programmato come input; nel nostro caso non abbiamo sfruttato questa funzione perché non ci serve acquisire lo stato degli ingressi. L'indirizzo I²C dell'MCP23017 viene selezionato tramite il dip-switch SW1.

Lo shield si alimenta tramite il Vin di Fishino, grazie ad un regolatore interno, basato su un L78L05, che ricava 5 V stabilizzati; per attivare questa opzione, il jumper JP5V va chiuso su 5VR. Chiudendolo su 5VA, lo shield riceve i 5 V direttamente dalla linea +5V di Fishino; ciò è possibile solo se il numero di schede è limitato a due.



inviare tweet duplicati in breve sequenza; per esempio:

#fishino 24 ON accende l'uscita 24
#fishino 24 OFF spegne l'uscita 24
#fishino 24 ON accende l'uscita 24

Il terzo tweet non viene accettato ma segnalato con un "hai già inviato questo tweet". A ciò si può ovviare molto semplicemente aggiungendo qualche carattere a caso dopo la scritta ON:

#fishino 24 ON 123456 accende l'uscita 24

I caratteri supplementari (123456) vengono trascurati dal ricevitore.

FUNZIONAMENTO

Partiamo dall'invio del tweet, che viene memorizzato e propagato

da Twitter. Chiunque può vedere, ricevere e cercare il tweet nella rete da qualsiasi parte del mondo. Utilizzando un PC o un telefono cellulare si può effettuare la ricerca, per esempio, impostando come parametri il mittente del tweet e l'hashtag (**Fig. 1**).

Il risultato della ricerca è qualcosa di simile alla **Fig. 2**.

Come si nota, vengono mostrati i tweet corrispondenti al criterio di ricerca, dall'ultimo arrivato a quelli precedenti.

Analizziamo un po' più in dettaglio un singolo tweet; ogni messaggio contiene:

- un mittente (@**mdelfede** in questo caso);
- nessuno, uno o più hashtag

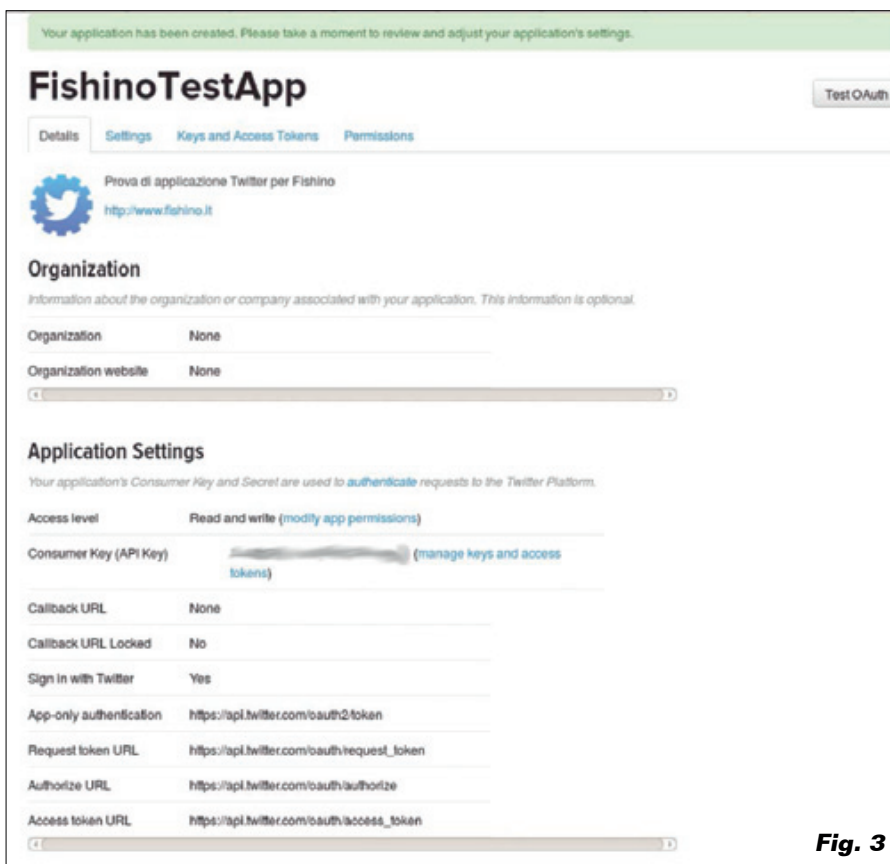


Fig. 3

(#fishino in questo caso);

- un messaggio di testo, comprendente gli hashtag e altro;
- un timestamp, ovvero data ed ora di invio.

Il tweet può anche contenere riferimenti ad altre persone, immagini, URL, eccetera.

Nel nostro sistema sono importanti il **mittente**, utilizzato per consentire o negare l'accesso, gli **hashtag**, utilizzati per differenziare un tweet normale da uno di comando al sistema, il testo, che contiene il comando vero e proprio, ed infine il **timestamp**, o data e ora di invio, che serve per discriminare i comandi già ese-

guiti da quelli ancora da eseguire. Ovviamente non possiamo utilizzare una ricerca simile tramite un PC per controllare i nostri dispositivi, ma abbiamo bisogno di qualcosa di più "digeribile" da un sistema automatico, ed il più semplice possibile.

Per questo ci vengono in aiuto le API (Application Programming Interfaces) fornite da Twitter che permettono di effettuare, tra le altre cose, delle ricerche sui tweet.

LE API DI TWITTER

Le API in questione sono molto estese, permettendo di effettuare moltissime operazioni, sia di ricerca che di invio di tweet tra-

mite software. Attualmente, per il nostro controller ci interessano solo le API di ricerca.

Il funzionamento di tutte le API è comunque abbastanza omogeneo e, una volta capiti i meccanismi di un loro sottoinsieme, risulta facilissimo utilizzarle.

Iniziamo con il superare il primo scoglio incontrato durante lo sviluppo, e che è lo scoglio più grosso che ha obbligato molti progetti analoghi ad utilizzare sistemi molto più complessi di Fishino e/o PC di appoggio per la ricerca dei tweet: la sicurezza. Twitter utilizza, per rendere sicuro il suo sistema, principalmente due meccanismi i quali prevedono che:

- la connessione al server delle API debba avvenire con protocollo sicuro https;
- l'utente delle API debba autenticarsi in uno dei due modi che vedremo in seguito.

Se il secondo requisito è relativamente semplice da soddisfare, il primo impone -per contro- l'utilizzo di un sistema in grado di effettuare connessioni con protocollo sicuro HTTPS. Questo esclude automaticamente gli Ethernet Shield e WiFi Shield di Arduino, per esempio, che non sono in grado di utilizzarlo. Invece Fishino è in grado di eseguire connessioni https anche se col limite di una connessione per volta, vista la grande richiesta di risorse di memoria necessaria per la crittografia.

Vediamo, nel **Listato 1**, una

Listato 1

```
GET
/1.1/search/tweets.json?count=1&q=%23fishino+from%3Amdelfede&result_type=recent&max_id=647068135569387520&since_id=646038066440896517 HTTP/1.1
Host: api.twitter.com
User-Agent: Fishino Twitter Client
Authorization: Bearer AAAAAAAAAAAAAAAAAA(altri caratteri della chiave di autorizzazione)
```

Application Management

To bring you Twitter, we and our partners use cookies on our and other websites. Cookies help personalize Twitter content, tailor Twitter Ads, measure their performance, and provide you with a better, faster, safer Twitter experience. By using our services, you agree to our [Cookie Use](#).

Create an application

Application Details

Name *
FishinoTestApp
Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *
Prova di applicazione Twitter per Fishino
Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *
http://www.fishino.it
Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens. (If you don't have a URL, yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their callback URL on the request token step, regardless of the value given here. To redirect your application from using callbacks, leave this field blank.

Fig. 4

FishinoTestApp

Details Settings Keys and Access Tokens Permissions

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key) [redacted]
Consumer Secret (API Secret) [redacted]
Access Level: Read and write (modify app permissions)
Owner: mdelfede
Owner ID: 229626414

Application Actions
Regenerate Consumer Key and Secret Change App Permissions

Your Access Token
You haven't authorized this application for your own account yet.
By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions
Create my access token

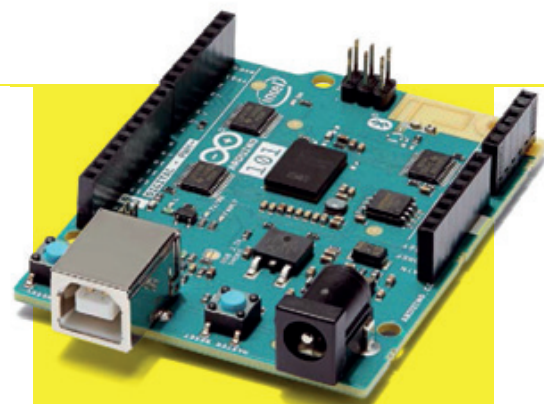
Fig. 5

richiesta https tipica alle API di Twitter, che andremo a esaminare in dettaglio. La prima riga è la richiesta vera e propria:

- **GET /1.1/search/tweets.json?** è la richiesta alle API di ricerca dei tweet;
- **count=1:** è il numero di risul-

tati richiesti; ne vogliamo uno per volta;

- **q= %23fishino+from %3Amdelfede** significa che cerchiamo l'hashtag (%23, #) fishino proveniente (%3A,@) da mdelfede;
- **result_type=recent** significa



Arduino sposa Intel

All'ultimo Rome Maker Faire, Intel e Massimo Banzi hanno presentato Genuino 101, una scheda di sviluppo pensata per la scuola, il mondo dei maker e per quanti si avvicinano per la prima volta alla programmazione. Genuino 101 è basata su Intel Curie, un SoC a basso consumo dalle dimensioni ridottissime, nato per applicazioni wearable; esso implementa un modulo Bluetooth LE e un accelerometro giroscopio a 6 assi. Il core è un Quark x86 a 32 bit con clock a 32 MHz nel quale gira un sistema operativo RTOS ViperOS. La board utilizza anche un core ARC (Argonaut RISC Core) a 32 bit anch'esso con clock a 32 MHz. Per il resto la basetta ha lo stesso form-factor di Arduino UNO e dispone di 14 pin digitali di I/O (di cui 4 possono essere utilizzati come uscite PWM), 6 ingressi analogici, un connettore USB per la comunicazione seriale e per il caricamento degli sketch, un jack di alimentazione e un connettore ICSP con SPI e I²C. Le linee di I/O funzionano a 3,3V ma sono protette contro le sovratensioni fino a 5 V. Genuino 101 sarà adottata da Creative Technologies in the Classroom, programma di physical computing attualmente implementato in oltre 300 scuole.

FishinoTestApp Test OAuth

[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[REDACTED]
Consumer Secret (API Secret)	[REDACTED]
Access Level	Read and write (modify app permissions)
Owner	mdelleda
Owner ID	229626414

Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[REDACTED]
Access Token Secret	[REDACTED]
Access Level	Read and write
Owner	mdelleda
Owner ID	229626414

Fig. 6

che vogliamo i risultati in ordine di tempo e non di popolarità;

- **max_id= 647068135569387520**; gli ID sono numeri di sequenza contenuti in ogni messaggio Twitter, e che forniscono la sequenza temporale dei medesimi; in questo caso vogliamo cercare un messaggio non successivo al numero impostato;
- **since_id= 46038066440896517**

vuol dire che vogliamo un messaggio successivo al numero impostato;

- **HTTP/1.1** chiude la query http specificando il formato del protocollo.

La seconda e la terza riga indicano rispettivamente l'host cui viene inviata la richiesta (*api.twitter.com*) e l'applicazione che la

sta effettuando (Fishino Twitter Client); quest'ultima è arbitraria e serve solo ad identificare l'applicazione che utilizza le API. La quarta riga è molto importante, perché è quella utilizzata per l'autenticazione della richiesta sulla rete twitter:

Authorization:
Bearer AAAAAAAAAAAAAAAAAAAAAA
 (altri caratteri della chiave di autorizzazione)

Come già detto Twitter utilizza due metodi per rafforzare la sicurezza delle connessioni: il protocollo https (che impedisce gli attacchi tipo "man in the middle") e delle chiavi di autenticazione. Queste ultime sono di due tipi: personali o di applicazione (application-only); con il primo tipo di autenticazione il programma accede all'API come se fosse un utente specifico di Twitter ed agisce in suo nome; può, per esempio, inviare tweet come se fossero stati inviati da un PC. Il secondo tipo di autenticazione è invece legato all'applicazione, quindi parzialmente anonimo; non permette l'invio di messaggi ma solo alcune operazioni: per esempio la ricerca dei tweet. Nel nostro progetto abbiamo utilizzato il secondo metodo, che da un lato è più limitato perché Twitter pone dei limiti temporali all'utilizzo delle API: non è possibile eseguire oltre 180 operazioni ogni 15 minuti come utente o come applicazione. Quindi, se eseguiamo 180 richieste in 15 minuti come applicazione ce ne rimangono altrettante a disposizione come utente. 180 richieste ogni 15 minuti equivalgono ad un intervallo di 5 secondi tra 2 richieste.

Superare questi limiti vuol dire rischiare di farsi bloccare l'account Twitter, quindi è opportuno evitare richieste troppo frequenti.

Twitter Bearer Token Generator

CONSUMER_KEY : [REDACTED]

CONSUMER_SECRET : [REDACTED]

[Generate token](#)

BEARER TOKEN: AAAAAAAAAAAAAAAAAAAAAA [REDACTED]

Fig. 7

Listato 2

```
////////////////////////////////////
////////////////////////////////////
// CONFIGURATION DATA          -- ADAPT TO YOUR NETWORK !!!
// DATI DI CONFIGURAZIONE -- ADATTARE ALLA PROPRIA RETE WiFi !!!

////////////////////////////////////
////////////////////////////////////
// CONFIGURATION DATA          -- ADAPT TO YOUR NETWORK !!!
// DATI DI CONFIGURAZIONE -- ADATTARE ALLA PROPRIA RETE WiFi !!!

// here put SSID of your network
// inserire qui lo SSID della rete WiFi
#define MY_SSID    "CASA"

// here put PASSWORD of your network. Use "" if none
// inserire qui la PASSWORD della rete WiFi -- Usare "" se la rete non è protetta
#define MY_PASS    "unapassword"

// here put required IP address of your Fishino
// comment out this line if you want AUTO IP (dhcp)
// NOTE : if you use auto IP you must find it somehow !
// inserire qui l'IP desiderato per il fishino
// commentare la linea sotto se si vuole l'IP automatico
// nota : se si utilizza l'IP automatico, occorre un metodo per trovarlo !
#define IPADDR     192, 168, 1, 251

// here put your application Bearer authorization key
// see https://dev.twitter.com/oauth/application-only for details
// inserire qui la chiave "bearer" di autenticazione dell'applicazione
// vedere https://dev.twitter.com/oauth/application-only per dettagli
#define BEARER     "AAAAAAAAAAAAAAAAAAAAA....."

// here put your Twitter user name; if empty the application will react to ANY user
// inserire qui il nome utente ai cui tweets il programma si attiva; se vuoto, il programma si attiva con TUTTI gli utenti
#define TWITTER_USER "pippo"

// here put twitter hashtag needed to trigger the event (without the # char!)
// inserire qui l'hashtag necessario ad avviare l'applicazione (senza il carattere #!)
#define TWITTER_HASHTAG "fishino"

// here put the delay in milliseconds between requests
// there is a limit of about 180 requests every 15 minutes, so don't put a too short time here
// inserire qui il ritardo tra due richieste successive
// c'è un limite di circa 180 richieste ogni 15 minuti, quindi non inserire un tempo troppo breve
#define REQUESTS_INTERVAL 5000
```

OTTENERE UNA CHIAVE 'BEARER' DI AUTORIZZAZIONE

Vediamo come creare un'applicazione Twitter e ottenere la relativa chiave di autorizzazione. Innanzitutto occorre essere registrati a Twitter e aprire l'indirizzo <https://apps.twitter.com/>.

Facendo clic sul pulsante Create New App appare la schermata di Fig. 3. Qui occorre riempire alcuni campi obbligatori quali:

- **Nome applicazione (Name);** in questo caso, **FishinoTestApp**;
- **Descrizione applicazione**

(Description); in questo caso "Prova di applicazione Twitter per Fishino";

- **Sito web (Website)** è un sito che dovrebbe contenere i dettagli dell'applicazione; è obbligatorio, ma può essere anche solo un segnaposto; in questo caso è www.fishino.it;
- **Callback URL**; non necessaria per i nostri scopi (lasciare in bianco).

In fondo alla pagina ci sono le usuali condizioni del servizio da accettare (selezionare la

checkbox) e il pulsante per creare l'applicazione (Create your Twitter application); cliccando appare la pagina di Fig. 4.

L'applicazione è stata creata e viene fornita (qui l'abbiamo oscurata) la 'consumer API key', che è una delle chiavi necessarie all'utilizzo delle API. Poiché vogliamo usare la 'Bearer token', ovvero la chiave necessaria in modalità 'Application only', serve un ulteriore passo: in alto cliccate sul tab 'Keys and access tokens'; vi apparirà la schermata in Fig. 5, che informa che l'applicazione

Listato 3

```
if (!client.connected() && !client.available())
{
    Serial.println("\nStarting connection to server...");
    if(client.connect("api.twitter.com", 443))
    {
        Serial.println("connected to server");

        // run the query looking from given hashtag and user
        doQuery(client, "%23" TWITTER_HASHTAG);
    }
}
```

Listato 4

```
// run a query on twitter
// esegue la query a Twitter
void doQuery(FishinoSecureClient &client, const char *query)
{
    // if walkback is finished (nextID == -1) we reset sinceID to next found maximum ID
    // otherwise we let it as it is and continue the walk back
    // se il walkback è finito (nextID == -1) re-impostiamo il sinceID a dopo
    // il massimo ID trovato
    // altrimenti lasciamo tutto come sta e continuiamo col walkback
    if(nextID == (uint64_t)-1)
        sinceID = maxID;

    client.print("GET /1.1/search/tweets.json?count=1&q=");
    client.print(query);
    if(strlen(TWITTER_USER))
    {
        client.print("&from%3A");
        client.print(TWITTER_USER);
    }
    client.print("&result_type=recent");
    if(nextID != (uint64_t)-1)
    {
        client.print("&max_id=");
        client << nextID;
    }
    client.print("&since_id=");
    client << sinceID;
    client.println(" HTTP/1.1");

    client.println("Host: api.twitter.com");
    client.println("User-Agent: Fishino Twitter Client");

    // send auth key to server
    // invia la chiave di autorizzazione al server
    client.println("Authorization: Bearer " BEARER);
    client.println();

    // reset nextID - it will be filled by parser if more results are pending
    // azzerare il nextID - verrà riempito dal parser se ci sono altri risultati da leggere
    nextID = -1;
}
```

non è ancora stata autorizzata (voce sotto **'Your Access Token'**). Facendo clic sul pulsante **'Create my access token'** in basso si ottiene la schermata di **Fig. 6**. Sotto la voce **'Your Access Token'** appaiono ora due chiavi che vanno utilizzate per l'uso dell'applicazione in modalità normale (**non Application-only**), vale a dire come se foste voi ad eseguire

i comandi. L'ultimo passo per ottenere la chiave **Application-only**, che è poi quella che useremo nella nostra applicazione; sarebbe il più complicato, visto che nelle pagine della documentazione delle API viene spiegato come fare ma, purtroppo, non viene fornito un sistema per farlo in modo semplice. Abbiamo quindi provveduto a realizzare

una pagina all'indirizzo *www.fishino.it/bearer.php* che permette di ottenere in modo facilissimo la cosiddetta **'Bearer token'**. È sufficiente inserire la **'consumer_key'** e la **'consumer_secret'** dell'applicazione, ricavate in precedenza, e premere il pulsante **'Generate token'** per ottenere la chiave da copiare ed incollare nello sketch. Ultima nota: la chiave appare sullo schermo in due linee per motivi di larghezza di pagina, ma è da copiare e incollare in una sola linea.

LO SKETCH PER FISHINO

Passiamo ora ad esaminare lo sketch (**Listato 2**) partendo dall'inizio del file, dove abbiamo scelto di configurare pochi elementi per dare modo agli "impazienti" di provare subito il sistema. Le impostazioni sono poche e semplici; chi già conosce le applicazioni dimostrative di Fishino noterà che lo stile è sempre lo stesso e i contenuti variano di poco; in particolare, verso la fine del gruppo di impostazioni si notano quelle rilevanti per Twitter, ovvero la Bearer Token di cui abbiamo parlato al paragrafo precedente (**BEARER**), il nome utente (**TWITTER_USER**) da cui devono provenire i tweet, l'hashtag che attiva il programma (**TWITTER_HASHTAG**), da inserire senza il carattere di cancelletto (#) e l'intervallo tra due richieste al sistema. Per quanto riguarda quest'ultimo, per rispettare il limite di 180 richieste in 15 minuti abbiamo impostato una richiesta ogni 5 secondi circa (#define REQUESTS_INTERVAL 5000). Sugeriamo di non scendere sotto il valore 5.000 (millisecondi) se non per brevi prove.

DESCRIZIONE DELLO SKETCH

Una buona parte dello sketch si occupa dell'inizializzazione della

Listato 5

```
// analyze data and act on devices
// analizza i dati in arrivo da Twitter
void parserCallback(uint8_t filter, uint8_t level, const char *name, const char *value)
{
    // check for user name
    // controlla il nome utente
    if(level == 3 && !strcmp(name, "screen_name") && !strcmp(value, "\"" TWITTER_USER "\""))
        gotUser = true;

    // check twitter message - MUST contain the requested hashtag, a port number and word ON or OFF
    // controlla il testo del messaggio - DEVE contenere l'hashtag richiesto, un numero di porta e le parole ON o OFF
    else if(level == 2 && !strcmp(name, "text") && value && value[0] && !strnicmp(value + 1, "#" TWITTER_HASHTAG, strlen(TWITTER_HASHTAG) + 1))
    {
        // hashtag found, skip it
        // trovato l'hashtag, lo salta
        const char *p = value + strlen(TWITTER_HASHTAG) + 3;

        // skip spaces after hashtag
        // salta gli spazi dopo l'hashtag
        while(*p && isspace(*p))
            p++;

        // read port number - sets it to 0xff if not found
        // legge il numero di porta - lo imposta a 0xff se non trovato
        if(isdigit(*p))
        {
            port = 0;
            while(isdigit(*p))
                port = port * 10 + *p++ - '0';
        }
        else
            port = 0xff;

        // if port not found, just leave
        // se non trova il numero di port ritorna
        if(port == 0xff)
            return;

        // skip spaces after port number
        // salta gli spazi dopo il numero di porta
        while(*p && isspace(*p))
            p++;

        // check for ON or OFF keywords - if not found sets port to 0xff and leave
        // cerca le parole chiave ON o OFF - se non trovate imposta la porta a 0xff ed esce
        if(!strnicmp(p, "on", 2))
            command = true;
        else if(!strnicmp(p, "off", 3))
            command = false;
        else
            port = 0xff;
    }

    // look if there are previous tweets to handle
    // cerca se deve elaborare tweets precedenti
    else if(level == 1 && !strcmp(name, "next_results"))
    {
        // get next tweet id to handle
        const char *s = strstr(value, "\"?max_id=");
        if(s)
        {
            s += 9;
            nextID = readID(s);
        }
    }

    // get max id of search, needet to go further after walk back
    // cerca il massimo ID corrispondente alla ricerca, necessario per proseguire dopo la ricerca all'indietro
    else if(level == 2 && !strcmp(name, "id"))
    {
        uint64_t id = readID(value);
        if(id > maxID)
            maxID = id;
    }
}
```

Listato 6

```
// waits for data from twitter and parse them - return true on success, false if no data
bool doReceive(FishinoSecureClient &client, JSONStreamingParser &parser)
{
    // wait 1 second max for data from server
    // attende al massimo 1 secondo i dati dal server
    unsigned long tim = millis() + 1000;
    while(!client.available() && millis() < tim)
        ;

    // if no data is available, close the connection
    // se non ci sono dati, chiude la connessione
    if(!client.available())
        return false;

    // skip data up to 2 consecutive cr
    bool cr = false;
    while(client.available())
    {
        if(cr)
        {
            if(client.read() == 0x0d && client.read() == 0x0a)
                break;
            else
                cr = false;
        }
        else if(client.read() == 0x0d && client.read() == 0x0a)
            cr = true;
    }

    while(client.available())
    {
        char c = client.read();
        parser.feed(c);
    }
    return true;
}
```

scheda, della connessione alla rete WiFi, eccetera.

Per brevità rimandiamo agli articoli di presentazione di Fishino; qui vediamo in dettaglio solo le parti più interessanti dello sketch, ovvero la gestione della connessione sicura alle API di Twitter e l'utilizzo delle schede di espansione I/O.

Per quanto riguarda la connessione alle API, notiamo subito una differenza rispetto agli sketch visti in precedenza:

```
// The web secure client used
to access twitter API
// il client https usato per
l'accesso all'API di Twitter
FishinoSecureClient client;
```

Qui si nota l'utilizzo del FishinoSecureClient al posto del precedente FishinoClient. Questo client viene utilizzato per le connessioni sicure https.

La connessione alle API avviene tramite lo spezzone di codice contenuto nel **Listato 3**: prima viene controllato se il client è ancora connesso dal loop precedente e/o se ha ancora dei dati da leggere, visto che i dati sono mantenuti in un buffer nel client anche dopo la chiusura dello stesso. In caso contrario viene aperta la connessione al sito *api.twitter.com* sulla porta 443, che è quella utilizzata nelle connessioni https.

Se la connessione ha esito positivo, viene inviata la richiesta vera e propria tramite la funzione `doQuery()`, che ha come parametri il client e l'hashtag da cercare. La funzione non fa altro che inviare i dati come visto durante la spiegazione delle API (**Listato 4**). La particolarità di questa funzione, a parte l'invio dei dati di ricerca veri e propri, sta nella gestione dei campi `since_id` e `max_id` che consentono di restringere la ricerca in ben precisi limiti temporali.

Listato 7

```
// if port has been set, change its output
// se la porta è stata impostata, ne modifica l'output
if(port != 0xff)
{
    // if no expanders found, check if pin is free on Fishino
    // and set its output (must be changed on Mega!)
    if(!numExpanders)
    {
        if(port == 4 || port == 7 || port >= 10)
            Serial << F("INVALID PORT #") << port << F(" -- COMMAND DISCARDED\n");
        else
        {
            Serial << "\n\nSETTING PORT " << port << " TO " << (command ? "HIGH" : "LOW") << "\n";
            pinMode(port, OUTPUT);
            digitalWrite(port, command ? HIGH : LOW);
        }
    }
    else
    {
        uint8_t board = port / 8;
        if(board >= numExpanders)
            Serial << F("INVALID PORT #") << port << F(" -- COMMAND DISCARDED\n");
        else
        {
            Serial << "\n\nSETTING PORT " << port << " TO " << (command ? "HIGH" : "LOW") << "\n";
            port %= 8;
            expanders[board]->digitalWrite(15-port, command ? HIGH : LOW);
        }
    }
}
```

Per motivi di spazio e velocità, ad ogni richiesta scegliamo infatti di ricevere un solo tweet, che è l'ultimo presente in rete tra i due ID forniti; giocando su questi ultimi è possibile andare a ritroso richiedendo i tweet precedenti, fino all'esaurimento dei medesimi. Finita la ricerca all'indietro, viene reimpostato il **since_id** in modo che nelle richieste successive vengano selezionati solo i tweet più nuovi.

Una volta inviata la richiesta, occorre ricevere ed analizzare i dati. Le API di Twitter rispondono con una serie di dati in formato JSON e di dimensioni molto variabili, anche piuttosto grandi. Vista la limitatezza delle risorse di memoria della scheda, una memorizzazione di questi dati è impensabile; occorre quindi analizzarli man mano che arrivano. Per far questo abbiamo scritto una libreria completa per il parsing (scansione) dei dati in arrivo in modalità streaming, ovvero carattere per carattere, senza memorizzazioni intermedie.

Questa libreria, **JSONStreamingParser**, è diversa dalla maggior parte delle librerie analoghe presenti in rete visto che, oltre all'analisi dei dati in tempo reale, non memorizza nemmeno i risultati, se non quello corrente.

In pratica, i dati JSON hanno un formato simile a questo:

```
{
  nome1 : dato1,
  nome2 : {nome3 : dato3, nome4 : dato4},
  nome5 : [dato6, dato7, dato8]
}
```

Ovvero, i dati possono contenere valori semplici (stringhe, numeri, eccetera) oppure liste di dati oppure array di dati. Il formato, arrivando ai dati semplici, è comunque sempre lo stesso:

```
nome : dato
```

Listato 8

```
// find all available IOExpander boards
void findIOExpanders(void)
{
  numExpanders = 0;
  for(uint8_t addr = 0; addr < 8; addr++)
  {
    IOExpanderMCP23017 *exp = new IOExpanderMCP23017;

    // set expander address
    exp->begin(addr);

    // this is because expander board hasn't reset connected to arduino one
    // and it checks register 0 upon init to see if board is present
    // (register should be 0xff, all A port as input)
    for(int i = 0; i < 16; i++)
      exp->pinMode(i, INPUT);

    if(exp->init())
    {
      // expander found, add to list and setup pins
      expanders[numExpanders++] = exp;

      // set pin modes
      for(int i = 0; i < 8; i++)
        exp->pinMode(15-i, OUTPUT);
    }
    else
      // not found, delete expander object
      delete exp;
  }
  if(numExpanders)
    Serial << F("Found ") << numExpanders << F(" I/O expander boards\n");
  else
    Serial << F("No expander boards found - using Fishino's I/O pins\n");
}
```

Il parser, quando determina un nome, esegue la scansione del dato; se questo è un tipo semplice, chiama una funzione predefinita con i seguenti parametri:

```
void callback(uint8_t filter,
uint8_t level, const char *name,
const char *value
```

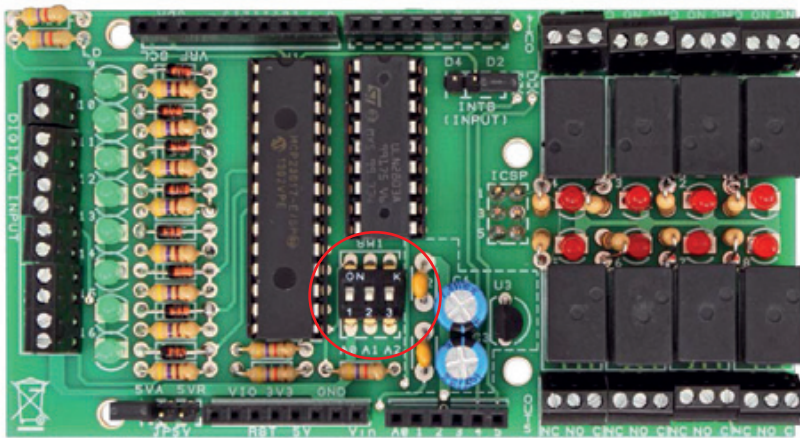
dove il parametro 'filter' è previsto per future espansioni, il parametro 'level' indica il livello di nidificazione del dato ed i parametri 'name' e 'value' indicano nome e valore del dato. Se il dato non è semplice ma costituito da una lista o da un array, il campo 'value' assume il valore "<LIST>" o "<ARRAY>" e la scansione prosegue al suo interno. Creando opportune funzioni di **callback** è quindi possibile analizzare al volo il contenuto del **JSON** ricevuto ed impostare le

variabili opportune nello sketch. L'attuale funzione di callback è contenuta nel **Listato 5**.

La funzione sembra complessa, ma in realtà non fa altro che sfruttare i parametri 'level' e 'name' per determinare se sono presenti o meno i campi richiesti nel tweet e, nel caso, ne estrae ed analizza i valori.

Qui si nota che viene controllato l'utente che ha inviato il tweet, gli hashtag per vedere se c'è quello impostato, il parametro 'text' per leggere i comandi inviati e gli ID utilizzati per controllare la sequenza temporale dei tweet.

Come dicevamo, la funzione viene richiamata dal **JSONStreamingParser** ogni volta che questo legge un valore dallo stream. Il parser, a sua volta viene utilizzato dalla funzione **doReceive()**



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

come mostrato nel **Listato 6**.

Questa funzione non fa altro che leggere i dati dal client, saltare il preambolo HTTP (la cui fine è indicata da una linea vuota) ed inviare i restanti caratteri, uno alla volta, al parser, per l'analisi. Una volta completata l'analisi dello stream di dati, in caso di comando valido la variabile globale 'port' conterrà il numero di porta da modificare e la variabile globale 'command' (booleana) sarà impostata a true se vorremo portare l'uscita a, valore HIGH, oppure false se la vorremo a LOW.

Vediamo la parte che attiva l'uscita, interessante per il fatto che prevede la possibilità di applicare gli I/O Expander a Fishino (**Listato 7**). Qui si nota per prima cosa il controllo sulla validità della porta (port); se è a 0xff il comando è da scartare, altrimenti viene proseguita l'analisi. Successivamente il codice agisce in modo differente se sono presenti o meno le espansioni. Nel primo caso, determina il numero di scheda di espansione (port / 8) e la porta sulla medesima (port % 8). Per esempio, un valore di port pari a 33 corrisponde alla scheda n° 4 (33/8) ed alla sua porta n° 1 (33 % 8 = 33 - 4 •

8 = 1). L'operatore '%' fornisce infatti il resto della divisione. Nel caso non siano presenti le espansioni, lo sketch utilizza il numero di porta direttamente come I/O digitale del Fishino; controlla che sia un valore valido e che non vada a modificare pin utilizzati per altri scopi ed agisce su quello.

UTILIZZO DELLE SCHEDE DI ESPANSIONE I/O

Lo sketch prevede, in modo completamente automatico, il riconoscimento di eventuali schede I/O Expander anche connesse in cascata, fino ad un massimo di 8. L'importante è avere l'accortezza di fornire indirizzi I²C differenti ad ogni scheda.

All'avvio lo sketch controlla, nel setup(), se sono presenti le schede e le numera in base all'indirizzo I²C. Se ne trova, assegna i port di uscita in sequenza: da 0 a 7 per la prima scheda, da 8 a 15 per la seconda e così via.

Nel caso non venga rilevata alcuna scheda di espansione, lo sketch va in modalità standalone utilizzando tutti gli I/O liberi di Fishino, e quindi le uscite 0, 1, 2, 3, 5, 6, 8 e 9, essendo le altre riservate al funzionamento delle periferiche incluse.

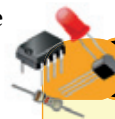
In questo caso il numero di porta corrisponde al numero dell'I/O digitale di Fishino.

La numerazione delle schede avviene tramite lo spezzone di codice contenuto nel setup (**Listato 8**).

Si nota il ciclo che prova tutti gli indirizzi I²C possibili (da 0 a 7); per ogni indirizzo crea una variabile IOExpanderMCP23017 (in modo dinamico, come puntatore), ne esegue il test di presenza e, nel caso funzioni la memorizza in un array di puntatori. Se la scheda non è presente elimina la variabile e riparte con la successiva. In questo modo le schede vengono rilevate in ordine di indirizzo I²C.

Con questo abbiamo concluso la descrizione dell'applicazione **Fish'n Tweets!**

In un prossimo articolo vedremo come fare l'operazione inversa, ovvero far inviare dei tweet alla scheda quando alcuni pin di ingresso cambiano di stato, cosa utilissima per esempio per realizzare degli allarmi o monitorare eventi in un luogo remoto. ■



per il MATERIALE

Tutto il materiale necessario per realizzare questo progetto è disponibile presso Futura Elettronica. Lo shield Arduino I/O expander in kit (cod. FT1079K) è disponibile a Euro 34,00. La board Fishino (cod. FISHINOUNO) viene fornita montata e collaudata al prezzo di Euro 36,00. Software e sketch sono scaricabili gratuitamente sul sito www.elettronica.in.it. 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>



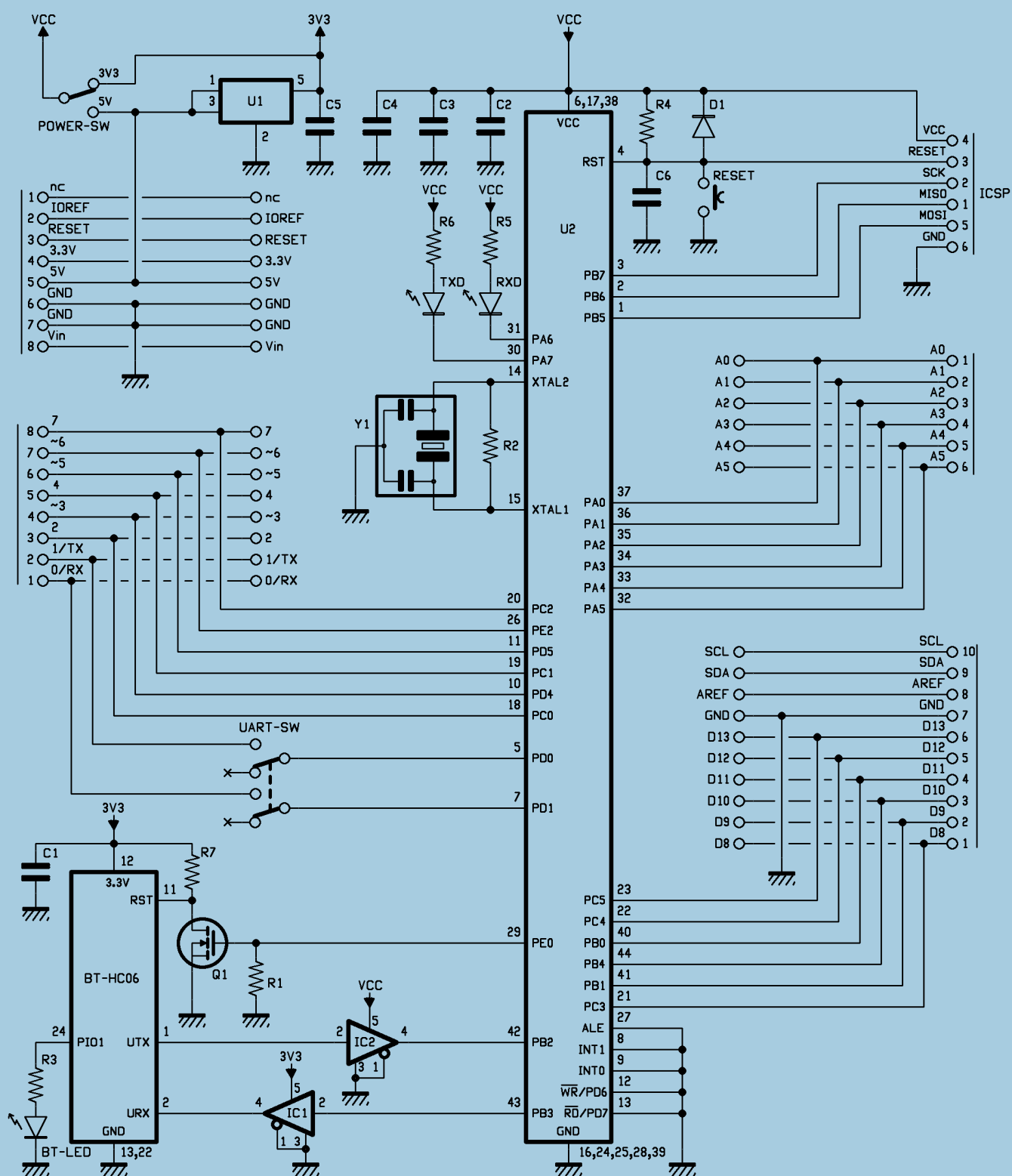
1SHEELD CONTAPERSONE

Uno shield virtuale e uno smartphone, cui aggiungiamo dei sensori a ultrasuoni, ci permettono di rilevare il passaggio e la direzione di chi transita da un varco.

..... dell'Ing. MIRCO SEGATELLO

Gran parte del successo di Arduino è dovuto anche al fatto che le funzionalità base di tale scheda possono essere espanse semplicemente aggiungendo degli shield, ovvero board contenenti l'hardware che serve per ogni specifica applicazione. La possibilità di aggiungere nuovo hardware, alla scheda base già esistente, fornisce quasi infinite possibilità di sviluppo e ha spronato le più importanti case produttrici di componenti e dispositivi elettronici a proporre soluzioni innovative e inte-

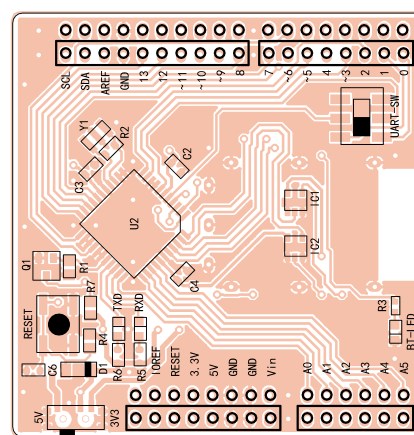
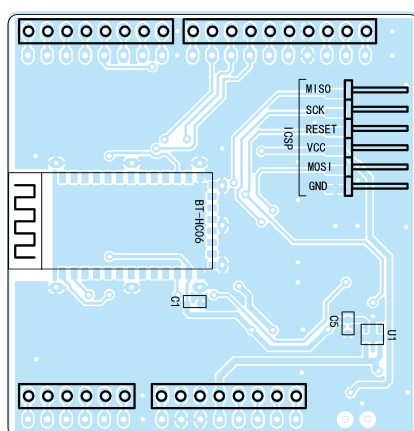
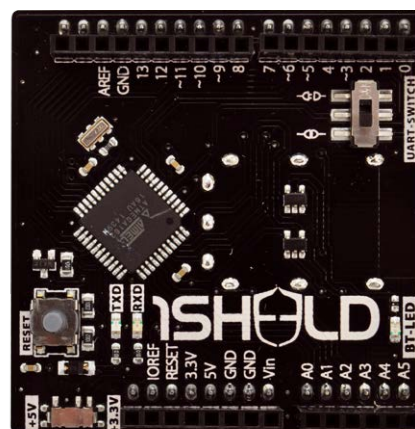
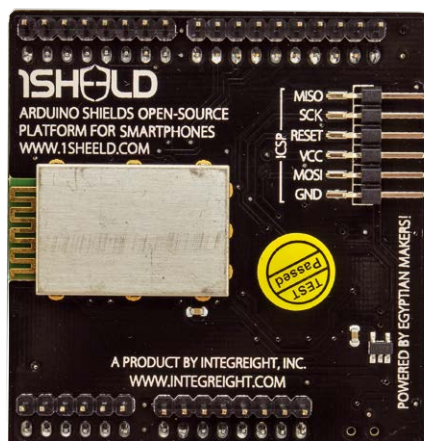
ressanti. Volete dotarvi di un localizzatore satellitare? Aggiungete uno shield con un ricevitore GPS! Volete un'applicazione con un display grafico? Basta aggiungere uno shield con un GLCD (Graphic Liquid Crystal Display). Oramai per ogni funzione esiste lo shield dedicato, disponibile commercialmente anche in numerose varianti a seconda del produttore. In questo scenario il team di Integreight ebbe un'idea quanto-mai geniale, pensando alle potenzialità dell'interazione tra una scheda Ardu-



ino e uno smartphone di ultima generazione. In effetti, tante funzioni svolte da degli shield sono già presenti in uno smartphone,

che in quanto a sensori integrati è ben fornito; inoltre contiene anche una sofisticata sezione di comunicazione che spazia dal

Bluetooth al 3G. E per finire, integra un display ad alta risoluzione. Ebbene, il team Integreight ha pensato che ogni funzionalità



Elenco Componenti:

C1: 100 nF ceramico (0603)
C2: 100 nF ceramico (0603)
C3: 100 nF ceramico (0603)
C4: 100 nF ceramico (0603)
C5: 1 μ F ceramico (0603)
C6: 100 nF ceramico (0603)
R1: 4,7 kohm (0805)
R2: 1 Mohm (0603)
R3: 470 ohm (0402)
R4: 10 kohm (0805)
R5: 1 kohm (0603)
R6: 1 kohm (0603)
R7: 10 kohm (0805)

IC1: 74LVC1G125DBV
IC2: 74LVC1G125DBV
D1: CD1206-S01575
BT-LED: LED Blu (0805)
RXD: LED Rosso (0805)
TXD: LED Rosso (0805)
BT-HC06: Modulo Bluetooth HC-06
Q1: TN0201K
Y1: Modulo a quarzo 7,37 MHz
U1: LP2985-33DBVR
U2: ATmega162
UART-SW: Deviatore 2 vie

POWER-SW: Deviatore 1 via
RESET: Microswitch

Varie:

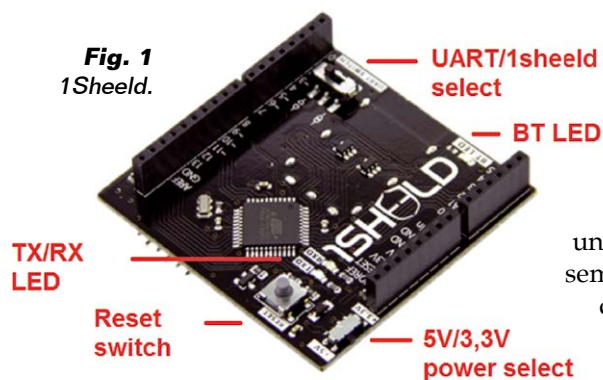
- Strip Femmina 6 vie
- Strip Femmina 8 vie (2 pz.)
- Strip Femmina 10 vie
- Strip Maschio 6 vie
- Strip Maschio 8 vie (2 pz.)
- Strip Maschio 10 vie
- Strip maschio 6 vie 90°
- Circuito stampato S1285 (55 x 56 mm)

dello smartphone potesse rappresentare uno shield per Arduino; era solo necessario metterli in comunicazione con Arduino ed il

gioco era fatto. Da qui nacque l'idea di 1Sheeld, ovvero un unico shield per Arduino che permette di sfruttare tutte le funzionalità

di uno smartphone; il gioco di parole è stato voluto, perché il nome si legge come "uno shield" ma è scritto 1Sheeld, ad indicare

Fig. 1
1Sheeld.



che è uno shield unico per tante funzioni.

Quindi se nel progetto ci occorre un localizzatore satellitare, invece di acquistare uno shield specifico sfruttiamo il ricevitore GPS dello smartphone, e se serve un display grafico possiamo sfruttare sempre quello dello smartphone. Ovviamente anche la sezione di comunicazione e il supporto per l'invio di e-mail, SMS o l'accesso a social network è a disposizione delle nostre applicazioni.

In totale, il team di Integreight è stato in grado di implementare con 1Sheeld più di 40 shield virtualizzati; la lista completa è disponibile all'indirizzo: <http://1sheeld.com/shields>. 1Sheeld è costituito da due blocchi hardware, uno dei quali è lo smartphone; l'altro è uno shield

di interfaccia specifico (visibile in Fig. 1) basato su un microcontrollore Atmel ATmega162 e un modulo Bluetooth che ha semplicemente lo scopo di far dialogare la nostra scheda Arduino con lo smartphone. L'impegno maggiore da parte dei progettisti è stato quello di sviluppare l'applicazione da far girare sullo smartphone, cercando di ottenere un prodotto molto semplice da utilizzare ma nel contempo dotato della massima flessibilità operativa, in quanto l'utente con pochi tap sul display è in grado di personalizzare l'interfaccia desiderata.

Lo shield Bluetooth interconnette Arduino allo smartphone, dove l'applicazione attiva, inizializza e utilizza i sensori richiesti, che a questo punto è come se fossero a bordo di Arduino e comunque a sua disposizione: una comodità non indifferente.

SHIELD UNICO E HARDWARE VIRTUALE

Vediamo ora più in dettaglio di cosa si tratta e come può essere impiegato questo nuovo prodotto, aiutandoci anche con lo

schema elettrico illustrato nelle pagine precedenti. 1Sheeld si presenta come un normalissimo shield per Arduino Uno/Arduino Mega. La sua tensione di alimentazione può essere impostata a 3,3V o 5 V manualmente tramite il deviatore POWER-SW, il che permette allo shield di funzionare anche con schede a 3,3 volt purché con pedinatura Arduino-compatibile. Comunque la logica dello shield funziona con la tensione prelevata dal cursore del deviatore; se POWER-SW è posizionato su 5V, il regolatore U1 ricava i 3,3V necessari al modulo Bluetooth, che deve funzionare a 3,3 volt. La scheda usa un modulo Bluetooth standard (Bluetooth 2.1) siglato BT-HC06 nello schema elettrico, che gli permette di dialogare con uno smartphone fino alla distanza di circa 10 metri. Il modulo Bluetooth è normalmente resettato attraverso la R7; mediante il MOSFET Q1, quando il microcontrollore pone la propria linea PE0 a livello logico alto, il contatto RST del modulo viene posto a livello basso e il reset viene disattivato. L'UART interno al modulo Bluetooth è interfacciato alla seriale virtuale implementata

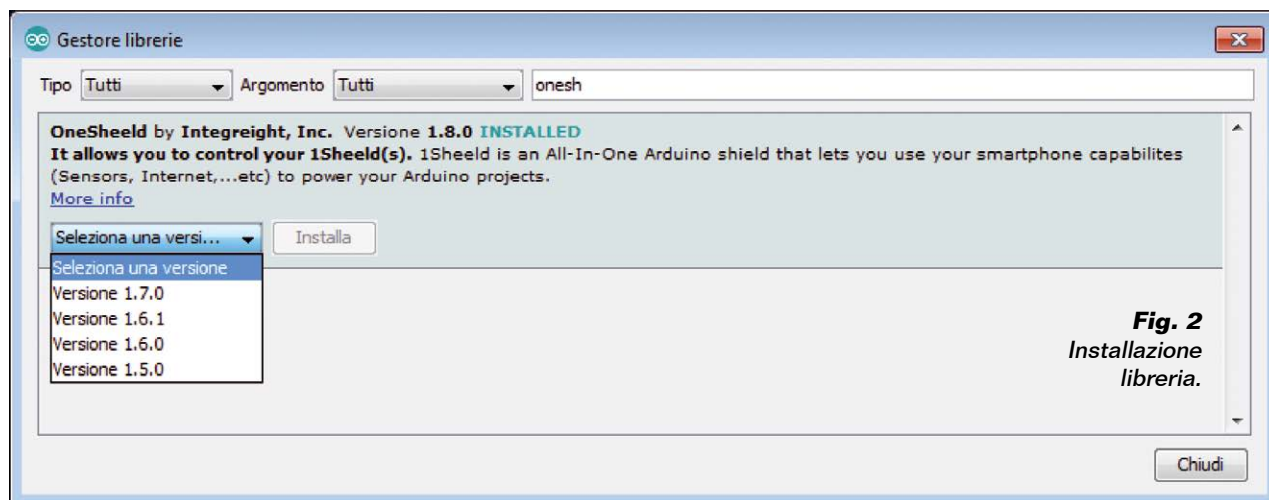


Fig. 2
Installazione
libreria.

in U2 dagli I/O PB2 e PB3 tramite due buffer 74LVC1G125 utilizzati come adattatori di livello quando lo shield funziona a 5V: in tal caso, infatti, la linea PB3 del microcontrollore fornisce impulsi TTL che vengono adattati a 3,3V dall'IC1, mentre la UTX del modulo fornisce impulsi 0/3,3V che vengono traslati a 0/5V dall'IC2. 1Sheeld comunica con la scheda Arduino tramite il modulo UART interno, per cui non sarà possibile utilizzare la funzionalità di Serial Monitor di Arduino mentre è in funzione. La linea PIO1 del modulo Bluetooth pilota, tramite la resistenza R3, il LED BT-LED che fornisce segnalazioni sull'attività della connessione. Per consentire comunque la programmazione di Arduino da PC, lo shield ha il deviatore UART-SW che permette di escludere la comunicazione tra l'UART di Arduino e 1Sheeld. I LED siglati TXD ed RXD (piedini PA6 e PA7 del micro U2) scandiscono l'attività sulla porta seriale. Nella scheda è anche presente un pulsante di reset che replica la funzionalità del pulsante di reset presente sulla scheda Arduino. Vediamo ora come mettere in funzione questo shield: per pri-



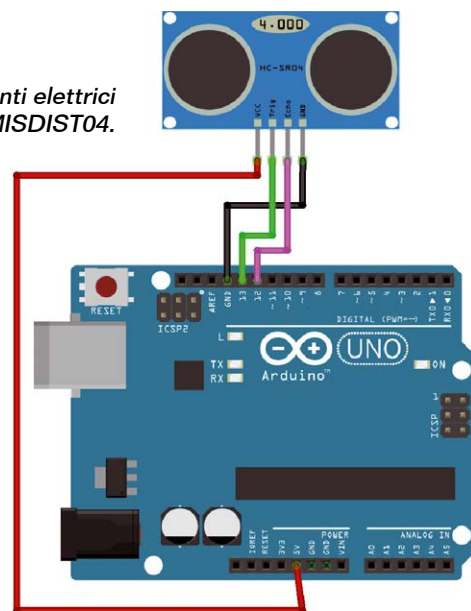
Shield e smartphone: l'accoppiata vincente di 1Sheeld.

ma cosa occorre recarsi sul sito di riferimento <http://1sheeld.com/> nel quale è disponibile una sezione dedicata ai tutorial ed una sezione dedicata ai progetti della comunità. La prima operazione consiste nello scaricare la libreria per Arduino, la quale si può trovare alla voce *Download* ma è altresì possibile utilizzare l'apposito tool fornito a corredo dell'ultima versione dell'IDE di Arduino. Avviate l'IDE di Arduino (versione 1.6.7 o più recente) ed andate nella sezione *sketch>inclusiones librerie>gestione libreria* quindi selezionate *onesheeld* ed installatela (Fig. 2); tutti gli sketch di questo articolo fanno riferimento alla versione 1.8.0 della libreria.

Potete subito prendere confidenza con lo shield provando i numerosi sketch già disponibili negli esempi. Lo shield viene fornito già assemblato, è comunque possibile autocostruirlo scaricando dal nostro sito www.elettronica.in le relative tracce lato rame. Il montaggio comunque richiede perizia, un saldatore a punta finissima e del filo di lega saldante di diametro non superiore a 0,5 mm, oltre a una lente d'ingrandimento e delle pinzette per posizionare i componenti più piccoli.

I primi componenti da montare sono resistenze e condensatori, cui seguono i diodi al silicio e i LED, poi il microcontrollore; si passa quindi al MOSFET e al modulo a quarzo, al regolatore e agli integrati IC1 e IC2. Si procede quindi con gli elementi restanti, ricordando che il modulo Bluetooth prende posto sul lato inferiore dello shield. Il montaggio termina con i pin-strip standard

Fig. 3 - Collegamenti elettrici del sensore a ultrasuoni MISDIST04.



Arduino, che devono avere i piedini lunghi 20 mm per innestare lo shield.

Per tutte le fasi della realizzazione tenete d'occhio i piani di montaggio pubblicati in queste pagine, che indicano l'orientamento dei semiconduttori.

IL NOSTRO PROGETTO

Per questo articolo proponiamo invece un'applicazione di utilità generale, che consiste in un contapersone. Se non sapete di cosa si tratta, vi diciamo che è un sistema in grado di rilevare sia il passaggio delle persone da un varco, sia la direzione in cui avviene; in questo modo è possibile sia avere un'indicazione visiva o acustica del passaggio, sia contare distintamente chi entra e chi

RADAR A ULTRASUONI

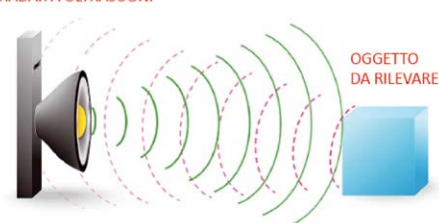


Fig. 4 - Rilevamento mediante un sensore di distanza a ultrasuoni.

Listato 1 - Parte essenziale per il conteggio delle persone in transito.

```
inUS();
if ( currentDistance>10 && currentDistance < maxDistance)
{
    if (previousDistance-currentDistance > triggerDistance && millis()-triggerTime > deadTime)
    {
        triggerTime=millis();
        count++;
        -> gestione 1sheeld <-
    }
    previousDistance=currentDistance;
}
```

esce così da verificare l'eventuale differenza e da capire se qualcuno è rimasto all'interno di un esercizio commerciale all'orario di chiusura.

Dunque la nostra applicazione può essere utile a un negoziante ad avere una segnalazione acustica dallo smartphone che tiene con sé per sapere se è entrato qualcuno mentre è indaffarato sul retro; ma può anche effettuare un semplice conteggio delle persone che entrano durante una fiera o un evento, il tutto su un dispositivo che non bisogna ricordarsi di prendere o aggiungere alla dotazione, perché lo

abbiamo sempre con noi.

Abbiamo pensato anche ad un impiego più casalingo, potendo usare il nostro progetto per sapere se qualcuno entra in una determinata stanza oppure se vostro figlio ha oltrepassato una zona proibita, gli impieghi sono veramente molteplici.

Comunque venga utilizzato, la parte più interessante del progetto è che l'informazione relativa al transito di una persona viene fornita direttamente sul vostro smartphone, con tanto di indicazione del numero di persone che sono passate e di avviso sonoro e vibrazione.

Abbiamo previsto due varianti a seconda che vogliate conoscere solo il numero di persone in transito oppure tenere conto di chi entra e di chi esce. Supponiamo pertanto che ci sia un varco di entrata e noi dobbiamo contare le persone che transitano, potremmo usare delle barriere fotoelettriche con la necessità di cablare trasmettitore e ricevitore da una parte all'altra del passaggio preferiamo, invece in questo progetto, utilizzare dei semplici (ed economici) sensori ad ultrasuoni in grado di identificare il passaggio di una persona dalla misura di distanza.

Come sensore abbiamo utilizzato l'economico MISDI-ST04 della Futura Elettronica (www.futurashop.it) che è un metro ad ultrasuoni preciso al centimetro sino alla distanza di qualche metro. Mettendo questo sensore sullo stipite di una porta rivolto verso lo stipite opposto, dalla misura fornita dal sensore

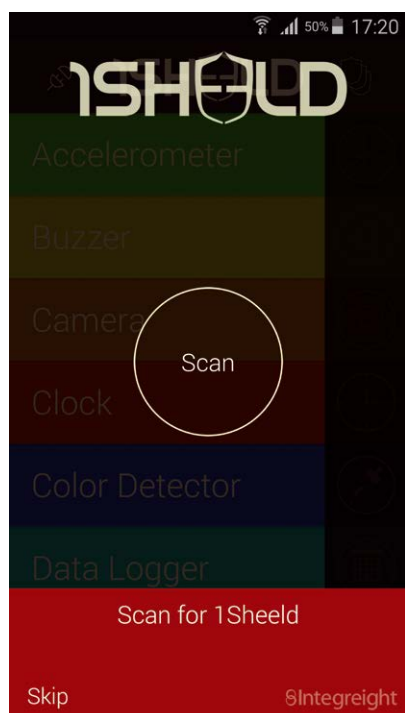


Fig. 5 - App di ricerca.

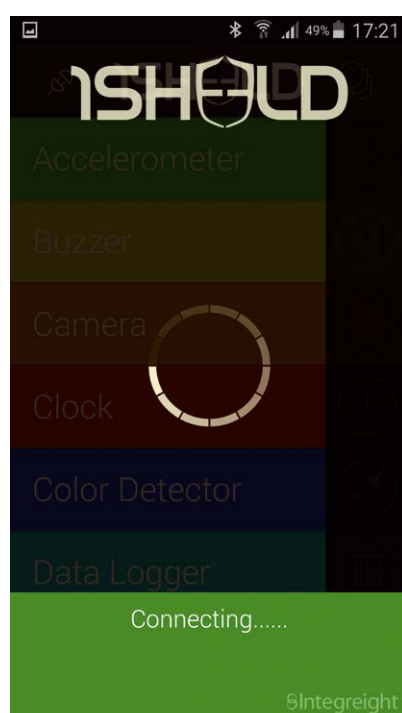


Fig. 6 - Connessione a 1Sheeld.

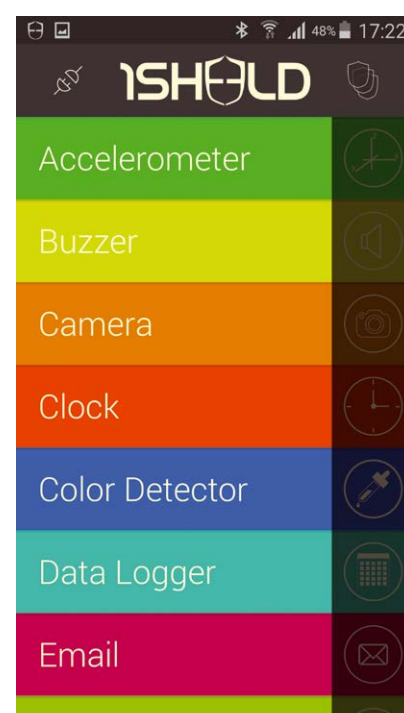


Fig. 7 - Schermata principale di 1Sheeld

Listato 2 - Definizioni per l'uso del 1Sheeld.

```
#define CUSTOM_SETTINGS
#define INCLUDE_VIBRATION_
#define INCLUDE_LCD_
#define INCLUDE_BUZZER_
#define INCLUDE_PUSH_BUTTON_
#define INCLUDE_NOTIFICATION_

/* Include 1Sheeld library. */
#include <OneSheeld.h>
```

Listato 3 - Codice per dialogare con lo smartphone.

```
Buzzer.buzzOn();
Notification.notifyPhone("Someone pass!");
LCD.clear();
String stringOne = String(count, DEC);
String dataText = "Count= " + stringOne;
char charBuf[16];
dataText.toCharArray(charBuf, 16);
LCD.print(charBuf);
delay(50);
Buzzer.buzzOff();
Vibration.start(1000);
```

potremmo valutare se si è verificato il passaggio di una persona, oppure di un oggetto.

In Fig. 3 trovate i collegamenti elettrici da effettuare; lo schema di principio è quello in Fig. 4. Per l'alimentazione dell'insieme consigliamo di usare un piccolo alimentatore switching ad alto rendimento con uscita a 5V ed attacco USB type B maschio adatto alla scheda Arduino. È possibile alimentare Arduino anche tramite il suo connettore plug a patto di usare un alimentatore con una tensione compresa tra 7 e 12 volt meglio di tipo switching per un migliore rendimento come un

alimentatore switching a 12 V disponibile presso la Futurashop (www.futurashop.it) con codice 8822-AL12V1A2.

La tecnica che abbiamo utilizzato per questa applicazione consiste nel leggere a cadenza regolare la misura fornita dal sensore e confrontarla con la misura precedente: se la misura fornisce un valore sensibilmente ridotto significa che un oggetto è passato di fronte al sensore. Per maggiore sicurezza scarteremo tutte le misure considerate inattendibili, come quelle inferiori ai 10 cm e superiori ai 150 cm; inoltre inibiremo letture successive per un tempo

prefissato, dopo aver rilevato il passaggio di una persona. Le righe di codice fondamentali in questo caso sono quelle che riportano le impostazioni dei parametri appena descritti, ossia i parametri di funzionamento dello sketch:

```
long maxDistance=150;
long triggerDistance=50;
long deadTime=1000;
```

dove *maxDistance* rappresenta la massima misura attendibile, oltre la quale la lettura viene scartata, mentre il parametro *triggerDistance* indica la variazione di misura oltre la quale conside-

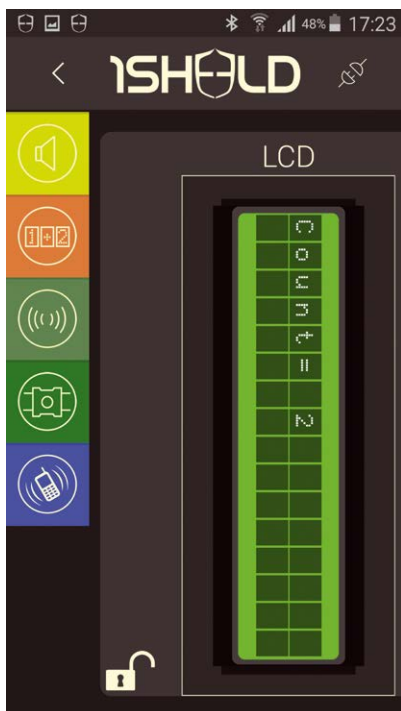


Fig. 8 - Visualizzazione LCD.

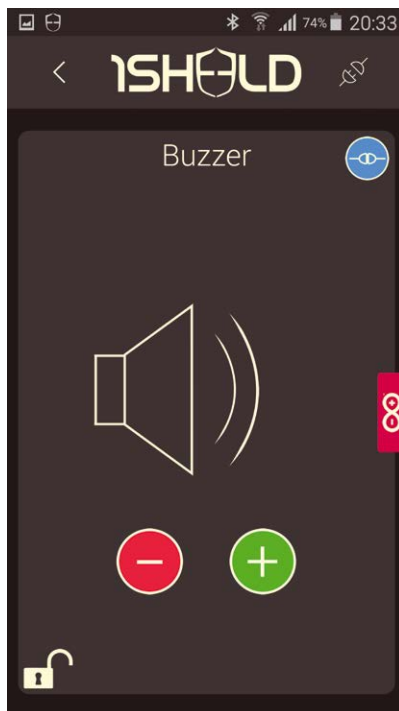


Fig. 9 - Visualizzazione BUZZER

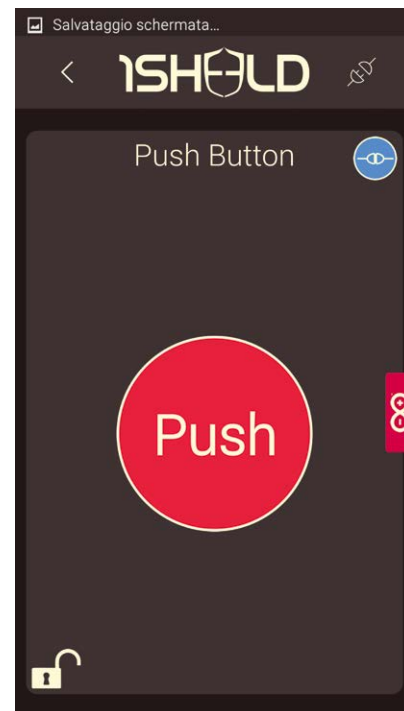


Fig. 10 - Visualizzazione BUTTON.

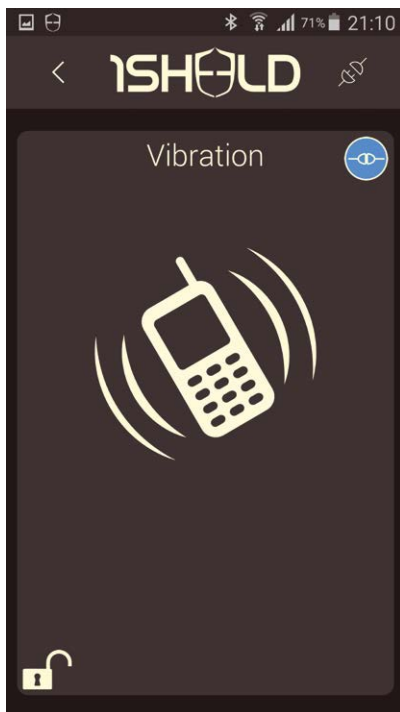


Fig. 11 - Visualizzazione VIBRATION.

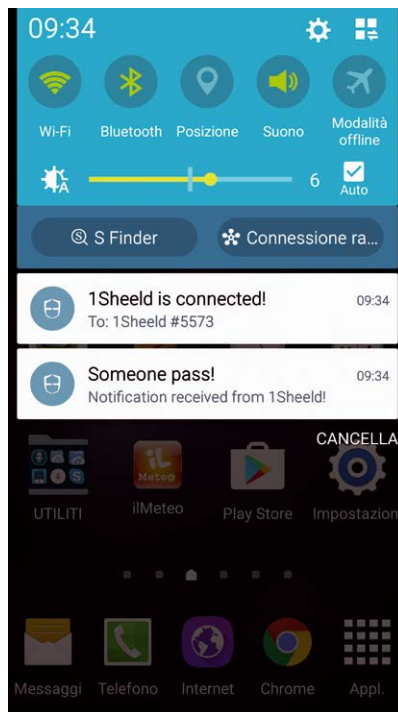


Fig. 12 - Visualizzazione delle notifiche.

riamo certo il passaggio di una persona; valori più bassi permettono una sensibilità maggiore ma la misura risulterà più alterabile dai disturbi.

Il parametro *deadTime* è il tempo di inibizione tra una lettura e la successiva, utile per evitare che fluttuazioni della misura di distanza possano comportare errori nel rilevamento del passaggio della persona. La parte di programma cui è demandato il conteggio delle persone in transito è riportata nel **Listato 1**.

La funzione *inUS()* viene usata per leggere la misura di distanza dal sensore; il valore corrispondente viene posto nella variabile *currentDistance*. Solo quando viene rilevato il passaggio di una persona, viene incrementata la variabile *count*. Durante le nostre prove, per visualizzare questo valore sul PC abbiamo utilizzato una porta seriale virtuale implementata con la libreria *softSerial* i cui dati venivano inviati al PC tramite un convertitore USB/seriale; questo perché con 1Sheeld

attivo il modulo UART di Arduino non può essere utilizzato per comunicare con il Personal Computer.

Veniamo ora alla parte più interessante, ovvero come integrare

in questo progetto le funzionalità di 1Sheeld. Per rendere operativo il tutto dobbiamo lavorare dal lato smartphone e per prima cosa installiamo l'applicazione ufficiale 1Sheeld accedendo direttamente al play store Android.

La app può anche essere scaricata nel dispositivo dal browser web accedendo alla pagina <http://1sheeld.com/downloads/> dalla quale è possibile ottenere la versione Android e le librerie.

Abilitiamo dunque il Bluetooth dello smartphone (se già non è abilitato) ed avviamo l'app: per prima cosa ci verrà chiesto di rilevare 1Sheeld (la Fig. 5 mostra l'app di ricerca dei dispositivi 1Sheeld via Bluetooth); seguirà la procedura di connessione (la Fig. 6 illustra la schermata a connessione avvenuta). A questo punto è possibile scegliere gli shield virtuali adatti alla nostra applicazione (Fig. 7): siccome vogliamo vedere il conteggio sul display dello smartphone, ci serve un LCD Shield (Fig. 8); desideriamo

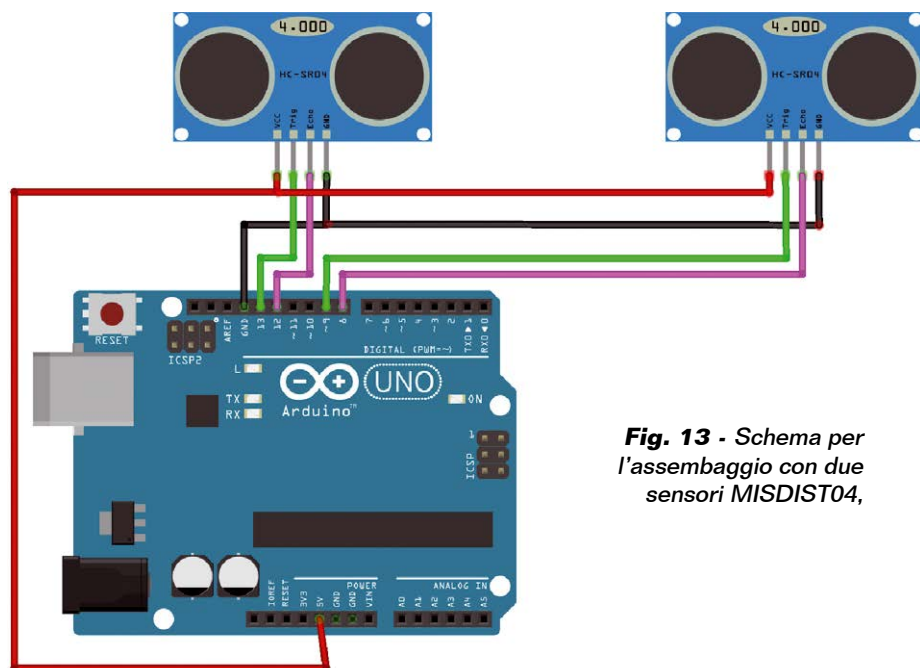


Fig. 13 - Schema per l'assemblaggio con due sensori MISDIST04,

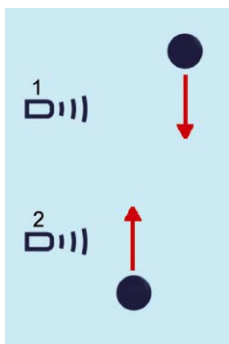


Fig. 14
Rilevamento
passaggio
con due
sensori.

inoltre un avviso acustico ad ogni passaggio, quindi usiamo un Buzzer Shield (Fig. 9). Poi vogliamo poter resettare il conteggio manualmente, quindi ci viene utile usare un Button Shield (Fig. 10); per finire, implementiamo una lieve vibrazione, per cui selezioniamo un Vibration Shield (Fig. 11).

Nel caso volessimo utilizzare il sistema per rilevare intrusioni o uscite non permesse, abbiamo previsto anche la possibilità di inviare una notifica allo smartphone utilizzando -allo scopo- un Notification Shield (Fig. 12). Nel Listato 2 riportiamo le righe di codice che provvedo ad includere la libreria 1Sheeld e ad

abilitare le funzionalità richieste. Come suggerito dagli stessi sviluppatori, è consigliato abilitare (con l'istruzione #include) solo gli shield effettivamente utilizzati, al fine di risparmiare memoria programma. Nel Listato 1, alla voce "gestione 1Sheeld" inseriremo le righe di programma che permettono di interagire con lo smartphone.

Come visibile nel Listato 3, le righe che permettono di inviare i dati allo smartphone sono semplicissime e consistono semplicemente nel richiamare la funzione desiderata.

Per visualizzare il valore numerico è tuttavia necessario convertirlo in un array di char; per il resto le istruzioni sono molto intuitive. In base alle necessità, alcune funzioni possono essere omesse, ad esempio la notifica ha senso solo nel caso di passaggi sporadici ma di una certa importanza, ha però il vantaggio di conoscere l'ora esatta in cui è avvenuto il passaggio. La gestione dell'azzeramento

del conteggio viene operata dalle seguenti righe di codice:

```
if (PushButton.isPressed())
{
    count=0;
    LCD.clear();
    LCD.print("Count= 0");
}
```

Come vedete, la gestione dell'azzeramento del conteggio avviene leggendo lo stato del pulsante implementato nello smartphone.

INIZIAMO A LAVORARE

Bene, stabilito come funziona il firmware e in che modo si opera dall'app, entriamo nel vivo del progetto: alla schermata principale dell'applicazione ci è sufficienteappare sull'icona a sinistra per includere lo shield nel progetto; cliccando sull'icona dello scudo in alto a destra si avvierà la comunicazione con lo shield ed il sistema è in funzione: veramente molto semplice ed intuitivo!

Nello shield, il LED di colore blu di nome BT-LED passerà da lam-

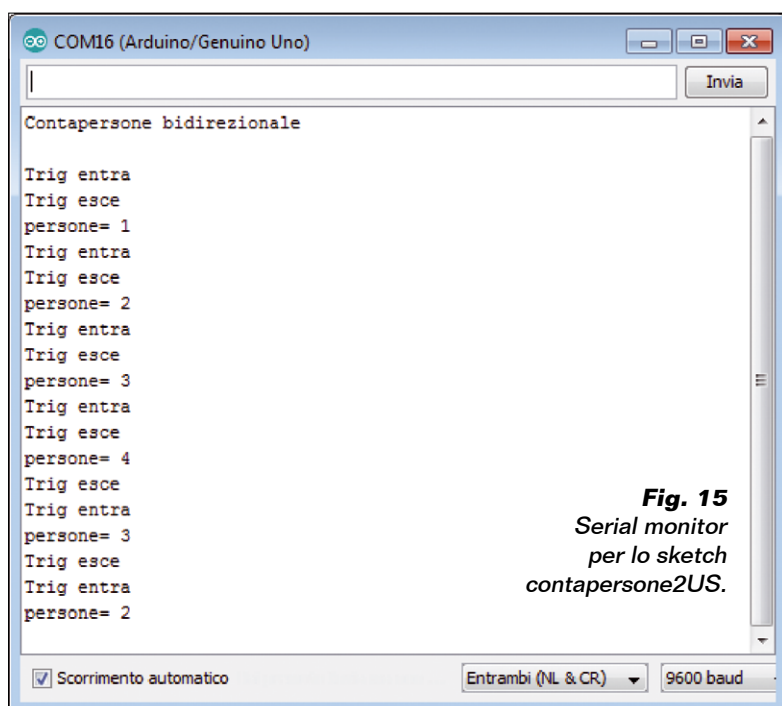


Fig. 15
Serial monitor
per lo sketch
contapersona2US.

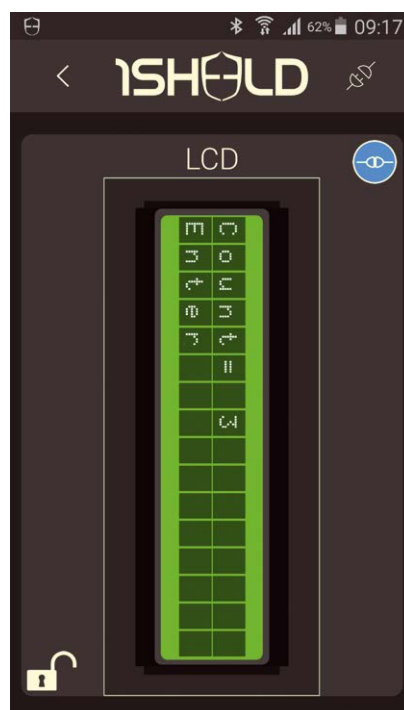
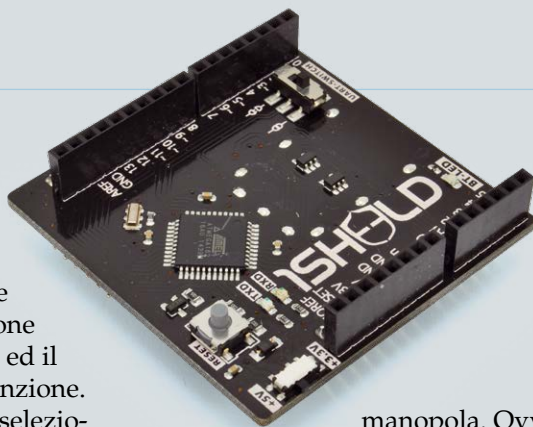


Fig. 16
Display
smartphone
per lo sketch
contapersona
2US.



peggiante a luce fissa ad indicare che la comunicazione è stata attivata ed il sistema è in funzione. Sarà possibile selezionare con uno swipe una volta lo shield desiderato, passando dalla visualizzazione del conteggio alla regolazione del volume del buzzer, sino ad arrivare alla schermata con il pulsante per l'azzeramento del conteggio.

RILEVARE IL VERSO DEL PASSAGGIO

Per specifiche applicazioni abbiamo pensato a una variante del sistema che prevede l'utilizzo di due sensori ad ultrasuoni posti ad una distanza di circa 25 cm l'uno dall'altro (Fig. 13). In questo modo, al passaggio di una persona sarà attivato prima un sensore e successivamente il secondo, con una sequenza che dipende dal verso di percorrenza; così possiamo capire se la persona sta entrando oppure uscendo. In altre parole, i sensori ad ultrasuoni rileveranno un aumento e una diminuzione della distanza dal corpo della persona in sequenza.

Facendo riferimento alla Fig. 14 possiamo dire che una persona che entra incontrerà prima il sensore 1 e successivamente il sensore 2, viceversa prima il sensore 2 e poi il sensore 1 se sta uscendo. È sufficiente controllare entrambi i sensori e memorizzare quale dei due ha fornito per primo il segnale, quindi attendere il segnale dal secondo sensore per determinare se la persona è entrata o uscita; il principio di funzionamento è molto simile a quello utilizzato nella lettura degli encoder rotativi per determinare il verso di rotazione della

manopola. Ovviamente se la seconda lettura non arriva entro un intervallo di tempo prestabilito, il sistema considera che la persona non ha attraversato completamente il passaggio e quindi non avviene il conteggio. Lo sketch scritto per questa variante del sistema si chiama *contapersone2US.ino* e la visualizzazione su Serial Monitor dei dati di debug è visibile in Fig. 15; il risultato sul display dello smartphone è visibile nella Fig. 16, che ci mostra la rappresentazione del display virtuale alfanumerico implementato dall'LLC Shield.

In quest'ultimo caso, dato che l'abbiamo implementata, sarà indicata anche la direzione di passaggio, ovvero se il soggetto sta entrando o uscendo; il conteggio sarà incrementato se la persona è in entrata e decrementato se la persona rilevata è in uscita. Posizionando un sistema siffatto all'ingresso di un ambiente potremmo conoscere il numero di persone al suo interno, ovviamente con le opportune limitazioni dovute, ad esempio, al sovrapporsi di più persone o a disturbi di varia natura dovuti allo svolazzare dei vestiti causato da eventuali correnti d'aria o alla presenza in luogo di una forte ventilazione. Infatti il sistema non può rilevare l'attraversamento simultaneo di più persone, nel senso che se non c'è spazio tra quella più avanzata e quella più arretrata viene rilevato un solo passaggio.

CONCLUSIONI

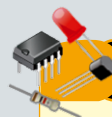
Si conclude qui questa nostra

trattazione riguardante 1Sheeld ed un paio delle applicazioni che esso rende possibile.

Crediamo di avervi fatto comprendere, in queste pagine, le potenzialità e le caratteristiche del prodotto tanto che ormai vi sarà facile immaginare quali applicazioni potete mettere in pratica sfruttando gli innumerevoli shield virtuali a disposizione, soprattutto quelle difficilmente implementabili con un classico shield hardware per Arduino, come ad esempio l'invio di messaggi attraverso i social network (che richiederebbe un WiFi shield e tanto altro hardware) oppure il riconoscimento vocale. Con 1Sheeld è invece immediato utilizzare un Voice Recognition Shield per identificare comandi vocali, piuttosto che un Camera Shield per acquisire filmati e analizzarli.

Insomma, sulla base di quel che vi abbiamo spiegato potete sperimentare da voi le possibilità offerte da un portafoglio di soluzioni che conta ben 40 shield virtuali.

Buon lavoro!



per il MATERIALE

Lo shield Arduino per smartphone Android (cod. 1SHEELD) è disponibile presso Futura Elettronica al prezzo di Euro 69,00 così come il sensore di distanza ad ultrasuoni (cod. MISDIST04) in vendita a Euro 6,00. L'applicazione, la libreria, la documentazione e il tutorial relativi allo shield sono disponibili nella scheda del prodotto sul sito www.futurashop.it.

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

Rivisitazione della popolare Arduino Nano, rispetto alla quale implementa in più l'architettura della nostra Fishino con tanto di connettività WiFi, caricabatteria ecc.

GUPPY: FISHINO SI FA NANO

di MASSIMO DEL FEDELE

Come vi abbiamo anticipato nell'articolo in cui abbiamo presentato la nostra Fishino MEGA (scheda compatibile con Arduino MEGA, contraddistinta da un gran numero di I/O e molta memoria di programma e/o RAM rispetto alla versione UNO) abbiamo sviluppato una board che può essere considerata l'altra faccia della medaglia, in quanto è la versione più essenziale della "saga" Fishino. Si tratta della Fishino Guppy, il cui nome deriva da un pesciolino ben conosciuto agli appassionati di acquari, che abbiamo voluto assegnarle in virtù delle ridottissime dimensioni della scheda, che è la più piccola Fishino. Malgrado sia la versione più miniaturizzata (come la Nano è per Arduino) la nostra piccola Guppy presenta funzionalità e caratteristiche di tutto rispetto (come potete leggere riquadro a pagina seguente). Le periferiche incluse nella board sono quelle delle Fishino UNO e MEGA, eccetto che per il modulo RTC, che fisicamente non

trova posto nel PCB che è stato voluto, in sintonia con la filosofia della Arduino Nano, il più piccolo possibile. Rispetto alla Fishino UNO, però, abbiamo aggiunto tutto lo stadio d'alimentazione in tecnologia switching e la possibilità di alimentare la board con una batteria ai polimeri di litio, cosa estremamente interessante perché si presta a realizzare progetti portatili, se non, addirittura, indossabili.

SCHEMA ELETTRICO

Passiamo ora alla descrizione dello schema elettrico, che si può considerare diviso nei seguenti blocchi:

- stadio di alimentazione;
- stadio caricabatteria;
- interfaccia USB;
- controller Atmega328p;
- adattatori di livello;
- modulo WiFi;
- connettore per microSD-Card.

CARATTERISTICHE TECNICHE

- 100% compatibile con Arduino Nano
- Scheda WiFi a bordo
- Connettore per microSD a bordo
- Tripla alimentazione: batteria LiPo, porta USB e connettore di alimentazione esterno (6,5÷20 volt)
- Stadio di alimentazione switching ad elevata efficienza, con step-up per ottenere i 5 volt dalla batteria e step-down per ricavarli dall'ingresso esterno,
- Sezione 3,3 volt potenziata per gestire il modulo WiFi, la microSD ed eventuali ulteriori componenti esterni
- Caricabatteria a bordo con indicazione dello stato di carica

SCHEMA ELETTRICO

Siccome, eccezion fatta per i primi due blocchi (lo stadio di alimentazione e il caricabatteria), lo schema elettrico della Guppy è molto simile, quasi identico a quello già presentato per la board Fishino UNO, ci limiteremo a descrivere in dettaglio solo i primi due stadi e daremo una spiegazione sommaria dei restanti, evidenziando le differenze rispetto alle precedenti schede. Per gli approfondimenti del caso rimandiamo all'articolo dove abbiamo descritto Fishino UNO (fascicolo n° 198 di Elettronica In). Lo schema elettrico cui faremo riferimento in questa descrizione è illustrato nella pagina qui accanto.

L'alimentazione

A differenza di quanto fatto nella Fishino Mega, nella Guppy non abbiamo utilizzato uno stadio Sepic, il quale, pur permettendoci di risparmiare un componente e di poter alimentare il circuito sul pin VIN con una tensione minima di 3,5 volt, si è rivelato troppo instabile a causa delle dimensioni ridotte della scheda.

Nel primissimo prototipo avevamo infatti utilizzato tale tipologia, unita ad un PCB a due layer; il tutto funzionava perfettamente finché non veniva utilizzato il modulo WiFi, dopodiché i disturbi provocati da questo "rientravano" nel feedback dell'alimentazione provocando instabilità. Siamo quindi ritornati sui nostri

passi optando per un doppio converter: uno step-up per l'ingresso batteria ed uno step-down per l'alimentazione dall'ingresso VIN. Per evitare l'instabilità possibile comunque anche con queste tipologie, abbiamo poi optato per un PCB a quattro layer, con i piani di massa ed alimentazione all'interno, in modo da creare una barriera che impedisce ai forti segnali impulsivi di trasmettere le proprie interferenze attraverso il circuito stampato stesso.

Con questi accorgimenti la nostra Guppy è quindi diventata estremamente stabile, anche se il prezzo da pagare è che non è possibile autocostruire la scheda con metodi artigianali; pubblichiamo comunque schema elettrico e piano di montaggio per consentirvi di conoscere a fondo l'hardware della board.

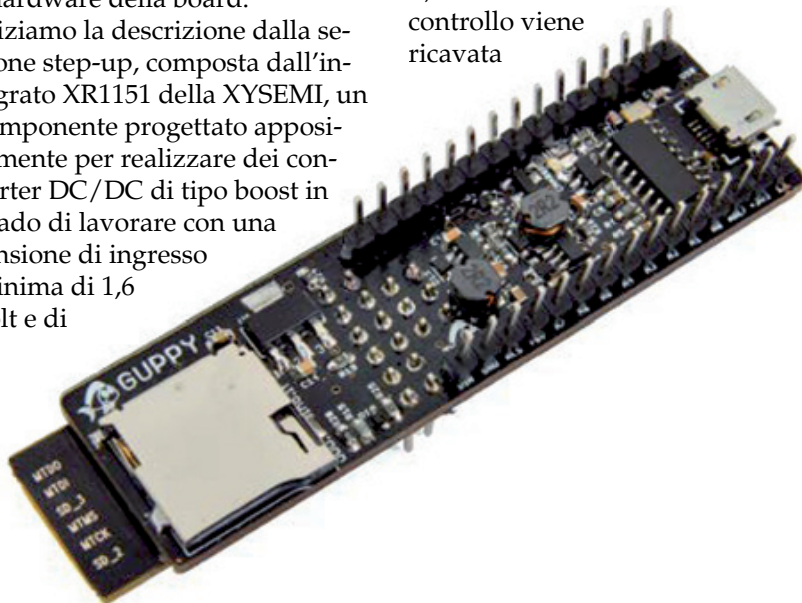
Iniziamo la descrizione dalla sezione step-up, composta dall'integrato XR1151 della XYSEMI, un componente progettato appositamente per realizzare dei converter DC/DC di tipo boost in grado di lavorare con una tensione di ingresso minima di 1,6 volt e di

fornire in uscita 5 volt con 800 mA di corrente, se alimentato con 3,3 volt in ingresso. È quindi ottimale per accettare l'alimentazione da una comune batteria LiPo 1s (3,7 volt, quindi a singola cella).

L'integrato contiene al suo interno tutti gli stadi necessari: un generatore PWM, un interruttore CMOS con una resistenza $R_{ds(on)}$ estremamente bassa (tipicamente 100 milliohm), la necessaria circuiteria di retroazione (feedback) per mantenere stabile la tensione in uscita ed un sistema di avvio progressivo (soft-start) per evitare picchi di corrente eccessivi nei componenti demandati alla commutazione della corrente d'uscita.

L'integrato lavora ad una frequenza tipica di 1,2 MHz e quindi consente l'utilizzo di componenti (induttanza e condensatori) di dimensioni estremamente contenuti. Opera in modo asincrono, quindi richiede un diodo esterno di commutazione (D12) che nel nostro caso è uno Schottky con bassissima tensione diretta per non penalizzare l'efficienza.

Il circuito di retroazione è calibrato per una tensione di 0,6 V; la tensione di controllo viene ricavata



dal partitore costituito dalle resistenze di precisione R29 ed R30 ed applicata nell'ingresso FB. La differenza di potenziale in uscita viene stabilizzata sul valore dato dalla formula:

$$V_{out} = 0.6V \cdot \left(1 + \frac{R_{29}}{R_{30}}\right) = 0.6V \cdot \left(1 + \frac{97.6K}{13.3K}\right) = 5.003V$$

La tensione viene riportata direttamente alla linea dei 5V senza alcun diodo aggiuntivo, essendo già presente il diodo D8 e l'interruttore interno all'integrato, disattivabile tramite l'ingresso EN dello stesso, sul quale ritorneremo più avanti.

Una nota: l'XR1151 è fatto per lavorare con tensioni molto basse, e quindi si danneggia se all'ingresso vengono applicati più di 6 volt; questo lo differenzia dal componente utilizzato nello stadio SEPIC della Fishino Mega, che in ingresso accettava tensioni ben più elevate. È quindi assolutamente da evitare l'applicazione di tensioni superiori ai 5÷5,5 V all'ingresso batteria, pena la distruzione del componente e di ciò che viene alimentato da esso. Proseguiamo con la sezione step-down, che ruota attorno all'integrato AOZ1280 dell'Alpha&Omega Semiconductor; questo componente accetta in ingresso tensioni da 3 volt fino a 26 volt ed è in grado di fornire un'uscita minima di 0,8 volt. La frequenza di lavoro è elevatissima, 1,5 MHz, quindi anche qui possiamo utilizzare componenti di dimensioni estremamente ridotte. Come nel caso visto per lo step-up, l'AOZ1280 contiene tutti i componenti necessari per realizzare un buck converter asincrono, ma richiede un diodo esterno di blocco, siglato D11 nello schema elettrico.

L'integrato dispone di un ingresso di compensazione che, tramite il condensatore C12 da 10 nF, permette di stabilizzare la rete di

retroazione in tutte le condizioni di carico.

L'ingresso EN (enable) in questo caso non viene utilizzato, ragion per cui lo abbiamo collegato direttamente alla tensione in ingresso, il che abilita il convertitore non appena viene alimentato.

L'ingresso di retroazione (FB, ossia feedback) è in questo caso calcolato per una tensione di riferimento interna pari a 0,8 volt; la tensione in uscita risulta quindi pari a:

$$V_{out} = 0.8V \cdot \left(1 + \frac{R_{31}}{R_{32}}\right) = 0.8V \cdot \left(1 + \frac{68.1K}{12.1K}\right) = 5.3V$$

La Vout è stata volutamente scelta leggermente superiore ai 5 volt per compensare la caduta sul diodo Schottky in uscita D10, che si aggira sugli 0,2÷0,3 volt; tale diodo è indispensabile per evitare che la tensione proveniente dallo step-up rientri nel componente e vada ad influire sull'ingresso VIN.

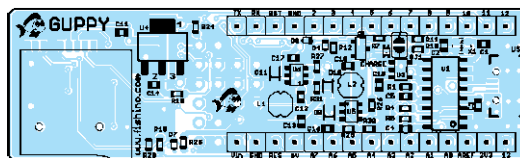
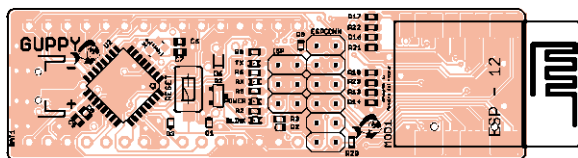
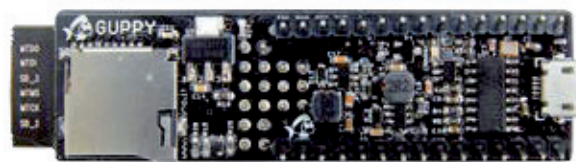
Detto ciò, andiamo ad analizzare la sezione di commutazione degli ingressi di alimentazione, differente rispetto allo schema delle altre Fishino; essa ruota attorno ai due diodi Schottky D4 e D5, al MOSFET a canale N Q1 e alle resistenze R12 e R26. In assenza di tensione sugli ingressi VIN o USBVCC, il gate di Q1 viene mantenuto a massa tramite la resistenza R12, spegnendo quindi il componente. La sua uscita si trova quindi libera di fluttuare ed è mantenuta a livello prossimo all'alimentazione dall'integrato caricabatteria (che vedremo in seguito) U3; l'ingresso EN (enable) dell'XR1151 verrà quindi a trovarsi a livello alto (tramite la resistenza R26) e quindi entrerà in funzione. L'uscita a 5 volt non potrà rientrare sui pin VIN o USBVCC a causa della presenza dei diodi D6 e D10.

Applicando, per contro, una tensione su uno degli ingressi VIN o USBVCC, il MOSFET Q1 verrà polarizzato positivamente attraverso il diodo D4 o D5 e quindi entrerà in conduzione, portando a massa l'ingresso EN dell'XR1151 spegnendolo.

Se la tensione all'ingresso VIN è minore di circa 6 volt, all'uscita del convertitore Step-Down sarà presente una tensione inferiore ai 5 volt, cosa che farà entrare in conduzione il diodo D6 attivando quindi l'alimentazione via USB; per contro, se sui VIN applichiamo una tensione sufficiente, l'uscita del buck converter AOZ1280 sarà sufficiente a portare il diodo D6 in interdizione e quindi disattivare anche l'alimentazione dalla porta USB. Abbiamo quindi ottenuto una commutazione completamente automatica delle tensioni di alimentazione, in questo modo: se è presente una tensione su VIN o sul plug USB, la batteria verrà sconnessa dal circuito e ne verrà contemporaneamente attivata la carica, come vedremo in seguito.

Se non è presente alcuna tensione esterna, entrerà in azione la batteria e, contemporaneamente, verrà disattivato il circuito di carica.

L'ultima parte che analizziamo, ma non ultima per importanza, è costituita dal regolatore a 3,3 volt che fornisce la tensione ai moduli WiFi e alla scheda microSD. Questa parte circuitale, per esigenze di spazio è stata realizzata tramite un consueto regolatore lineare, ma di capacità superiore a quello della Arduino Nano; è infatti in grado di erogare in uscita continuamente oltre 800 mA di corrente, contro i 500 mA di picco del regolatore di tensione originale della Nano.



R1, R2, R4, R5, R8, R18: 1 kohm (0603)
 R3, R10, R12, R14, R19, R24÷R26, R28: 10 kohm (0503)
 R6: 2,4 kohm (0603)
 R7, R9: 470 ohm (0603)
 R11: 2,7 kohm (0603)
 R13, R23: 3,3 kohm (0603)
 R15÷R17: 680 ohm (0603)
 R20÷R22: 1,5 kohm (0603)
 R27: 12,1 kohm (0603)
 R29: 97,6 kohm (0603)
 R30: 13,3 kohm (0603)
 R31: 68,1 kohm (0603)
 C1, C2, C6, C7: 22 pF ceramico (0603)
 C3, C5: 1 µF ceramico (0603)

C4, C8, C9, C11: 100 nF ceramico (0603)
 C10: 4,7 µF ceramico (0603)
 C12: 10 nF ceramico (0603)
 C13÷C17: 22 µF ceramico (0603)
 D1÷D5, D7, D9: RB521S SOD523
 D6, D8, D10, D11: PMEG3020EJ SOD323
 X1, X2: Quarzo 16 Mhz
 Q1, Q2: 2N7002 SOT23
 L1, L2: Bobina 2,2 µH CD32
 TX: LED Blu (0603)
 RX: LED Giallo (0603)
 BLINK: LED Bianco (0603)
 POWER: LED Verde (0603)
 CHARGE: LED Rosso (0603)
 U1: CH340G SOIC16

U2: ATMEGA328P-AU TQFP32-08
 U3: MCP73831 SOT23-5
 U4: NCP1117ST33 SOT223
 U5: XR1151 SOT23-6L
 U6: AOZ1280 SOT23-6L
 RESET: Microswitch
 SD1: Connettore micro-SD
 ESP: Modulo Wi-Fi ESP8266
 USB1: Connettore micro-USB
 BAT1: Connettore JST 2mm 2 vie

Varie:

- Strip maschio 2x3 vie
- Strip maschio 6 vie (2 pz.)
- Strip maschio 15 vie (2 pz.)
- Circuito stampato S1270

Stadio caricabatteria

Questo stadio ruota attorno al notissimo MCP73831 (U3) che abbiamo già trattato in vari progetti e che quindi non stiamo a rispiegare. L'unica parte degna di nota è il circuito di attivazione della carica, che fa capo ad alcuni componenti già visti nel paragrafo riguardante la commutazione dell'alimentazione: Q1, D4, D5 ed R12.

Abbiamo infatti già visto che quando è presente una tensione di alimentazione esterna su VIN o USBVCC, il MOSFET Q1 viene portato in conduzione e quindi il suo drain viene a trovarsi collegato a massa. Questo, oltre che spegnere lo step-up XR1151 come visto in precedenza, collega a massa la resistenza R10 (con eventualmente in parallelo la R11 se scelta la carica a 500 mA),

attivando così la ricarica.

Il diodo led rosso CHARGE si accende durante la carica e si spegne quando essa è completa, in assenza di batteria o quando il caricabatteria è disattivato.

INTERFACCIA USB

Lo stadio d'interfaccia USB della Guppy è identico a quello utilizzato a bordo delle schede Fishino UNO e Fishino Mega e gravita attorno all'ormai noto CH340G, un'integrato in grado di realizzare un completo convertitore USB/Seriale con tutti i relativi segnali di controllo, dei quali noi utilizziamo solo i canali dati (RX e TX) ed il segnale DTR per l'autoreset della Fishino Guppy quando viene programmata dal PC mediante l'IDE Arduino.

Per problemi di spazio, qui non

abbiamo previsto la possibilità di disattivare l'autoreset, funzione, peraltro, quasi mai utilizzata. Per una descrizione più approfondita di questo modulo rimandiamo all'articolo relativo dove abbiamo pubblicato il progetto della Fishino UNO dove è stato descritto in dettaglio (fascicolo n° 198).

MICROCONTROLLORE

Questo stadio risulta praticamente identico a quello della scheda Fishino UNO e utilizza lo stesso microcontrollore, ossia l'ATmega328P; differisce per l'utilizzo di due ingressi analogici aggiuntivi, vale a dire A5 ed A6, che non sono presenti nel layout della "UNO". Per il resto non sono state introdotte modifiche, quindi anche in questo caso qui rimaniamo alla descrizione dettagliata

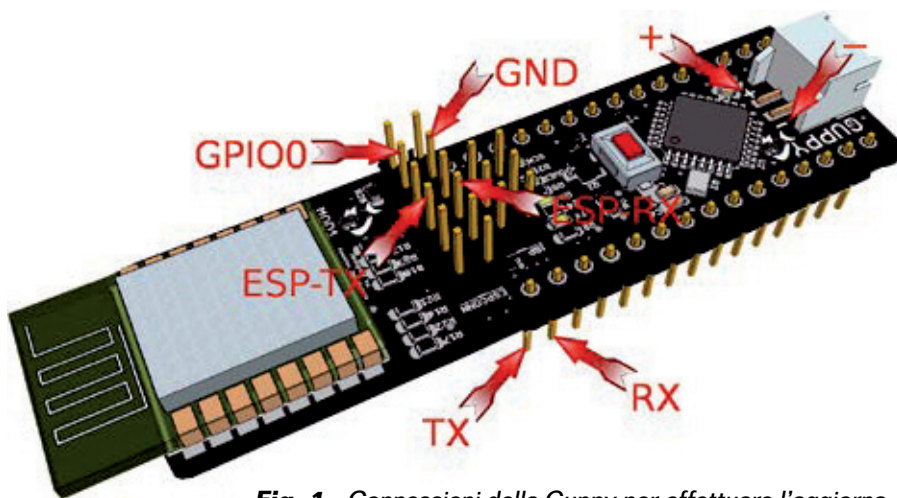


Fig. 1 - Connessioni della Guppy per effettuare l'aggiornamento del firmware.

reperibile nell'articolo descrittivo della Fishino UNO.

Una sola nota riguardante il bootloader : pur essendo questa scheda compatibile con l'Arduino Nano, abbiamo preferito installare il bootloader di Arduino UNO, per il semplice motivo che quest'ultimo occupa 1,5 kByte di Flash in meno e consente quindi di caricare sketch di dimensioni maggiori rispetto a quanto possibile con il bootloader della Nano. Pertanto è necessario, nell'IDE, selezionare come scheda Arduino UNO e non Arduino Nano, altrimenti l'IDE non riconoscerà il prodotto.

Nell'improbabile eventualità che si necessiti la piena compatibilità anche dal lato bootloader è comunque possibile caricare sul Guppy il bootloader originale di Arduino Nano.

Prestate attenzione alla piedinatura del connettore ISP, che nella Guppy è "rovesciata" rispetto a quella della UNO; ciò per seguire la piedinatura originale della Arduino Nano. Vi diciamo questo perché a chi scrive è capitato di "bruciare" una scheda Nano per aver collegato il connettore a rovescio, quindi, in caso di dubbio, vi consigliamo di consultare lo schema elettrico ed il layout del PCB prima di eseguire la programmazione del bootloader!

STADIO ADATTATORE DI LIVELLO

Anche in questo stadio non vi sono modifiche rispetto alle schede Fishino precedenti, fatto salvo il valore delle resistenze (da R13 a R23) che è dimezzato rispetto a quello delle corrispondenti resistenze di Fishino UNO e Fishino MEGA, per garantire una maggior immunità ai disturbi; tale requisito è stato imposto dalle dimensioni estremamente ridotte della scheda. Ad un'impedenza minore, infatti, corrisponde una maggior immunità ai disturbi esterni al costo di un piccolo aumento dei consumi durante il funzionamento del WiFi o della scheda microSD, da considerarsi comunque del tutto trascurabile. Per la descrizione dettagliata del funzionamento vi rimandiamo anche qui all'articolo relativo al Fishino UNO; qui indichiamo soltanto che gli stadi sono costituiti da partitori resistivi

che abbassano il valore massimo della tensione in uscita dall'Atmega dai 5 volt a circa 3,3 Volt, per non danneggiare i componenti a tensione più bassa; in senso inverso non occorre alcuna traslazione, essendo i 3,3 volt in grado di pilotare direttamente le logiche a 5 volt, dato che i pin sono 5V-tolerant.

MODULO WIFI

Come nella Fishino UNO e nella Fishino MEGA, la sezione di interfaccia wireless ruota attorno al modulo ESP12-F (l'ultima uscita della serie 12) basata sul chip WiFi ESP8266; la particolarità del modulo qui utilizzato è che ha un firmware studiato ad-hoc per comunicare con l'Atmega328 tramite interfaccia SPI, ben più performante della seriale che il firmware di fabbrica prevede di utilizzare per l'interfacciamento. A fine articolo riporteremo il procedimento di sostituzione di tale firmware con quello che occorre per il colloquio su SPI (la procedura è comunque presente anche sul sito web di riferimento del progetto, www.fishino.it). Maggiori dettagli sul modulo WiFi li trovate negli articoli dedicati a Fishino UNO e Fishino MEGA.



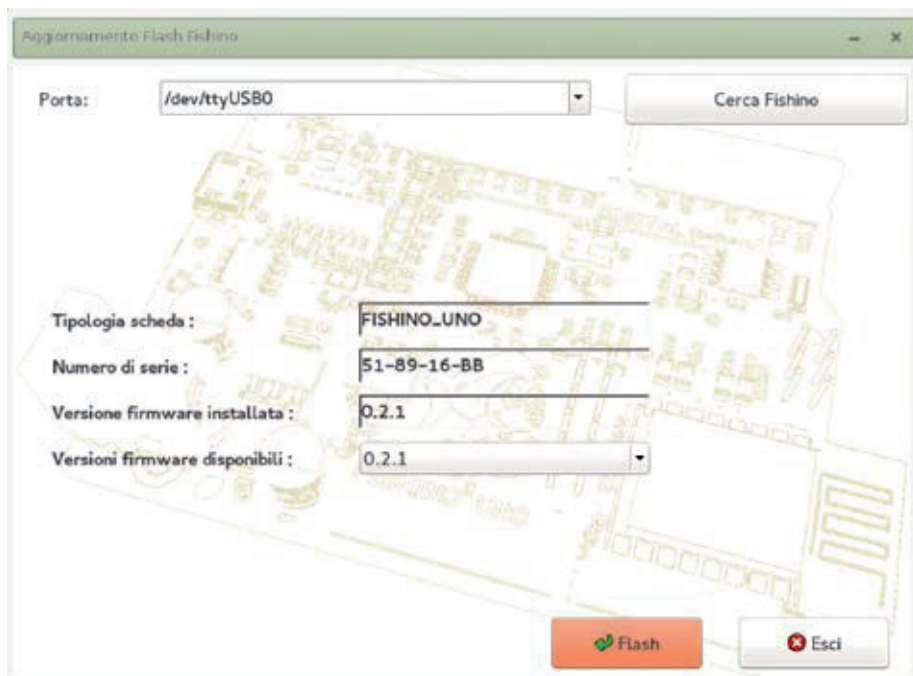


Fig. 2

CONNETTORE PER MICROSD

Anche questo stadio non è stato modificato rispetto alle Fishino MEGA e Fishino UNO, per cui rimandiamo chi volesse approfondirne il funzionamento agli articoli relativi a tali schede.

AGGIORNAMENTO DEL FIRMWARE DEL MODULO WIFI

Come per i modelli precedenti, suggeriamo di eseguire un aggiornamento del firmware al primo utilizzo e comunque periodicamente, perché è un programma in continuo sviluppo ed intimamente connesso con le librerie fornite. La procedura di aggiornamento è praticamente identica a quella dei precedenti modelli di Fishino; differiscono le connessioni da effettuare, che sono quelle illustrate nella **Fig. 1**. La procedura di aggiornamento è semplificata da un programma apposito, disponibile sia per la piattaforma Windows che per Linux, che esegue l'operazione in modo completamente automatico ed a prova di errore. I passi da compiere per l'aggiornamento sono i seguenti.

1. Caricare uno sketch che non

utilizzi la porta seriale. L'esempio base BLINK (quello che fa lampeggiare il led sulla scheda) è perfetto; tale passaggio serve ad evitare interferenze tra lo sketch caricato ed il collegamento seriale tramite l'ATmega e l'ESP; se il programma di flash non rileva Fishino, al 99% il problema è uno sketch sbagliato caricato.

2. Connettere la porta TX di Fishino Guppy con la porta ESP-TX sul connettore ESP, e la porta RX della Fishino Guppy con la porta ESP-RX sul connettore ESP (vedere la **Fig. 1**).
3. Connettere la porta GPIO0 a massa tramite un cavetto o un ponticello sempre sul connettore ESP (vedere figura a fianco).
4. Collegare la Guppy al PC (o premere il pulsante di RESET, se già connessa).
5. Lanciare il programma **FishinoFlasher**, assicurandosi che il PC sia connesso ad Internet.

Se i collegamenti sono stati eseguiti correttamente, il programma rileverà la porta a cui è connessa la Guppy, determinerà il modello e la versione del

firmware attualmente installata, si collegherà ad un server remoto e scaricherà la lista dei firmware disponibili, mostrando l'ultimo ma permettendo comunque la selezione delle versioni precedenti (**Fig. 2**) nel caso si voglia fare un downgrade. Facendo clic sul pulsante "Flash" verrà avviata la procedura di aggiornamento alla fine della quale apparirà un messaggio di conferma. Per terminare il programma occorre fare clic sul pulsante "Esci".

Nel caso la Guppy non venga rilevata automaticamente, è possibile provare a selezionare la porta manualmente. È comunque probabile che siano stati commessi degli errori nei collegamenti.

La selezione manuale risulta indispensabile nel raro caso in cui più di una Fishino sia connessa contemporaneamente al PC, nel qual caso il primo viene rilevato automaticamente ma resta la possibilità di sceglierne un altro. Una volta terminata la procedura è sufficiente eliminare i tre collegamenti e **Fishino** sarà pronto per l'uso con il nuovo firmware.

Bene, abbiamo concluso: siete pronti per sviluppare le vostre applicazioni con la Guppy! ■



per il MATERIALE

La board Guppy (cod. Guppy) è in vendita presso Futura Elettronica al prezzo di Euro 33,90. Il prezzo si intende IVA compresa.

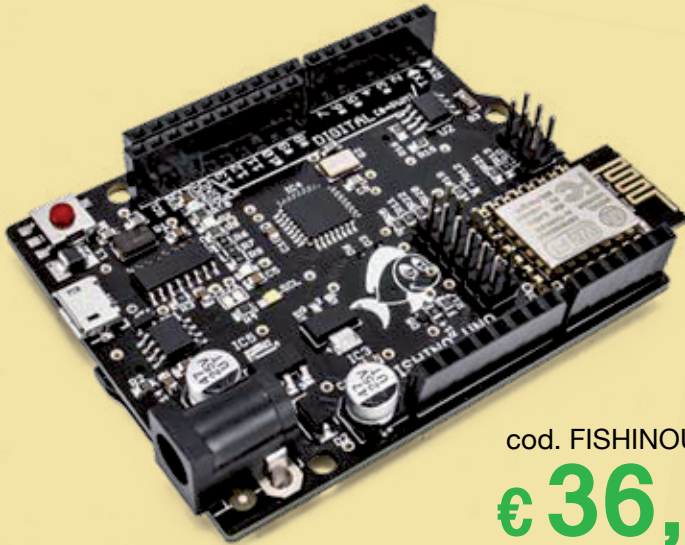
Il materiale va richiesto a:
Futura Elettronica, Via Adige 11,
21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
<http://www.futurashop.it>



Fishino

Design With Simplicity

I TUOI PROGETTI DIVENTANO WIRELESS



cod. FISHINO UNO

€ 36,⁰⁰

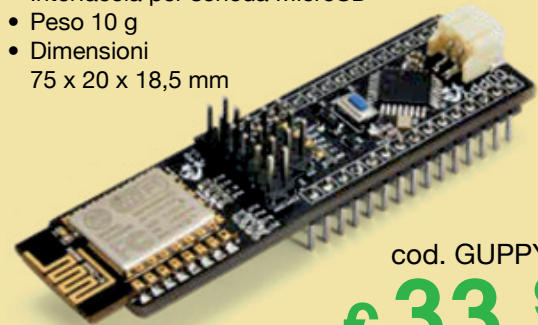
IVA inclusa

FISHINO UNO

- Alimentazione:
 - da 7 a 12 Vdc (tramite plug)
 - 5 Vdc (tramite porta USB)
- Compatibile al 100% con Arduino UNO
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Modulo RTC con batteria di mantenimento
- Sezione di alimentazione a 3,3 V potenziata
- Compatibile con shield e schede millefori
- Peso 25 g
- Dimensioni 76,5 x 53,5 x 14 mm

FISHINO GUPPY

- Alimentazione:
 - 3,6 Vdc (tramite batteria esterna al litio)
 - da 6,5 a 20 Vdc (tramite plug)
 - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile al 100% con Arduino NANO
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Peso 10 g
- Dimensioni 75 x 20 x 18,5 mm



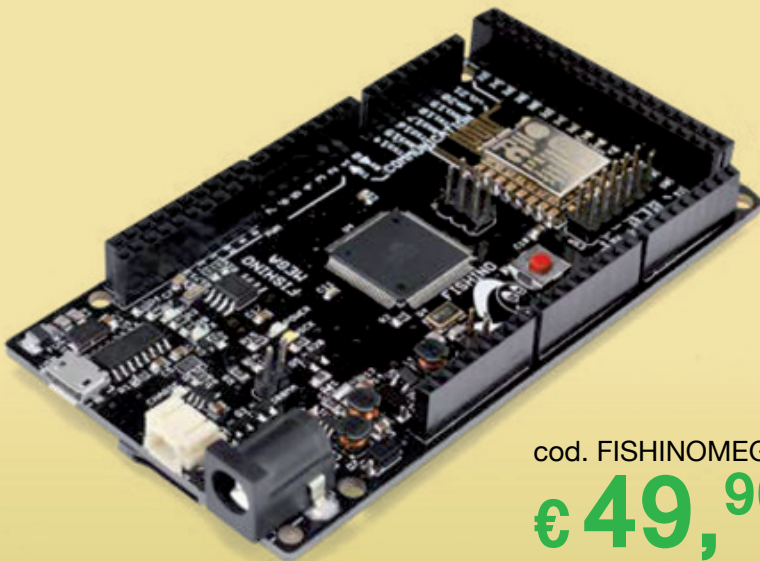
cod. GUPPY

€ 33,⁹⁰

IVA inclusa

FISHINO MEGA

- Alimentazione:
 - 3,6 Vdc (tramite batteria esterna al litio)
 - da 3,5 a 20 Vdc (tramite plug)
 - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile al 100% con Arduino MEGA
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Modulo RTC con batteria di mantenimento
- Stadio di alimentazione switching (5 Vdc e 3,3 Vdc)
- Compatibile con shield e schede millefori
- Peso 37 g
- Dimensioni 101,5 x 53,5 x 15 mm



cod. FISHINO MEGA

€ 49,⁹⁰

IVA inclusa



RILEVATORE DI PRESENZE

Sistema per il rilevamento e il conteggio delle persone in un ambiente, basato su Arduino e utile per applicazioni Smart come la gestione della climatizzazione e dell'illuminazione negli ambienti.

..... di ERICA VACCARI e FRANCESCO MARINO

Nei negozi e nei luoghi aperti al pubblico, specie in quelli in cui la conformazione (molte stanze, per esempio) rende difficile verificare al momento della chiusura se vi sia rimasta gente all'interno, vengono utilizzati sistemi contapersone collocati sul varco di accesso (tipicamente la porta di ingresso e uscita); si tratta di qualcosa in grado di rilevare i passaggi in entrambe le direzioni e da qui determinare, diciamo, il "saldo".

In questo articolo verrà presentato un sistema per il rilevamento e il conteggio delle persone in un

ambiente, con il fine di fornire questo dato a sistemi di supervisione ed anche per attuare una gestione energetica razionale e ottimizzata, tagliando tutti i flussi energetici inutili e mantenendo nella location lo stato di comfort ottimale per il numero di occupanti. In particolare, il sistema che descriveremo è parte del "Sistema multisensoriale intelligente per applicazioni smart" sviluppato dall'autore e da un team cui esso appartiene, del quale è già stata depositata domanda di brevetto (N.102016000091625). Nulla vieta, comunque, di impiegarlo come sempli-

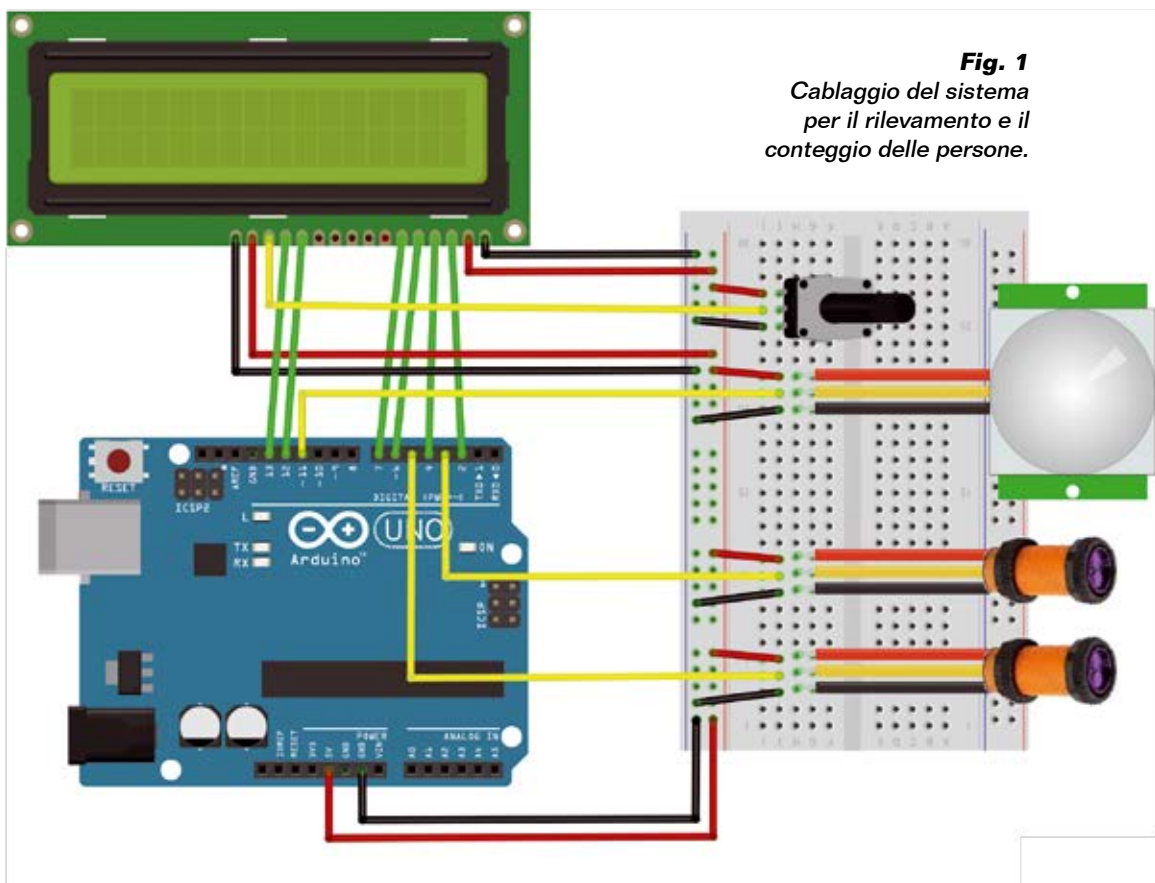


Fig. 1
Cablaggio del sistema
per il rilevamento e il
conteggio delle persone.

ce rilevatore di presenze residue in un locale o in un negozio o ufficio all'orario di chiusura.

STATO DELL'ARTE

Attualmente il problema del controllo e del conteggio degli accessi ad un locale viene risolto spesso utilizzando telecamere e sistemi di riconoscimento più costosi della nostra proposta e con tecnologie totalmente differenti. Alcuni sistemi fanno uso di soluzioni RFID che però hanno un inconveniente: necessitano che le persone interessate portino addosso un adeguato transponder, il che implica evidenti problemi di privacy e di incertezza del conteggio; per non parlare del fatto che per luoghi aperti al pubblico sarebbe impraticabile, in quanto non è pensabile fornire a chi vi entra un tag RFID e ritirarlo all'uscita. Inoltre, di per sé quest'operazione risolverebbe il problema alla base e farebbe venire meno la necessità di automatismi.

Una seconda alternativa, molto economica, vede l'utilizzo di sensori ad infrarosso passivi PIR che di fatto non sono dei sensori di presenza ma di movimento, in quanto sono in grado di rilevare solo le variazioni di radiazione nell'infrarosso emesse da corpi caldi in movimento (quindi delle persone). Questa soluzione presenta però alcuni limiti, uno dei quali è che quando gli occupanti della stanza, non muovendosi, non fanno attivare il sensore, non vi è rilevamento né attivazione; li troviamo applicati in sistemi di azionamento automatico delle luci dei bagni per attivare a tempo luce e aspiratore, dove non a caso accade questo. Nei sensori dedicati a tale applicazione e a quelle simili (ad esempio il controllo del condizionamento e/o dell'illuminazione del locale) per ovviare al problema che in assenza di movimento di disattivano le utenze collegate si ritarda l'intervento di spegni-

mento di un tempo che può arrivare anche a 20 minuti (tempo in cui si presume che gli occupanti abbiano avuto un movimento percettibile dal sistema). Esistono poi sistemi come BLE (Bluetooth Low Energy) ed NFC (Near Field Communication) che consentono il rilevamento della presenza ma presentano il difetto che le persone devono portare sempre dietro lo smartphone.

DESCRIZIONE DEL SISTEMA

Dunque, viste le problematiche riscontrate nel rilevamento e soprattutto nel conteggio delle persone all'interno di una stanza, il metodo di risoluzione ideato in questo sistema, di cui in Fig. 1 riportiamo lo schema di cablaggio, si compone di due sensori di prossimità a infrarosso e di un sensore di movimento PIR come mezzo ridondante, per mettersi al riparo dagli errori. Il conteggio delle persone viene poi riportato nel display LCD.



Fig. 2 - Sensori di prossimità a infrarosso.

I due sensori a infrarosso codificati qui utilizzati sono tarati su una soglia di distanza tra il sensore e il bersaglio (persona) e commutano lo stato della propria uscita dal livello alto a quello basso quando tale distanza si riduce al disotto della soglia impostata (**Fig. 2**).

Ogni sensore è formato da un emettitore di infrarossi modulati da un segnale modulante ben preciso e da un ricevitore costituito da un fotodiode sensibile alla banda emessa dall'emettitore. L'elettronica rileva come segnale utile quello modulante, rendendo questa misura insensibile

ai "rumori" esterni quali la luce ambiente e tutte le emissioni non volute anche nello spettro degli infrarossi, che possono manifestarsi nelle vicinanze dei fotodiode di ricevitori.

La necessità di disporre di una coppia di sensori nasce dall'esigenza di discriminare il verso del transito; con uno solo sarebbe stato impossibile stabilire se si tratta di un ingresso o di un'uscita.

SENSORE FOTOELETTRICO

CARATTERISTICHE TECNICHE

- Uscita: 100mA
- Alimentazione: DC 5V
- Consumo: < 25mA
- Tempo di risposta: < 2ms
- Distanza di rilevamento: 3÷80 cm



SENSORE PIR

CARATTERISTICHE TECNICHE

- Uscita: TTL
- Alimentazione: DC 5V÷12V
- Consumo: < 1mA
- Tempo di risposta: < 2ms
- Distanza di rilevamento: 300÷700 cm

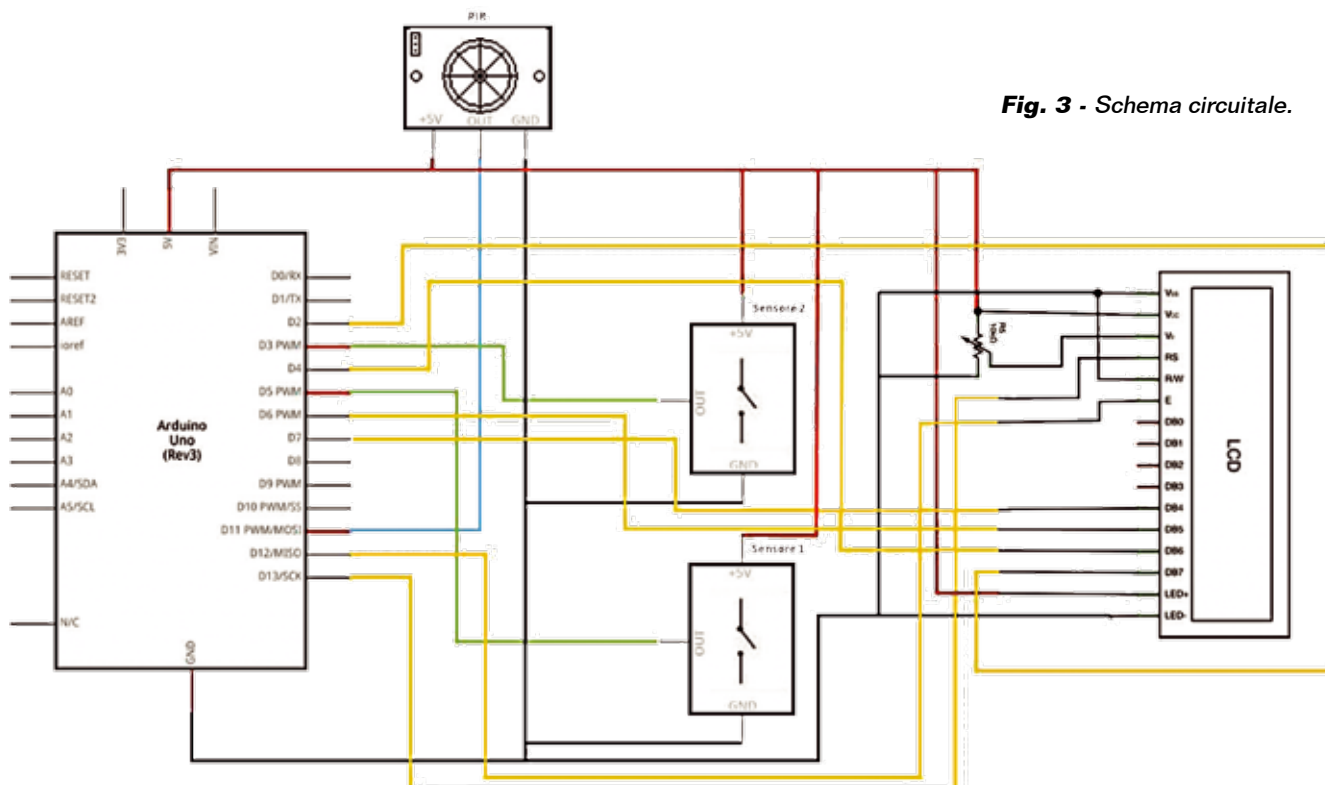


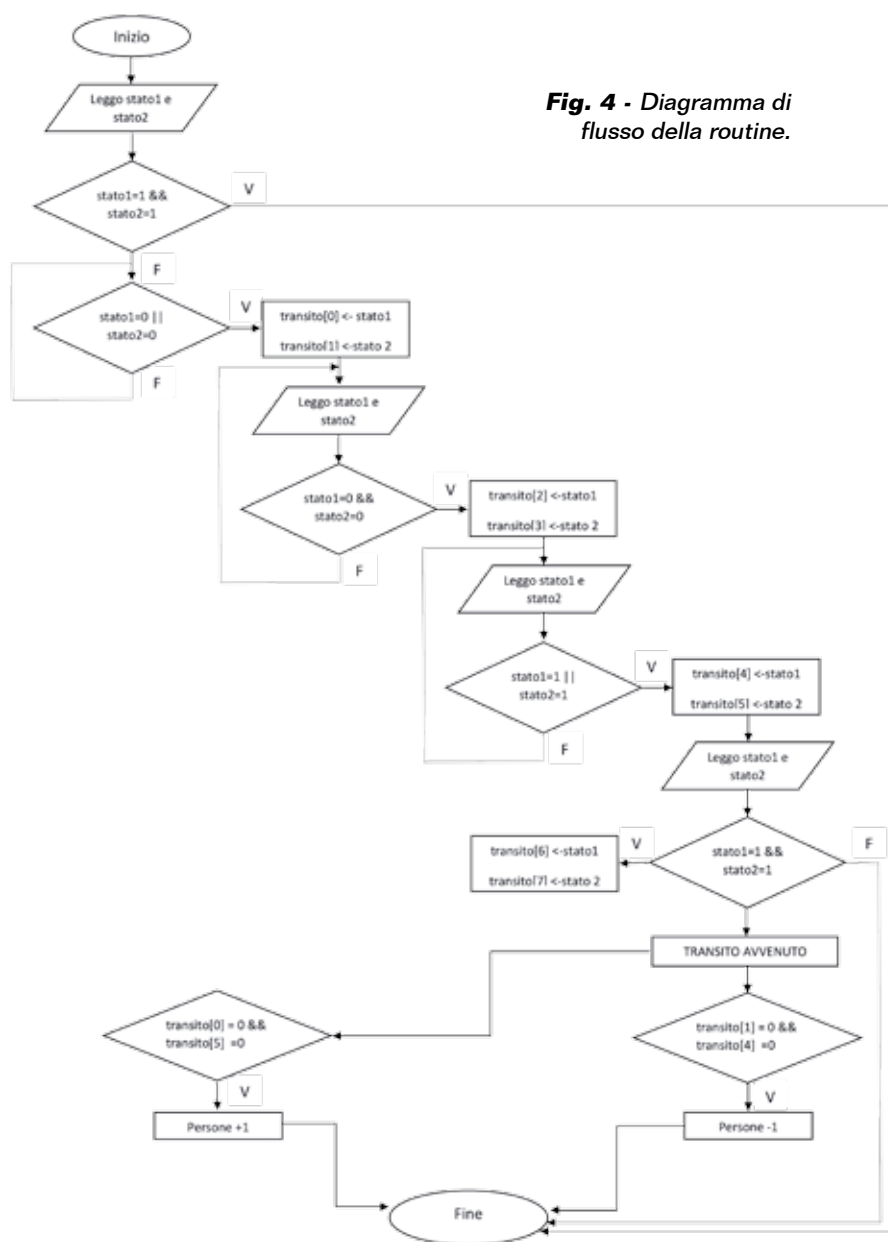
Fig. 3 - Schema circuitale.

Tabella 1 - Tabella di ingresso.

Sensore esterno	Sensore interno	Nota
1	1	Entrambi i sensori non attivi
0	1	Attivato il sensore esterno
0	0	Entrambi i sensori attivi
1	0	Attivato il sensore interno
1	1	Entrambi i sensori non attivati

Tabella 2 - Matrice di uscita.

Sensore esterno	Sensore interno	Nota
1	1	Entrambi i sensori non attivi
1	0	Attivato il sensore interno
0	0	Entrambi i sensori attivi
0	1	Attivato il sensore esterno
1	1	Entrambi i sensori non attivati



scita dal locale monitorato. I due sensori, infatti, comunicano la loro uscita a soglia (attivato/disattivato) alla piattaforma di gestione (Arduino) e in seguito una routine firmware valuta se si è verificato il transito discriminandone il verso. La routine, poi, oltre a riconoscere l'ingresso o l'uscita, provvede a tenere in memoria il numero delle persone (nella variabile *presenza*) nella stanza controllata. La variabile *presenza* viene poi interrogata dalla routine per la gestione di accensioni e spegnimenti delle utenze, aumentando così il grado di efficienza e annullando i tempi morti.

Il sistema è robusto e affidabile, ma, per escludere eventuali errori nel conteggio, abbiamo pensato di renderlo ridondante, come anticipato, con un ulteriore rivelatore di movimento PIR, che, se rileva movimenti quando il contatore vale zero, incrementa di uno il suo valore o, se qualora con contatore maggiore di uno, dopo un certo tempo (molto lungo, da definire) non rileva movimento, porta il contatore a zero, forzando lo spegnimento. Il sistema descritto è stato ideato, studiato, progettato e realizzato in forma prototipale; lo schema elettrico che lo descrive si trova nella Fig. 3. Il cuore dell'unità centrale è un microcontrollore e nel nostro caso la piattaforma utilizzata è Arduino UNO (che si basa sul processore ATmega328) sulla quale gira il firmware da noi sviluppato.

ROUTINE PER IL CONTEGGIO

La routine, utilizzando una logica booleana, valuta la sequenza dei cambiamenti di stato dei sensori di prossimità e considera un transito avvenuto quando si concretizza una delle seguenti due matrici (matrice ingresso o

Listato 1

```

/* dichiaro le variabili */
int transito[8];
int counter;
int persona = 0;
int sensore1 = 5;
int sensore2 = 3;
int statosensore1 = 0;
int statosensore2 = 0;
int pinPIR = 11;

/* inizializzo la comunicazione seriale e definisco i pin */
void setup(){
  Serial.begin(9600);
  pinMode(sensore1, INPUT);
  pinMode(sensore2, INPUT);
  pinMode(pinPIR, INPUT);
}

void loop(){
  /* conteggio persone */
  statosensore1 = digitalRead(sensore1);
  statosensore2 = digitalRead(sensore2);
  //quando i sensori non rilevano passaggio lo stato è 1, quando rilevano un passaggio lo stato è 0
  if (statosensore1 == 1 && statosensore2 == 1)goto continua; //se non passa nessuno eseguo il resto del programma
  if (statosensore1 == 0 || statosensore2 == 0) //inizia entrata o uscita: condizione1 10 oppure 01
  {
    transito[0] = statosensore1;
    transito[1] = statosensore2;
  }
  sensorloop:
  statosensore1 = digitalRead(sensore1);
  statosensore2 = digitalRead(sensore2);
  if (statosensore1 == 1 && statosensore2 == 1)goto continua; //torno indietro
  if (statosensore1 == 0 && statosensore2 == 0) //mi trovo davanti ai due sensori: condizione2 00 transito in corso
  {
    transito[2] = statosensore1;
    transito[3] = statosensore2;
  }
  else goto sensorloop;

  sensorloop2:
  statosensore1 = digitalRead(sensore1);
  statosensore2 = digitalRead(sensore2);
  if (statosensore1 == 1 || statosensore2 == 1)
  // continuo il percorso in entrata o uscita, uno dei due va a 1: condizione3 10 oppure 01
  {
    transito[4] = statosensore1;
    transito[5] = statosensore2;
  }
  else goto sensorloop2;

  sensorloop3:
  statosensore1 = digitalRead(sensore1);
  statosensore2 = digitalRead(sensore2);
  if (statosensore1 == 1 && statosensore2 == 1) //transito finito, si ritorna allo stato iniziale con 11 e faccio conteggio
  {
    transito[6] = statosensore1;
    transito[7] = statosensore2;
  }
  else if (statosensore1 == 0 && statosensore2 == 0)goto sensorloop;
  //ripensamento del transito, tornato indietro quindi mi ritrovo con 00
  else goto sensorloop3;

  if ( transito[0] == 0 && transito[4] == 0) Serial.println("***** ripensamento in entrata *****");
  if ( transito[1] == 0 && transito[5] == 0) Serial.println("***** ripensamento in uscita *****");
  if ( transito[0] == 0 && transito[1] == 0)goto continua;
  if ( transito[6] == 0 || transito[7] == 0)goto continua;

  // ***** incremento persona
  if ( transito[0] == 0 && transito[5] == 0)persona = persona + 1;
  if ( transito[1] == 0 && transito[4] == 0)persona = persona - 1;
  if ( persona < 0 ) persona = 0;
  if ( persona < 0)Serial.println("===== Attenzione: ANOMALIA conteggio =====");
  Serial.print("Numero persone presenti= "); Serial.println(persona);
  /* PIR: correzione conteggio persone */
  if ( persona == 0 && digitalRead(pinPIR) == 1 ) persona = 1; // leggo lo stato del PIR e correggo il numero delle persone
  qualora il conteggio principale non avesse rilevato un ingresso
}

```




L'IoT suona alla porta

Al suo esordio al CES di Las Vegas, lo scorso gennaio, BTicino, il colosso degli impianti elettrici e della domotica, ha presentato l'innovativo videocitofono "connesso" Classe 300, che si connette alla rete dell'abitazione attraverso il collegamento WiFi integrato e, grazie all'integrazione con l'app appositamente sviluppata permette di gestire via smartphone o tablet le chiamate videocitofoniche, visualizzando sul device l'immagine di chi suona alla porta e dando così la possibilità di scegliere se rispondere, aprire il cancello, attivare una telecamera opzionale per guardare intorno in caso di sospetti, o semplicemente azionare le luci del giardino o dell'ingresso/passaggio che porta all'abitazione per agevolare l'arrivo degli ospiti. L'app, anch'essa presentata al CES 2017 e disponibile sia per Android (Google Play) che per iOS (su iTunes Store), si chiama Door Entry e grazie ad essa è possibile avere a portata di mano tutte le funzionalità del videocitofono e rispondere o aprire la porta mentre ci si trova in qualsiasi parte della casa o anche dall'altra parte del mondo, ad esempio per aprire alla domestica. La configurazione dell'app è veloce e grazie al Cloud BTicino, la connessione da remoto è totalmente automatizzata e gestita in sicurezza.

matrice uscita) dove le due colonne riportano le sequenze dei cinque stati logici che i due sensori assumeranno durante il transito e le cinque righe indicano lo stato dei sensori stessi nelle varie fasi di transito (**Tabella 1**).

Al verificarsi della precedente matrice di eventi si considera che sia avvenuto un ingresso nel locale controllato e quindi viene incrementato il contatore presenze (**Tabella 2**).

Al verificarsi della precedente matrice di eventi si considera che sia avvenuta l'uscita di una persona e quindi viene decrementato il contatore presenze.

Quindi se si verificano sequenze

diverse dalle due precedenti, significa che si sono manifestate situazioni diverse da un transito completo e il contatore non verrà attivato, né in incremento né, tantomeno, in decremento. In altre parole, ad esempio per la matrice uscita, dopo la situazione 1-1 ci si deve attendere 1-0; se ciò si verifica ci attendiamo 0-0, se ciò si verifica aspettiamo 0-1 e se anche questa condizione si verifica ci aspettiamo 1-1 per concludere la matrice di uscita. Se questa sequenza viene interrotta, il transito è considerato fallito e il sistema si rimette in attesa di un nuovo tentativo.

La routine scritta viene spiegata

più nel dettaglio con il diagramma di flusso che trovate nella **Fig. 4** e del quale nel **Listato 1** trovate riportato il codice corrispondente contenuto nello sketch Arduino.

CONCLUSIONI

In queste pagine abbiamo presentato un sistema intelligente per il rilevamento certo della presenza delle persone in un locale e soprattutto per il conteggio al fine di valutare, dalla differenza entrate-uscite, quante ve ne restano.

Oltre al miglioramento che il sistema sviluppato presenta rispetto a soluzioni analoghe esistenti, abbiamo anche prospettato scenari di utilizzo che si riallacciano al discorso, oggi attualissimo, dei sistemi smart; grazie al nostro contapersone si potranno quindi attuare logiche per il miglioramento degli ambienti, per aumentare l'efficienza energetica negli edifici, ma anche per garantire la sicurezza antieffrazione e per fornire assistenza alle persone fragili ("assisted living"). ■



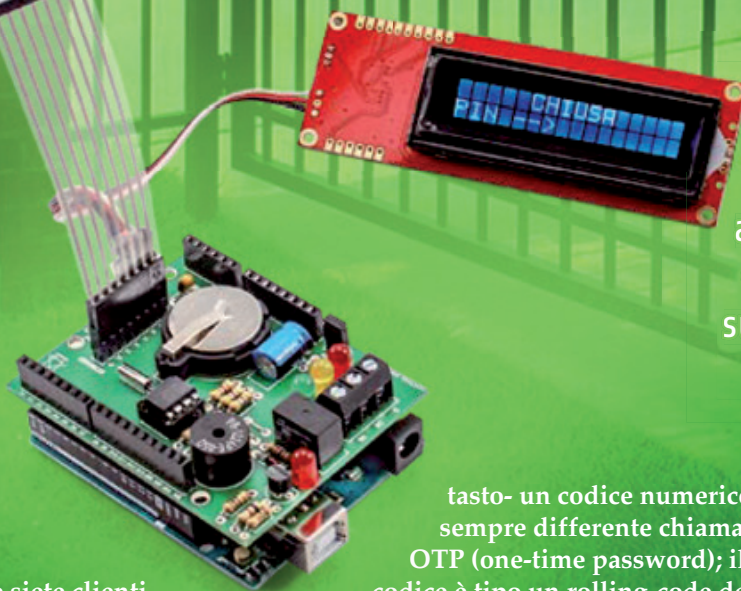
per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono disponibili presso Futura Elettronica. Il sensore fotoelettrico (cod. IRFOTOSENS) è in vendita a euro Euro 7,50 (cadauno), il sensore PIR (cod. PIRMOD) costa Euro 5,00. La board Arduino Uno Rev3 (cod. ARDUINOUNO-REV3) è disponibile a Euro 24,50 e il display LCD 16x2 retroilluminato (cod. LCD1602A) a Euro 7,50.

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

SERRATURA CON CODICE OTP

dell' Ing. LUCA DENTELLA



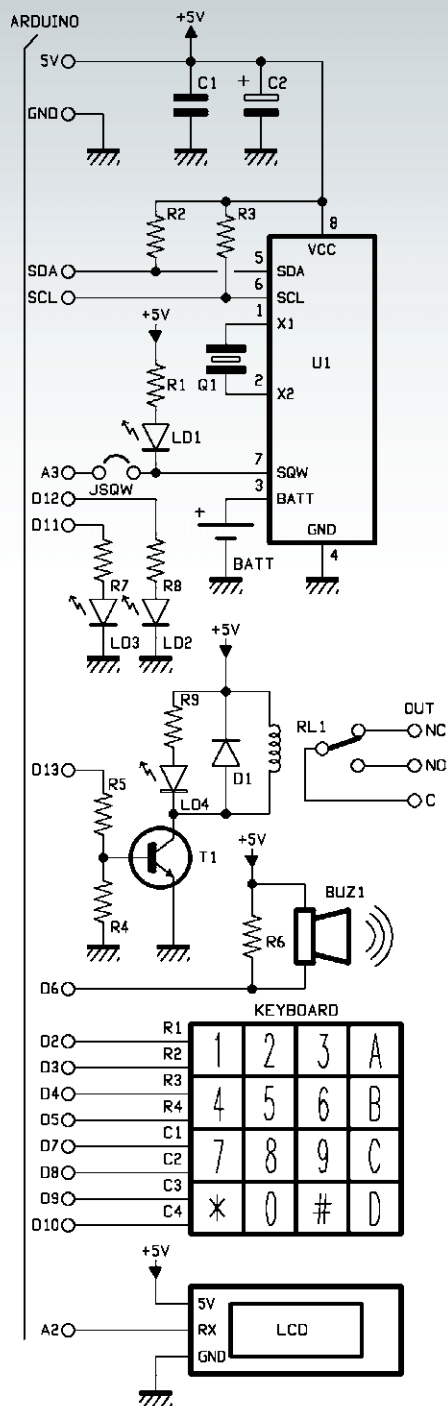
Se siete clienti di una banca on-line, probabilmente già utilizzate un apparecchio chiamato "token" per accedere ai servizi di home-banking o per autorizzare le operazioni. Questi dispositivi, spesso simili a dei portachiavi, sono dotati di un display sul quale appare -ogni tot secondi o alla pressione di un

tasto- un codice numerico sempre differente chiamato OTP (one-time password); il codice è tipo un rolling-code dei radiocomandi a elevata sicurezza e viene generato con un algoritmo pseudo-casuale che tiene conto anche del timestamp, ossia dell'orario attuale. Recen-

Realizziamo una serratura a comando elettrico che si apre grazie ad un codice, sempre diverso, generato dal nostro cellulare sulla base di un algoritmo che tiene conto dell'orario.

temente hanno iniziato a diffondersi anche i soft-token, ovvero applicazioni per smartphone che generano i medesimi codici senza la necessità di avere con sé un ulteriore dispositivo fisico.





Il progetto che vi presentiamo in questo articolo vi consentirà di realizzare una serratura che può essere aperta inserendo, tramite un tastierino alfanumerico, un codice generato da un'applicazione presente sul vostro smartphone. Come vedremo sarà possibile,

tramite semplici modifiche allo sketch di Arduino, legare una serratura ad uno o più cellulari e modificarne il comportamento.

IL CODICE OTP

La generazione dei codici OTP parte da alcuni dati iniziali cui viene applicato un algoritmo, ovvero una serie di operazioni matematiche che trasformano tali dati in un codice numerico. L'algoritmo è deterministico: partendo dai medesimi dati iniziali, si ottiene sempre lo stesso codice. Perché il codice generato cambi, è quindi necessario che almeno uno dei dati di partenza sia variabile; allo scopo i token più diffusi utilizzano come dato variabile il numero di clic sul pulsante (quindi ad ogni pressione del pulsante viene generato un nuovo codice) oppure un valore legato a data e ora attuali. Non c'è normalmente comunicazione diretta tra chi genera i codici (nel nostro caso l'applicazione sul cellulare) e chi li valida (la serratura): la verifica del codice inserito avviene grazie al fatto che entrambi condividono gli stessi dati di partenza e lo stesso algoritmo e quindi possono, in ogni istante, generare il medesimo codice (vedi Fig. 1).

Esistono diversi algoritmi per la generazione dei codici e alcuni produttori utilizzano, inoltre, sistemi chiusi, rendendo i propri token compatibili solo con applicazioni "certificate". Per il nostro progetto, abbiamo deciso di adottare un algoritmo open,

definito dalla specifica RFC6238 – "TOTP: Time-Based One-Time Password Algorithm" (<https://tools.ietf.org/html/rfc6238>); in questo modo sarà possibile generare i codici da una qualsiasi applicazione o token fisico compatibile con tale specifica.

L'algoritmo si basa su una chiave segreta (*shared secret*) e sul valore di *timestamp Unix*, ovvero il numero di secondi trascorsi dalla data Epoch, corrispondente alla mezzanotte del giorno 01/01/1970. Molto importante, lo vedremo nuovamente in seguito, è che tale timestamp sia espresso secondo il fuso orario UTC e che smartphone e serratura siano sincronizzati proprio con l'orario. Se ormai tutti i cellulari si mantengono sincronizzati automaticamente via Internet, per la nostra serratura sarà fondamentale avere a disposizione un "orologio" quanto più preciso possibile, che nello specifico otteniamo con un modulo RTC.

Per generare dei codici che variano ogni *n* secondi, si divide il timestamp per la durata in secondi desiderata e si prende il solo quoziente della divisione: tale numero sarà costante per tutta la durata dell'intervallo. Nella **Tabella 1** vediamo alcuni esempi per capire meglio il meccanismo.

L'HARDWARE

La serratura che realizzeremo sarà basata su una scheda Arduino, al quale applicheremo

Tabella 1

Data e ora	Unix timestamp	Unix timestamp/30
01/01/2016 08:00:00	1451635200	48387840
01/01/2016 08:00:10	1451635210	48387840
01/01/2016 08:00:31	1451635231	48387841
01/01/2016 08:00:59	1451635259	48387841
01/01/2016 08:01:05	1451635265	48387842

uno shield cui collegheremo:

- un tastierino alfanumerico a membrana per l'inserimento dei codici;
- un display seriale per visualizzare lo stato della serratura e i codici inseriti;
- un relé per attivare un carico esterno (ad esempio una elettroserratura) se il codice inserito è corretto;
- un buzzer per generare segnalazioni acustiche;
- due LED (uno rosso e l'altro verde) per visualizzare lo stato della serratura;
- un RTC (*real-time clock*) per avere a disposizione sempre un valore di timestamp aggiornato e corretto;
- una batteria tampone per alimentare l'RTC anche se la scheda non è alimentata.

Per rendere più facile la realizzazione del progetto, abbiamo preparato uno shield che comprende tutti i componenti sopra elencati, il cui schema elettrico è illustrato in queste pagine. Il Real Time Clock è un integrato DS1307 della Maxim/Dallas e si avvale di un quarzo (Q1) da 32.768 kHz che fornisce la frequenza base per l'oscillatore interno; l'integrato è un contatore BCD che conta secondi, minuti, ore, giorni, mesi e anni, provvisto di 56 byte di RAM statica non volatile; il componente determina automaticamente quali sono i mesi con meno di 31 giorni e ad effettuare la correzione per l'anno bisestile.

Fig. 2



Fig. 1



Listato 1

```
// configurazione della tastiera
#define KEY_ROWS      4
#define KEY_COLUMNS    4

char keys[KEY_ROWS][KEY_COLUMNS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[KEY_ROWS] = {5, 4, 3, 2};
byte colPins[KEY_COLUMNS] = {10, 9, 8, 7};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, KEY_ROWS, KEY_COLUMNS);
```

L'orologio può operare nelle modalità 12 o 24 ore, con indicazione delle ore antimeridiane (AM) e di quelle pomeridiane (PM). Le informazioni sull'ora e la data vengono collocate in un apposito registro e trasferite all'esterno al microcontrollore di Arduino mediante l'I²C-bus del quale il chip è provvisto; Arduino fa da unità master dell'I²C-Bus, mentre il chip della Maxim-Dallas è lo slave. Oltre a ciò, il DS1307 dispone di un'uscita di clock programmabile: più esattamente, rende disponibile un'onda quadra ricavata dalla frequenza di clock dell'orologio (determina-

ta, a sua volta, dal quarzo da 32.768 kHz collegato ai piedini 1 e 2) che, mediante un apposito divisore interno, può essere divisa di frequenza ottenendo 1 Hz, 4.096 kHz, 8.192 kHz o l'intero clock.

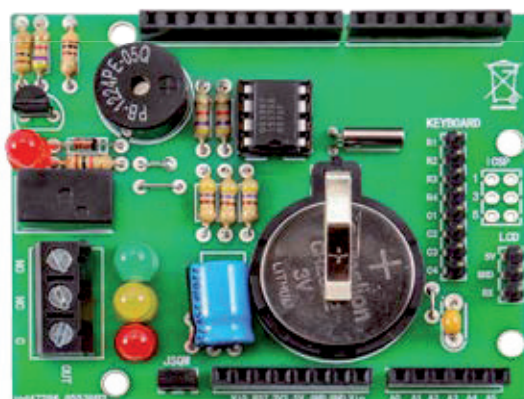
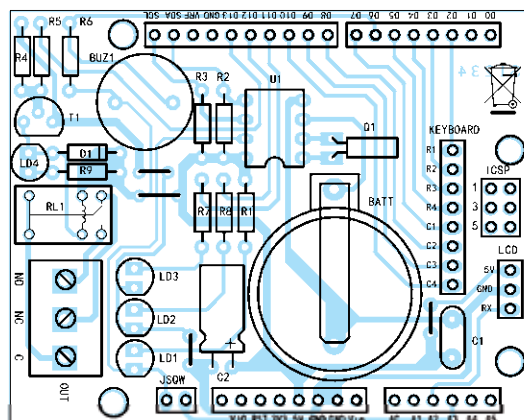
Le frequenze ottenibili non sono state scelte a caso: per esempio, 1 Hz può servire a far lampeggiare i due punti o il punto dei secondi di un eventuale display che mostra l'ora, in un'applicazione da orologio stand-alone. La condizione dell'uscita di clock ausiliario (SQWE, piedino 7) si definisce impostando opportunamente lo stato logico dei bit RS0 (0) ed RS1 (1) del registro di controllo, secondo quanto mostrato nell'apposito riquadro; ad esempio, 1 Hz si ottiene con entrambi i bit a zero. Si noti che quando sia il quarzo (SQWE) che il settimo bit (OUT) si trovano a zero logico, l'uscita di clock si pone fissa a

Elenco Componenti:

R1: 470 ohm
 R2: 4,7 kohm
 R3: 4,7 kohm
 R4: 10 kohm
 R5: 4,7 kohm
 R6: 100 ohm
 R7: 470 ohm
 R8: 470 ohm
 R9: 1 kohm
 C1: 100 nF ceramico
 C2: 220 μ F 16 VL
 elettrolitico
 D1: 1N4148
 LD1: LED 5mm verde
 LD2: LED 5mm giallo
 LD3, LD4: LED 5mm rosso
 T1: BC547
 RL1: Relé minitura 5V
 BUZ1: Buzzer senza
 elettronica
 Q1: Quarzo 32,768 kHz
 BATT: Porta batteria CR2032
 U1: DS1307
 LCD: Display LCD seriale

Varie:

- Morsetto 3 vie
- Strip maschio 2 vie
- Strip maschio 3 vie
- Strip maschio 8 vie
- Strip Maschio/Femmina 6 vie
- Strip Maschio/Femmina 8 vie (2 pz.)
- Strip Maschio/Femmina 10 vie
- Strip Maschio/Femmina 2x3 vie



- Zoccolo 4+4
- Jumper
- Batteria CR2032
- Tastiera a membrana 4x4
- Circuito stampato S1234

alimentazione e che deve essere collegata tra il piedino 3 (Vbat) e la massa (pin 4, ossia GND). La sezione di controllo dell'alimentazione è dimensionata in modo da svolgere due compiti: preservare la memoria in cui sono contenuti i dati sull'ora e la data attuali e mantenere attivo il contatore dell'orologio. Il primo viene svolto conservando nella RAM non volatile le informazioni corrispondenti, mentre il secondo è espletato dall'alimentazione di riserva presa dalla pila o batteria tampone collegata fra il piedino 3 e il 4. La sezione di controllo dell'alimentazione interviene sulle altre funzioni del DS1307, bloccando la comunicazione con l'I²C-bus e la generazione dell'eventuale segnale di clock ausiliario SQW, in modo da minimizzare il consumo di energia ed estendere quanto più possibile l'autonomia. L'intervento della protezione avviene quando la tensione letta tra il piedino dell'alimentazione principale (8) e massa (4) è minore di 1,25 volte la tensione della pila. Il modulo RTC, che deve essere alimentato con una tensione

livello basso; se, invece, il bit 7 è ad uno logico e il 4 a zero, l'uscita assume costantemente lo stato alto. Nel nostro shield l'uscita SQW pilota in modo sink un LED, facendolo pulsare alla stessa frequenza dell'onda quadra prodotta; inoltre, tramite il ponticello JSQW possiamo decidere se far leggere o no ad Arduino, tramite la linea A3, il segnale corrispondente. Come accennato, questo clock ausiliario può servire per attivare dei visualizzatori o scandire certe sequenze: per

esempio far suonare un cicalino ogni secondo, il tutto senza impegnare Arduino in routine di temporizzazione. Il DS1307 dispone anche di un circuito di controllo in grado di verificare la mancanza o l'insufficienza dell'alimentazione principale (Vcc) e fare in modo che la tensione occorrente al proprio funzionamento venga prelevata dalla batteria di backup, nel nostro caso una pila a bottone CR2032 da 3 V, che garantisce un'autonomia di circa 6 mesi in assenza di

Listato 2

```
void pulisciSchermo() {
    displaySerial.write(254);
    displaySerial.write(128);
    displaySerial.write("          ");
    displaySerial.write("          ");
}

void aggiornaSecondaRiga() {
    displaySerial.write(254);
    displaySerial.write(192);
    displaySerial.write("PIN -->          ");
    displaySerial.write(254);
    displaySerial.write(200);
    for(int i = 0; i <= codiceIndex; i++)
        displaySerial.write(codiceInserito[i]);
}
```

continua ben stabilizzata del valore di 5 volt, assorbe una corrente (davvero esigua) dell'ordine di 1,5 milliampere, che scende a 500 nA quando ad alimentarlo è la pila. I 5 volt vengono prelevati dall'omonima linea di Arduino (5V).

Per conoscere l'ora e la data, Arduino deve interrogare il DS1307 mediante l'I²C-bus di cui il chip è provvisto; ad ogni interrogazione, il DS1307 risponde inviando all'ATmega (sempre lungo il suo bus I²C) la risposta e le informazioni su ora e data. Nel nostro shield l'RTC è quindi collegato ai due pin SDA e SCL di Arduino; le due resistenze da 1 kohm R2 ed R3 fungono da resistenze di *pull-up* per il bus I²C, che serve al microcontrollore ATmega di Arduino a interrogare il DS1307 per chiedergli l'orario e leggerlo.

Il relé è comandato dal pin 13 di Arduino tramite il transistor T1; mentre il diodo D1 serve a sopprimere la tensione inversa che la bobina del relé genera quando il transistor va in interdizione.

Il LED LD4, collegato con la sua resistenza in serie e a sua volta in parallelo alla bobina del relé, si accende quando RL1 è attivo. Due ulteriori LED, uno rosso e uno giallo, sono collegati ai pin 11 (LD3) e 12 (LD2) di Arduino tramite due resistenze di limitazione della corrente. Il cicalino viene pilotato dal pin 6 di Arduino.

Sullo shield, infine sono presenti due connettori a strip: uno a 8 pin cui collegare la tastiera a membrana e uno a 3 pin a cui collegare il display LCD a interfaccia seriale. La morsettiera tripolare consente invece di collegare un carico esterno al relé: il contatto centrale della morsettiera è il normalmente chiuso (NC), mentre quelli laterali sono

Listato 3

```
DateTime now = rtc.now();
long GMT = now.unixtime();
long timeStep = GMT / 30;

bool codiceOk = false;
for(int i = -1; i <= 1; i++) {
  har* codiceGenerato = totp.getCodeFromSteps(timeStep + i);
  if(strcmp(codiceInserito, codiceGenerato) == 0) codiceOk = true;
}
```

OTP Tool for Arduino and Google Authenticator

Choose an account name:

Insert your secret (10 characters):

Arduino HEX array:

Google Authenticator code:

QRCode:



Fig. 3

FuturaOTP CONFIG.h

```
1 // chiave privata per la generazione dei codici OTP
2 uint8_t hmacKey[] = {0x70, 0x69, 0x70, 0x70, 0x6f, 0x70, 0x6c, 0x75, 0x74, 0x6f};
3
```

Fig. 4

il contatto normalmente aperto (NO) e quello comune (C).

LO SKETCH

Lo sketch da installare in Arduino è scaricabile dalla pagina del progetto. Per la sua compilazione, è necessario installare tre librerie:

- keypad;
- RTCLib;
- TOTP Library.

Tutte le librerie posso essere installate tramite il Library Manager incluso nelle versioni più recenti dell'IDE di Arduino; dal menu Sketch impartiamo il comando *Include Library* → *Manage Libraries*. Inseriamo il nome della libreria nel campo di ricerca e, una volta trovata, clicchiamo sul pulsante

Install come indicato in Fig. 2.

Analizziamo ora lo sketch, partendo dall'interfacciamento verso le periferiche esterne. Grazie allo shield, il tastierino alfanumerico viene automaticamente collegato a otto pin digitali di Arduino (2-3-4-5 per le righe, 7-8-9-10 per le colonne).

Il Listato 1 mostra come viene utilizzata la libreria Keypad per leggere correttamente i tasti premuti: per prima cosa viene definita la dimensione della tastiera (4 righe x 4 colonne); quindi è configurato il suo *layout* (i vari tasti e la loro posizione) e infine i pin di Arduino a cui è collegata. Una volta effettuata la corretta configurazione, all'interno dello sketch sarà possibile leggere il



Fig. 5

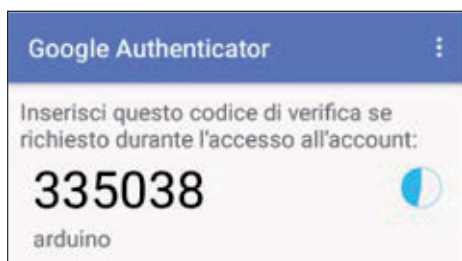


Fig. 6

tasto premuto con l'istruzione `keypad.getKey()`: tale istruzione restituisce una variabile di tipo `char` contenente il simbolo del tasto premuto o la costante `NO_KEY` se non è stato premuto alcun tasto.

Il display LCD scelto per il progetto è di tipo **seriale**, ovvero è possibile controllarlo con un solo pin digitale di Arduino inviando semplici istruzioni. Per non impegnare la seriale *hardware* di Arduino (utilizzata, come vedremo in seguito, per poter configurare l'RTC), all'interno dello sketch è stata definita una seriale *software* che utilizza il pin A2 come pin di trasmissione dati (TX) verso il display. Inviando via seriale un carattere al display, questo lo visualizza nella posizione corrente del cursore, spostando poi quest'ultimo di un carattere a destra o sulla riga sottostante se si è raggiunto la fine della riga. Fa eccezione il carattere 254 (0xFE) che consente di inviare al display

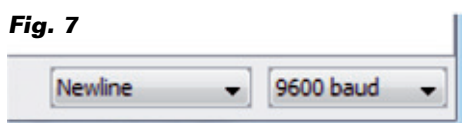


Fig. 7

un comando e il carattere 124 (0x7C) che consente di regolare l'intensità della retroilluminazione. Potete trovare l'elenco dei comandi disponibili per il controllo del display nel data-sheet, scaricabile dalla pagina del prodotto sul sito www.futurashop.it; nello sketch vengono utilizzati i comandi per spostare il cursore all'inizio della prima riga (comando 128) e all'inizio della seconda riga (comando 192) come illustrato nel **Listato 2**.

Il buzzer scelto non è di tipo attivo, quindi per emettere una nota acustica dobbiamo alimentarlo con una tensione variabile unidirezionale (o anche alternata) alla frequenza della nota desiderata; nel nostro caso ci viene in aiuto la funzione `tone(pin, frequenza)` di Arduino, che consente di generare sul pin indicato un'onda quadra della frequenza specificata. Nel nostro caso l'istruzione utilizzata è `tone(PIN_BUZZER, 440)`; che fa generare ad Arduino un'onda quadra alla frequenza di 440 Hz, allo scopo di far emettere al cicalino una nota acustica corrispondente al "La internazionale", vale a dire la frequenza del diapason (LA4). Il pin è chiamato come definito nell'inizializzazione degli I/O di Arduino, all'interno del **setup**.

Vediamo ora la parte più importante dello sketch: all'interno del **loop()** viene continuamente verificato se è stato ricevuto un nuovo carattere via seriale o se è stato premuto un tasto sul tastierino numerico. Nel primo

Fig. 8

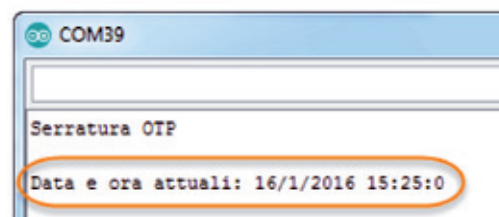
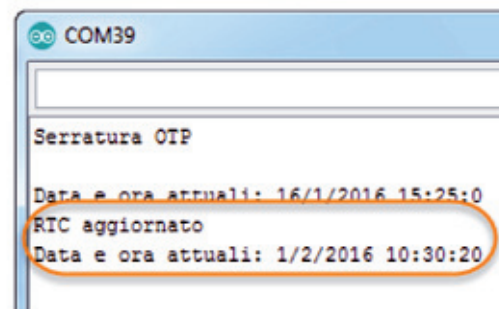


Fig. 9



caso, il carattere viene aggiunto ad un *buffer* e, se si tratta del carattere di NEWLINE (0x0A), si passa ad interpretare il comando ricevuto. Nel secondo caso, la nuova cifra inserita viene aggiunta ad un *buffer* e, se si è completato il codice di 6 cifre; tale codice viene verificato. Possono verificarsi due casi particolari: alla pressione del tasto * viene cancellata l'ultima cifra inserita, mentre alla pressione del tasto # viene cancellato l'intero codice fin qui inserito.

Le righe di codice che riguarda la verifica del codice inserito con la tastiera è quello che trovate nel **Listato 3**; commentiamolo brevemente: per prima cosa viene letto il valore di *timestamp* attuale dall'RTC. Quindi questo viene diviso per 30, corrispondente all'intervallo di validità di ogni codice. Quindi, grazie alla funzione `totp.getCodeFromSteps()`, vengono generati i 3 codici OTP relativi all'intervallo attuale, a quello precedente e a quello successivo; l'utilizzo di tre codici consente infatti un piccolo disallineamento tra l'orario presente nell'RTC e quello del cellulare. I tre codici sono quindi confrontati con quello inserito via tastiera e, se uno corrisponde, la serratura viene aperta.

SETUP INIZIALE

Per poter utilizzare la serratura, per prima cosa è necessario installare sul proprio smartphone una applicazione in grado di generare codici OTP con l'algoritmo scelto. Per iOS ed Android è possibile installare dal relativo market l'applicazione Google Authenticator, mentre per Windows Phone si può utilizzare Authenticator+. Tutte le applicazioni sono gratuite; ecco di seguito gli indirizzi web:

- Google Authenticator per iOS: <https://itunes.apple.com/it/app/google-authenticator/id388497605?mt=8>;
- Google Authenticator per Android: <https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl=it>;
- Authenticator+ per Windows Phone: <https://www.microsoft.com/en-us/store/apps/authenticator/9nblggh08h54>.

Come spiegato all'inizio dell'articolo, è necessario che applicazione e sketch su Arduino condividano la medesima chiave privata per poter generare gli stessi codici; per semplificare la configurazione abbiamo preparato un'applicazione web all'indirizzo <http://www.lucadentella.it/OTP/>.

Dopo aver inserito una password di dieci caratteri a piacere, premere il pulsante GO: apparirà l'array da inserire in Arduino e codice o QRCode per configurare l'app. È anche possibile cambiare il nome dell'account che sarà visualizzato dall'app al di sotto dei codici generati (Fig. 3). Il valore del campo Arduino HEX array va riportato nel file CONFIG.h dello sketch (Fig. 4). Mentre per configurare l'app sarà possibile –dal menu di aggiunta nuovo account– decidere di inserire manualmente il codice o di inquadrare con la fotocamera del telefono il QRCode per una

configurazione automatica (Fig. 5). In entrambi i casi, una volta ultimata la configurazione l'applicazione inizierà a generare i codici (Fig. 6). Dobbiamo infine verificare l'orario presente sull'RTC, ricordando che deve essere quanto più possibile sincronizzato con quello dello smartphone e secondo il fuso orario UTC (non è invece necessario che lo smartphone sia configurato su tale fuso orario perché è l'applicazione stessa a fare la corretta conversione rispetto all'ora locale). Lo sketch consente di configurare l'RTC tramite comandi seriali: apriamo il Serial Monitor e configuriamolo per 9.600 baud con Newline come *line ending* (Fig. 7). Se inviamo il carattere?, lo sketch determinerà la visualizzazione dell'orario corrente (Fig. 8). Se tale orario non è corretto, è possibile aggiornarlo inviando il comando **!ggMMaahhmmss**. Ad esempio per configurare il giorno 01/02/2016 alle 10:30:20 il comando da inviare sarà **!010216103020**. Se il comando è corretto, Arduino risponderà con la conferma della configurazione e visualizzerà il nuovo orario (Fig. 9).

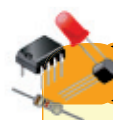
CONFIGURAZIONE SERRATURA

Lo sketch scritto per il progetto consente alcune personalizzazioni, effettuabili modificando il file CONFIG.h. Per prima cosa, è possibile modificare il comportamento della serratura scegliendo tra due diverse modalità alternative l'una all'altra:

- MODO_IMPULSO = la serratura rimane aperta per tot millisecondi e poi si chiude automaticamente;
- MODO_STABILE = la serratura rimane aperta fino alla pressione di un tasto.

Se è stata scelta la modalità IMPULSO, è possibile cambiare la costante TEMPO_APERTURA

per definire il tempo (specificare il numero di millisecondi) in cui la serratura rimarrà aperta a seguito di ogni attivazione del relé. Se invece si è scelta la modalità STABILE, è possibile cambiare la costante TASTO_CHIUSURA per scegliere quale tasto sarà utilizzato per chiudere la serratura (molto importante è che il tasto scelto sia sempre indicato tra apici singoli!). Infine è possibile disattivare il suono del buzzer commentando la riga `#define BUZZER_ON`. Detto ciò concludiamo ricordando che il relé montato sullo shield permette di gestire utilizzatori che assorbano fino a 1 ampere in circuiti funzionanti al massimo a 60 Vcc: quindi elettroserrature che tipicamente vanno a 12/24 Vcc o ca. Se intendete gestire carichi di maggior potenza dovete utilizzare un servo relé di cui alimentare la bobina attraverso lo scambio C/NO del relé posto sullo shield, che utilizzerete quindi come interruttore. L'isolamento galvanico sarà assicurato dal servo relé. ■



per il MATERIALE

Tutto il materiale necessario per realizzare questo progetto è disponibile presso Futura Elettronica. Lo shield attivatore OTP (cod. FT1234K) è disponibile a 20,00 Euro, la tastiera a membrana a 16 tasti (cod. TST16MEMB) costa 3,60 Euro, il display LCD seriale 16x2 cm nero/verde (cod. LCDSER16X2NV) è in vendita a 28,00 Euro e la board Arduino UNO REV3 (cod. ARDUINOUNOREV3) è in vendita a 24,50 Euro.

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

Elettronica In - la più prestigiosa rivista di progettazione elettronica



**TUTTI i MESI
in EDICOLA!**

- Progetti pratici
- Articoli didattici
- Corsi hardware e software
- News dal mondo dell'elettronica
- ...e molto altro ancora.

ABBONATI! SOLO PER TE ALTRI VANTAGGI:

- + La DISCOUNT CARD che ti consentirà di usufruire di uno sconto del 10% su tutti i prodotti acquistati direttamente e on-line da Futura Elettronica.
- + Un volume a scelta della collana "L'Elettronica per tutti".
- + La possibilità di leggere la versione digitale della rivista direttamente da PC.

TI SEI PERSO DEGLI ARTICOLI INTERESSANTI? CREA LA TUA RACCOLTA ARRETRATI SU SUPPORTO USB

Crea con pochi clic una raccolta digitale di arretrati personalizzata. Riceverai, direttamente a casa tua, una card contenente una pen drive USB da 2GB nella quale troverai i pdf dei numeri di Elettronica In che hai scelto.

Le raccolte sono disponibili in tre differenti formati: da 10, da 25 e da 50 arretrati.

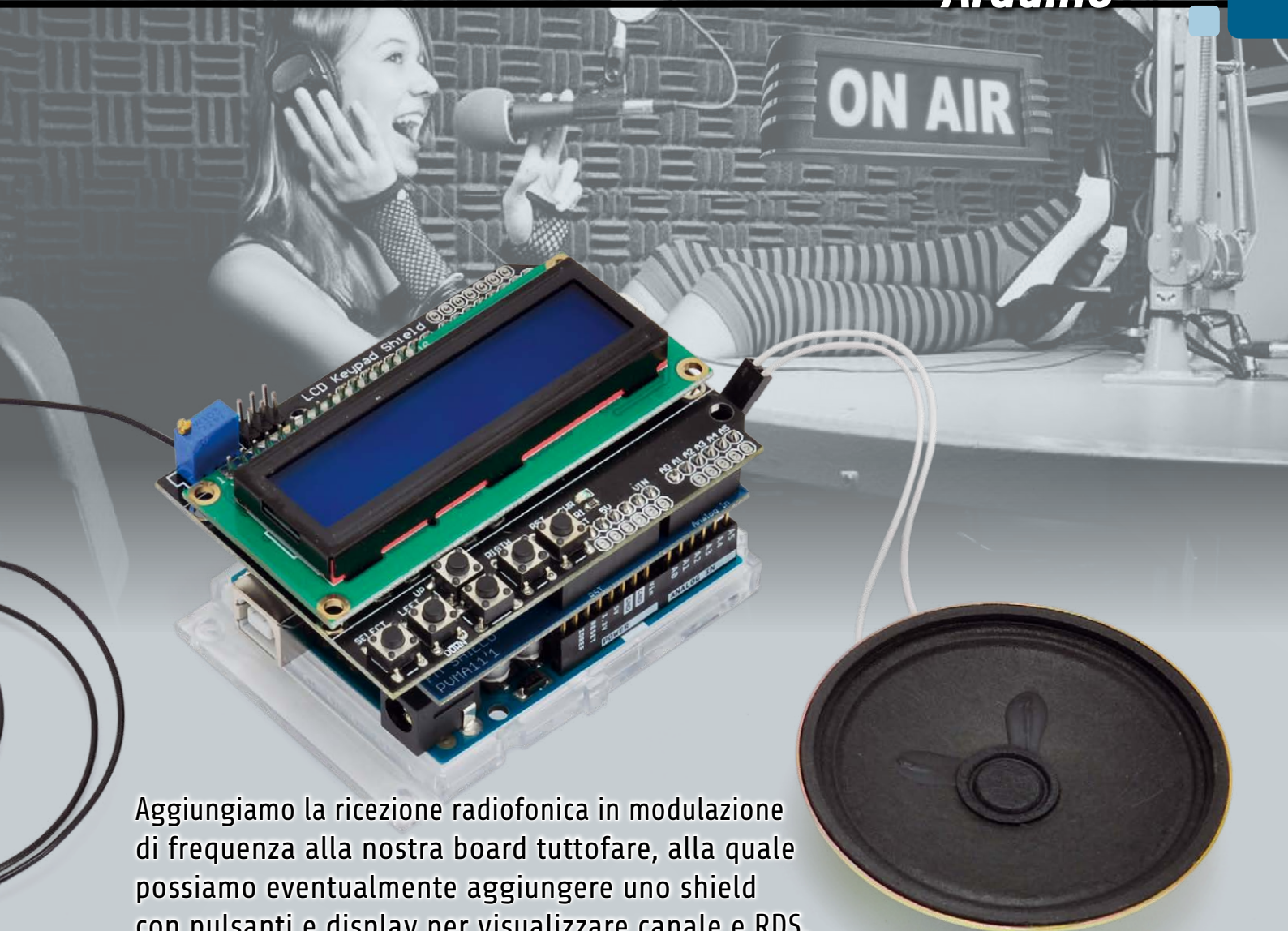


PER ABBONARTI COLLEGATI AL SITO

www.elettronica.in.it

**Futura
Group**
Edizioni

FUTURA GROUP srl
Via Adige 11
21013 Gallarate (VA)
TEL.: 0331 752668
Fax: 0331 792287



Aggiungiamo la ricezione radiofonica in modulazione di frequenza alla nostra board tuttofare, alla quale possiamo eventualmente aggiungere uno shield con pulsanti e display per visualizzare canale e RDS.

FM RADIO SHIELD

di BORIS LANDONI

Da quando ci è apparsa per la prima volta, la popolare scheda di prototipazione nata a Ivrea ed evolutasi in molteplici trasformazioni, l'abbiamo vista svolgere tantissimi e svariati compiti, tanto da farci affermare che Arduino è universale; dall'interfacciamento con sensori e piccoli attuatori, dalla casa al tempo libero, l'abbiamo vista adattarsi a tutte le applicazioni normalmente affidate a semplici microcontrollori consumer. Ora, raggiunta da un pezzo la piena maturità, per Arduino è giunto

il momento di sbarcare nel mondo della radio; ecco quindi nascere uno shield ad hoc, contenente uno dei più recenti radioricevitori FM single chip, già utilizzato in dispositivi (per esempio gli smartphone o le radio USB) dove serve una radio FM compatta, low-power, dalle buone caratteristiche e realizzabile con pochissimi componenti esterni, evitando le tanto odiate bobine di sintonia, vero e proprio spauracchio per lo sperimentatore che voglia avvicinarsi al mondo della radiofrequenza.

CARATTERISTICHE TECNICHE

- Banda di frequenza: FM
- Frequenza sintonizzabile: 76÷108 MHz
- Sensibilità: 1,7 μ V
- Audio: stereo
- Uscite BF per altoparlanti: 2W, 8ohm
- Uscite BF ad alta impedenza: 80 mVrms
- Gestione da PC o da shield opzionale

La radio vera e propria è tutta contenuta in uno shield standard applicabile ad Arduino Uno ed anche alla nostra Fishino Uno e può essere controllata sia da PC, sia localmente, semplicemente

aggiungendo all'insieme uno shield dotato di display alfanumerico e pulsanti.

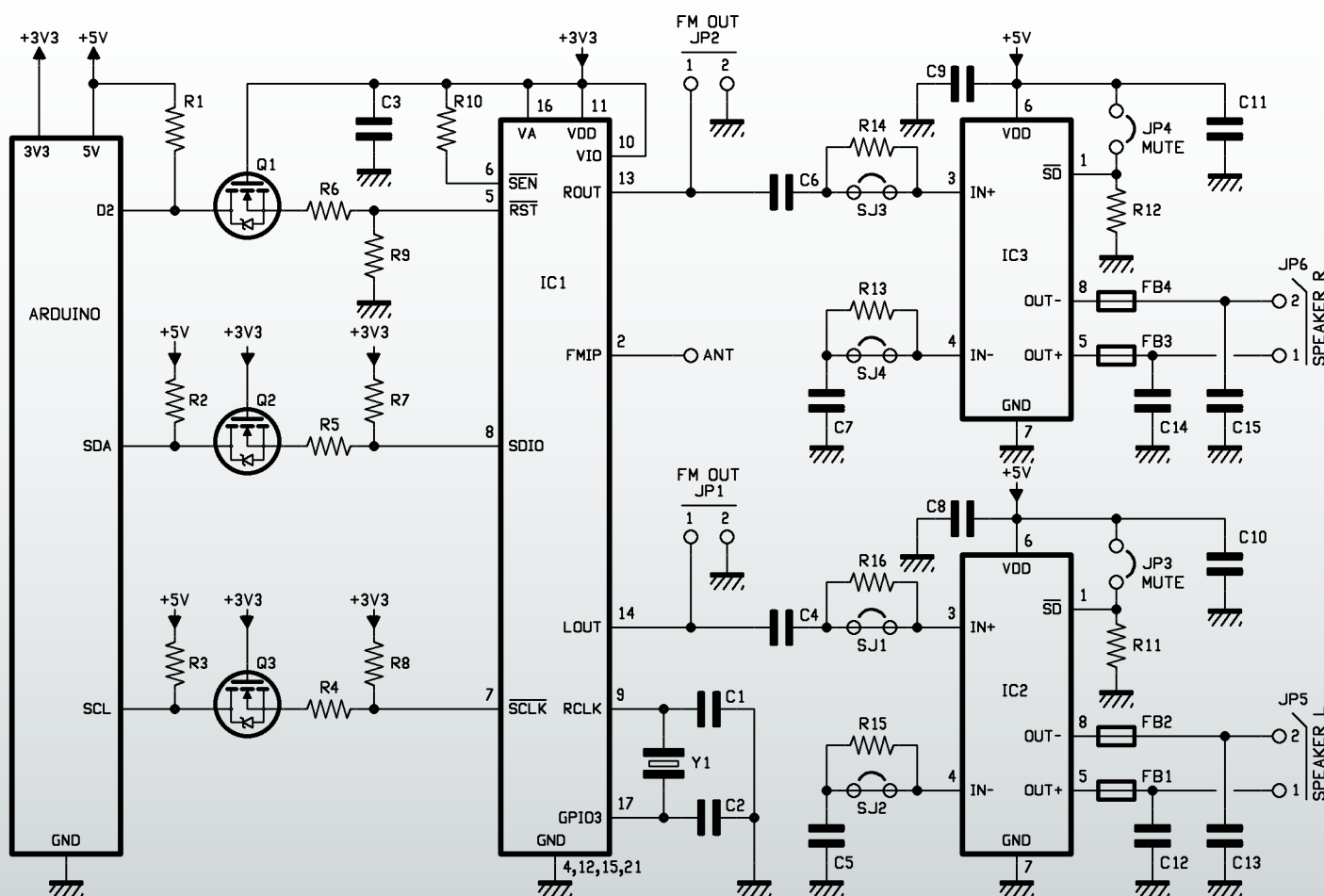
Ma vediamo meglio di cosa si tratta, analizzando il circuito elettronico di tale shield.

SCHEMA ELETTRICO

Escludendo Arduino, che rappresentiamo nello schema elettrico pur non essendo parte dello shield, l'elettronica è composta in buona parte dall'integrato IC1, che ne è il cuore in quanto è un completo ricevitore FM stereo single-chip, oltre che da due amplificatori BF con uscita a ponte siglati IC2 e IC3; l'IC1 è un Si4703 della Silabs e contiene

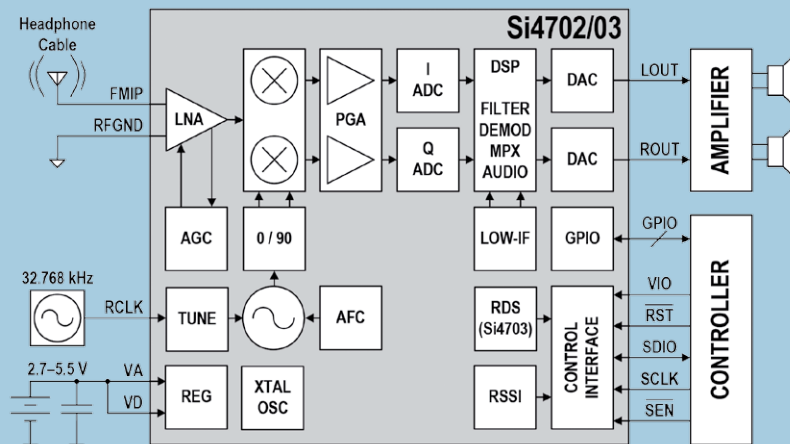
tutto quello che serve a realizzare un radiorecettore FM stereo con tanto di decodifica dei dati RDS (Radio Data System) utilizzati nelle moderne radio per mostrare informazioni come il nome dell'emittente e la frequenza del canale radio corrispondente, il nome del brano e dell'autore, ecc. Il ricevitore è del tipo a conversione di frequenza e all'ingresso dell'antenna ha un amplificatore a basso rumore (LNA=Low Noise Amplifier) che eleva il segnale in modo da conferire al circuito una notevole sensibilità; tale stadio ha un'impedenza di qualche kohm, ben più alta di quella dei classici

[schema **ELETTRICO**]



front-end. Segue un filtro digitale che seleziona la banda corrispondente al canale da sintonizzare, funzionante di pari passo con un oscillatore locale quarzato il cui segnale viene miscelato e da esso si origina la media frequenza, poi demodulata dall'apposito stadio dal quale vengono estratti il segnale digitale corrispondente all'audio mono e i dati dell'RDS ed eventuali altri.

Il decoder stereo estrae i segnali della stereofonia e due DAC (uno per canale) convertono i segnali in audio analogico, reso disponibile alle uscite BF, che sono LOUT ed ROUT, ciascuna delle quali dispone di un test-point (jumper JP1 per il canale sinistro e JP2 per quello destro) per consentire di verificare la presenza del segnale con un oscilloscopio in caso di guasto di uno degli stadi di potenza, ovvero per prelevare il segnale e inviarlo a un amplificatore di potenza. Gli stessi segnali raggiungono due stadi amplificatori a integrato, realizzati con IC2 e IC3 (entrambi sono SSM2305 dell'Analog Devices), che elevano l'ampiezza del segnale BF quanto basta a pilotare un piccolo altoparlante ciascuno, da 8 ohm. Ogni integrato contiene un preamplificatore d'ingresso differenziale, di cui impedenza di input e guadagno si possono impostare entro certi limiti con le resistenze in serie ai piedini 3 e 4 (IN+ e IN-), ed uno stadio finale a ponte in classe D, che permette una potenza d'uscita relativamente elevata con un consumo e un riscaldamento minimi. Lo stadio d'ingresso può essere configurato per funzionare in modo sia sbilanciato che differenziale; nel nostro caso adottiamo la prima soluzione, infatti colleghiamo l'IN- a massa con R13 per il canale destro ed R15 per il sinistro. L'ingresso quindi è



Schema a blocchi dell'integrato Si4703.

La radio, ovvero il radiorecettore FM per Digital Radio, sta tutto dentro a un piccolissimo circuito integrato SMD incapsulato in case QFN, che comprende tutti gli stadi occorrenti, ossia un amplificatore d'antenna (notate che nello schema a blocchi riportato in questo riquadro si fa riferimento al cavo della cuffia perché negli smartphone l'antenna della radio è ottenuta dal cavo dell'auricolare...) un controllo automatico del guadagno (AGC) un oscillatore locale sintonizzato con l'aiuto del quarzo da 32,768 kHz, lo stadio di sintonia digitale, il miscelatore AF, il demodulatore e il decoder stereo, più il decoder dei messaggi dell'RDS (RBDS) e dell'RSSI. Completano la dotazione del chip lo stadio AFC (Automatic Frequency Control) che stabilizza l'oscillatore locale sulla base della deviazione di frequenza riscontrata nello stadio IF (media frequenza) e un regolatore di tensione lineare a basso drop-out che fornisce la tensione stabilizzata necessaria alla logica. L'integrato è progettato per essere gestito da un microcontrollore attraverso un'interfaccia tipo SPI utilizzabile sia in modalità 2-wire, sia in 3-wire. L'integrato in questione è l'Si4703 e fa parte di una famiglia di FM radio receiver integrati in tecnologia CMOS prodotta dalla Silicon Labs. Il sintonizzatore è progettato per supportare la radiorecezione in banda FM in tutto il mondo, nel senso che copre l'intero spettro della FM e non

solo quello italiano/europeo, che si colloca tra 87 e 108 MHz, bensì spazia tra 76 e 108 MHz, così da poter adattare il nostro ricevitore a funzionare in varie zone del Globo. All'interno troviamo un digital low-IF receiver, un sintetizzatore di frequenza basato su VCO e quant'altro descritto.

L'interfaccia digitale permette di implementare il controllo della sintonia (seek tuning) e del volume di uscita sui due canali, oltre che fornire informazioni come l'intensità del segnale radio, l'RDS ecc. Il decoder stereo integrato è progettato per una deenfasi di 50/75 µs (impostabile), così da adattarsi alle convenzioni internazionali in fatto di trasmissione stereo.

L'integrato si alimenta a tensione continua di valore compreso tra 2,7 e 5,5 V ed il suo oscillatore richiede, per funzionare, un semplice quarzo da applicare all'esterno. Non servono circuiti accordati esterni né bobine. La sensibilità tipica della sezione radiorecevente è 1,7 microvolt, ma affinché possano essere interpretati i dati trasmessi con l'RDS occorre un segnale in antenna di almeno 15 microvolt. Le uscite audio forniscono tipicamente 80mVeff. ciascuna su un'impedenza di qualche kohm; la separazione tra i canali è di 25 dB e la risposta in frequenza estesa fra 30 Hz e 15 kHz. La selettività è di -50 dB entro ±200 kHz dal centro banda.

referito a massa. In serie a ciascun resistore c'è l'inevitabile condensatore di disaccoppiamento che blocca la componente continua di polarizzazione dell'amplificatore d'ingresso. Notate che le resistenze suddette ed anche R14 ed R16 (poste in serie alle linee del segnale stereo) sono normalmente cortocircuitate da un ponticello sul circuito stampato (quindi se non vi serve inserirle non montatele), ma possono essere inserite tagliando lo stesso jumper (formato da una pista) se occorre ridurre leggermente il guadagno in tensione e quindi il livello audio in uscita dallo shield. Ogni uscita è a ponte, quindi ha due elettrodi che non devono essere collegati a massa, ma ciascuno ad un capo dell'altoparlante.

Ciascun SSM2305 contiene un modulatore PWM ad alta efficienza e basso rumore con stadio di commutazione a MOSFET, che non richiede il filtro LC accordato all'uscita, ma semplicemente un filtro che sopprime il residuo dell'onda rettangolare; le induttanze utilizzate nei filtri posti all'uscita sono generiche per il filtraggio EMI (Electro Magnetic Interference).

La modulazione utilizzata nell'SSM2305 è una sorta di PWM (è chiamata esattamente Spread-Spectrum Pulse Density Modulation) che permette di implementare basse emissioni di spurie rispetto alla tradizionale classe D, nella quale la modulazione PWM produce impulsi rettangolari i cui fronti di salita e discesa originano armoniche e disturbi (spike) dovuti alla natura induttiva del carico.

In altre parole, gli impulsi vengono distribuiti nello spettro del

segnale portante più o meno fitti, rispetto alla tecnica PWM che prevede una mera modulazione della larghezza degli impulsi. Grazie a tale tecnica l'SSM2305 funziona con un'efficienza del 90% a una potenza di uscita di 1,3 W su 8 Ω e dell'83% a 2,2 W su 4 Ω , alimentato a 5 Vcc, assicurando un rapporto segnale/rumore di oltre 98 dB. L'integrato richiede un'alimentazione singola di 2,5÷5,5 V e può erogare una potenza nominale di 2,2 W su 4 ohm con una distorsione armonica (THD) inferiore all'1% a 5 V. All'interno dell'SSM2305, oltre al modulatore PWM, al generatore d'onda triangolare, al regolatore di tensione che fornisce la polarizzazione (BIAS) al modulatore, allo stadio finale a MOSFET, si trova una protezione termica che spegne lo stadio finale in caso venga raggiunta la massima temperatura ammessa nel chip (150 °C), condizione che nel nostro caso non si verifica. Un'ulteriore protezione è implementata nel blocco soppressore di pop e click il quale evita che all'accensione l'altoparlante riceva il picco di tensione caratteristico dell'assestamento dello stadio finale e del modulatore; si tratta di una versione allo stato solido dell'anti-bump dei finali di potenza.

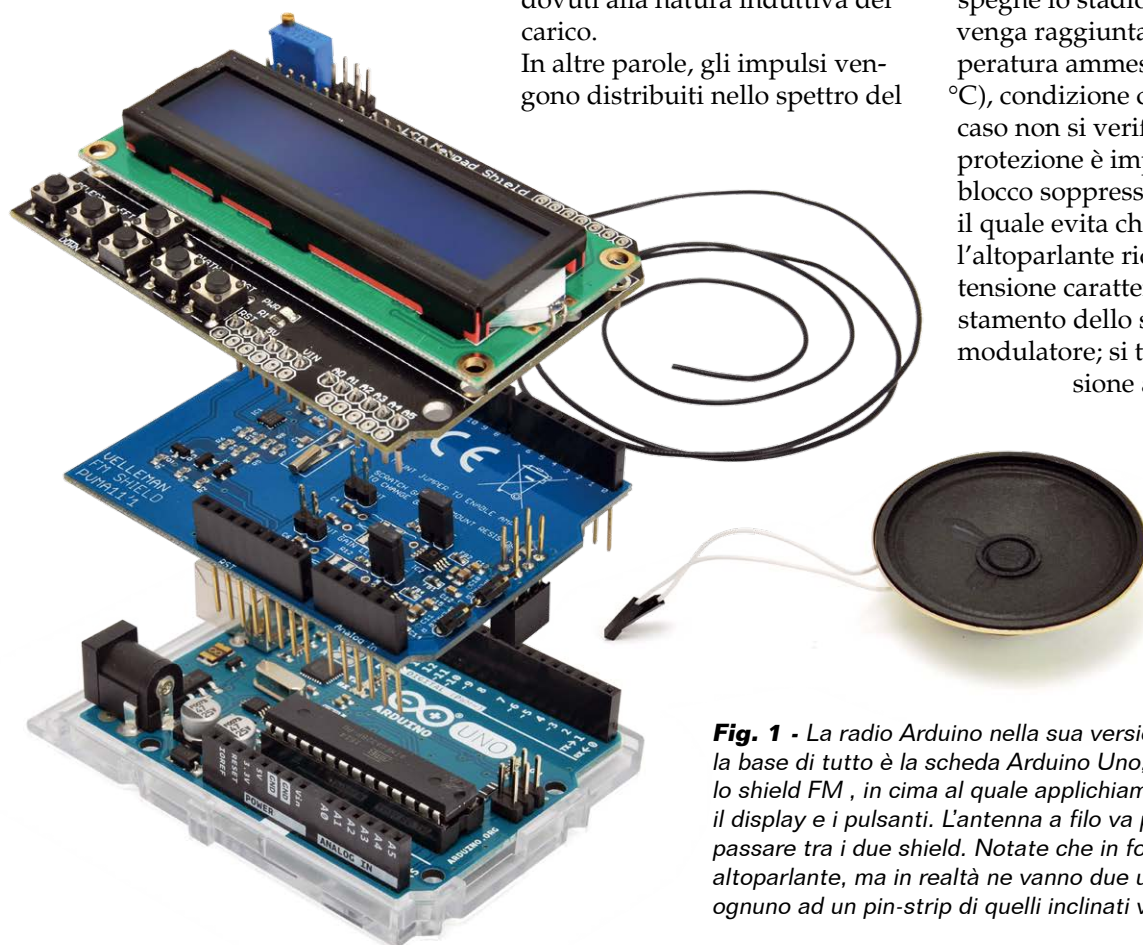


Fig. 1 - La radio Arduino nella sua versione completa: la base di tutto è la scheda Arduino Uno, sulla quale montiamo lo shield FM, in cima al quale applichiamo lo shield contenente il display e i pulsanti. L'antenna a filo va piegata in modo da passare tra i due shield. Notate che in foto appare un solo altoparlante, ma in realtà ne vanno due uguali, collegati ognuno ad un pin-strip di quelli inclinati visibili nello shield FM.

L'SSM2305 prevede la modalità di shutdown, che si attiva ponendo a zero logico il piedino /SD; l'assorbimento di corrente in shutdown è appena 20 nA, tanto basso da poter considerare il componente spento e da consentire di mantenerlo sempre alimentato accendendolo e spegnendolo con un livello logico senza influire significativamente sull'autonomia, nelle applicazioni alimentate a pile o batterie. Il relativo pin è collegato nello schema a un ingresso di MUTE, collegato a massa mediante un resistore (R11 per IC2 e R12 per IC3) e al positivo mediante un jumper, che se viene rimosso tace l'altoparlante corrispondente; quindi normalmente i jumper vanno chiusi. Chi lo desidera può collegare i contatti dei pin /SD all'esterno in modo da poter gestire la funzione da dispositivi esterni. Volendo si può inviarli a dei pin digitali di Arduino, tanto la logica è TTL (0/5V) perché l'intero shield prende l'alimentazione dal pin 5V di Arduino e a 5 volt funziona tutto il circuito. I condensatori posti sui pin di alimentazione (C8, C9, C10, C11) degli integrati SSM2305 servono a filtrare le ondulazioni della tensione sulla linea 5V dovuta al carico quando gli amplificatori sono in funzione; se tali ondulazioni non venissero filtrate disturberebbero il funzionamento dell'integrato radio. La comunicazione con Arduino avviene tramite un bus seriale che fa capo alle linee D2, SDA ed SCL, ovvero all'I²C-Bus di Arduino e all'I/O digitale D2, che in questo caso ci serve per gestire il reset dell'interfaccia del Si4703. La linea /SEN, che è il chip-enable dell'integrato, è collegata fissa ad uno logico perché non ci serve indirizzare più periferiche sul bus: qui ne abbiamo

una sola, che è appunto l'IC1. I tre MOSFET sono stati inseriti come traslatori di livello e la loro configurazione, sebbene appaia strana, serve ad adattare i livelli logici TTL (0/5V) di Arduino a quelli 0/3,3V dell'integrato IC1, ricavati dal proprio regolatore interno LDO; tutti i MOSFET hanno il gate polarizzato con i 3,3V della linea di alimentazione del Si4703 e il drain collegato ad Arduino e dotato di resistore di pull-up terminante, però sul +5V di quest'ultimo. Il source termina sulla linea corrispondente dell'IC1, dotata di resistore di pull-up alimentato dai 3,3 volt. Per comprendere il funziona-

mento della sezione analizziamo quello della parte relativa alla linea SCL, che è un'uscita di Arduino che pilota l'ingresso /SCLK dell'integrato IC1. Quando l'uscita SCL di Arduino è a livello logico basso, il MOSFET Q3, che è a canale N, si trova interdetto in quanto il gate è equipotenziale rispetto al source, tuttavia il suo diodo di protezione stavolta è polarizzato direttamente e porta a 0 logico (0,6V) la linea /SCLK. Invece quando la linea corrispondente di Arduino fornisce l'1 logico, il diodo si interdice ed anche il MOSFET è interdetto, ma il resistore R8 assicura i 3,3 volt sulla linea /SCLK.

Digital Radio anche su Raspberry Pi

La radio va sempre più verso il digitale e non solo su Arduino, tant'è che in commercio si trovano da tempo moduli SDR (Software Defined Radio) composti da un insieme di blocchi funzionali combinabili e gestibili da software per realizzare molteplici configurazioni, specifiche per una certa radoricezione. Un prodotto recente è il dongle USB della Nongles (www.nongles.com), chiamato N3, dotato di oscillatore ad alta stabilità (deviazione di $\pm 0,5$ ppm), chipset standard R820T2 + RTL2832U (con EEPROM 24C02), regolatori integrati 1,2 e 3,3 V, connessione mini USB da cui prendere eventualmente alimentazione, connettore d'antenna F, filtri antidisturbo. Moduli del genere sono supportati da software SDR come RTL-SDR, che gira ad esempio su Raspberry Pi, cosicché basta connetterli all'USB di questa scheda per realizzare una vera radio interamente programmabile gestita da Linux.



Listato 1

```
#include <VMA11.h>
#include <Wire.h>
#include <LiquidCrystal.h>
#include <EEPROM.h>

int resetPin = 2;
int SDIO = A4;
int SCLK = A5;

//Dichiarazioni display
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
int analogPin = A0; //Porta su cui sono collegati i pulsanti
int adc_key_old; //Precedente valore letto su A0
int adc_key_in; //Valore attuale di A0
int NUM_KEYS = 5; //Numero dei pulsanti della scheda
int key=-1; //Variabile di comodo che identifica il pulsante premuto
int adc_key_val[5] = {30, 150, 360, 535, 760 }; //Mappatura valore A0 per ogni pulsante
char msg[5][3] = {"u", "+", "-", "d", "r"}; //Associazione comando a pulsante

VMA11 radio(resetPin, SDIO, SCLK); //Dichiarazione per accesso libreria radio FM
int channel; //Canale impostato
int volume; //Volume impostato
char rdsname[9]; //Nome stazione radio
char rdsrt[65]; //Stringa aggiuntiva di info RDS (non visualizzate su display)
char previousRadioText[65];
uint8_t lastChar;
int countdownRDS=0;
int val_tmp;

void setup()
{
  Serial.begin(9600);
  Serial.println("\n\nVMA11 Test");
  Serial.println("=====");
  Serial.println("+ - Volume (max 15)");
  Serial.println("u d FM + / FM -");
  Serial.println("r Info RDS (15 sec timeout)");

  adc_key_old = analogRead(analogPin); // memorizza lo stato dei pulsanti non premuti
  volume = EEPROM.read(0); //Lettura volume impostato al rpedente avvio
  if ((volume>15) || (volume==0)) //Imposto il volume nel range valido se ancora
  //non salvato
  {
    volume=5;
    EEPROM.write(0, volume);
  }
  //lettura stazione radio impostata al precedente avvio (2 byte)
  val_tmp=EEPROM.read(1);
  channel=val_tmp;
  channel=channel*10;
  val_tmp=EEPROM.read(2);
  channel=channel+val_tmp;
  if ((channel<800) || (channel>1090))
  {
    channel=1025; //Canale che si desidera impostare per default se non c'è già
    //un canale in memoria
  }
  radio.powerOn(); //Attivo modulo radio
  radio.setVolume(volume); //Imposto il volume
  radio.setChannel(channel); //Imposto il canale
  memset(previousRadioText,0,65);
  memset(rdsrt,0,65);
  lcd.begin(16,2); //Display LCD 16x2
  displayInfo(); //Visualizzo info canale
}

void loop()
{
  delay(200);
  ControllaRDS(); //Recupero dati RDS
  if(radio.readRDSRadioText(rdsrt))
  {
    if(strcmp(rdsrt,previousRadioText))
    {
      Serial.println(rdsrt);
    }
  }
}
```

Quindi con l'uscita di Arduino a 0 volt /SCLK si trova a zero logico, mentre con la stessa a 5 V è a 1 logico, cioè a 3,3 volt.

Alla luce di questo modo di funzionamento verrebbe da obiettare che per ottenerlo basterebbero solo il diodo e le due resistenze di pull-up, ma i MOSFET servono in realtà per garantire la comunicazione bidirezionale e in effetti in questo circuito ci occorrono solo per la linea SDA, giacché le altre due sono unidirezionali (da Arduino all'IC1), però sono stati inseriti per futuri sviluppi (ad esempio se si desidera rendere D2 una linea dati bidirezionale).

Vediamo quindi come funziona la cosa nel caso di comunicazione dall'IC1 verso Arduino: quando SDIO deve inviare dati all'SDA, se si porta a zero logico manda in conduzione il MOSFET Q2, giacché quest'ultimo ha il gate polarizzato con 3,3V; il drain si porta circa allo stesso potenziale e trascina a zero logico SDA. Invece quando SDIO è a livello alto, il MOSFET Q2 è interdetto e il resistore R2 mantiene SDA a 5 volt.

Tutti gli stadi logici e la sezione RF quarzata sono sincronizzati attraverso il generatore di clock interno all'Si4703, che si avvale, come componenti esterni, del quarzo Y1 e dei condensatori C1 e C2 per definire e stabilizzare la propria frequenza di oscillazione.

IL CONTROLLO DELLA RADIO

La radio FM costituita dal nostro shield e da Arduino funziona da sola, tuttavia per gestirne le funzionalità (per esempio scegliere il canale, variare il volume) dev'essere controllata da un terzo elemento, che nel caso specifico è un Personal Computer; in pratica, grazie al codice integrato in un'apposita libreria che si

(Continua)

Listato 1 (segue)

```

        strcpy(previousRadioText, rdsrt);
    }
}

adc_key_in=Leggi_Tasti(); //verifica tasti premuti
if (adc_key_in!=-1)
{
    delay(300);
    if (adc_key_in == 0) //Funzione incremento ricerca
    {
        channel = radio.seekUp();
        displayInfo();
        val_tmp=channel/10;
        EEPROM.write(1, val_tmp);
        val_tmp=val_tmp*10;
        val_tmp=channel-val_tmp;
        EEPROM.write(2, val_tmp);
    }
    else if (adc_key_in == 3) //Funzione decremento ricerca
    {
        channel = radio.seekDown();
        displayInfo();
        val_tmp=channel/10;
        EEPROM.write(1, val_tmp);
        val_tmp=val_tmp*10;
        val_tmp=channel-val_tmp;
        EEPROM.write(2, val_tmp);
    }
    else if (adc_key_in == 1) //Funzione incremento Volume
    {
        volume++;
        if (volume >=16) volume = 15;
        radio.setVolume(volume);
        EEPROM.write(0, volume);
        //displayInfo();
    }
    else if (adc_key_in == 2) //Funzione decremento Volume
    {
        volume--;
        if (volume < 0) volume = 0;
        radio.setVolume(volume);
        EEPROM.write(0, volume);
        //displayInfo();
    }
    else if (adc_key_in == 4) //Funzione forzatura ricerca RDS
    {
        // Richiesta info RDS prelevata da Si4703
        radio.readRDSRadioStation(rdsname);
        Serial.println(rdsname);
        lcd.setCursor(0, 1);
        lcd.print(rdsname);
    }
}
}

//Visualizza sul display il canale FM scelto ed eventuale informazione RDS.
void displayInfo()
{
    int valore;
    Serial.print("Channel:"); Serial.print(channel);
    Serial.print(" Volume:"); Serial.println(volume);
    valore=channel/10;
    lcd.setCursor(4, 0);
    lcd.print("FM ");
    if (valore<100)
    {
        lcd.print("0");
    }
    lcd.print(valore);
    lcd.print(".");
    valore=valore*10;
    valore=channel-valore;
    lcd.print(valore);
    lcd.setCursor(0, 1);
    lcd.print(" "); //Cancello seconda riga display
    countdownRDS=0; //Forzo a zero per velocizzare la ricerca RDS
}

```

(Continua)

appoggia a un PC con emulatore di terminale, invia sulla seriale sulla quale è connessa la board Arduino i dati dell'RDS più altre informazioni, mentre accetta comandi, sotto forma di pressione dei tasti freccia della tastiera del computer, per alzare/abbassare il volume d'ascolto, ovvero salire e scendere di frequenza con la sintonia.

Per chi desidera qualcosa in più, è possibile integrare il nostro radioricevitore con un apposito shield che incorpora un display alfanumerico e dei pulsanti; per gestire questo hardware addizionale è previsto un apposito sketch (**Listato 1**) che consente di visualizzare sul display LCD 16x2 tutte le informazioni della radio, vale a dire la frequenza sintonizzata, le informazioni dell'RDS ecc. e di comandare dai pulsanti di bordo il volume d'ascolto e la ricerca dei canali. Non pubblichiamo lo schema di tale shield perché è un prodotto commercializzato in versione già montata e collaudata dalla Futura Elettronica (www.futurashop.it) con il nome Shield LCD keypad per Arduino (codice prodotto 1686-SHIELD-LCD); si tratta di una periferica di input e output in formato shield per Arduino (adattabile alle versioni Uno Rev3, Mega, Duemilanove e Leonardo) con display LCD 16x2 caratteri bianchi, retroilluminazione blu, trimmer per la regolazione del contrasto, 6 pulsanti miniatura (Up, Down, Left, Right, Select e Reset) e piazzole per connettore ICSP.

Per gestire tale shield si utilizza la libreria Arduino *LiquidCrystal.h*, scaricabile dal sito www.futurashop.it e integrata nel nostro sketch (come mostra il **Listato 1**).

Sul piano hardware, lo shield in oggetto si interfaccia con Ardui-

Listato 1 (segue)

```
//Effettua la lettura della porta analogica A0 per verificare se vi è qualche
//tasto premuto sulla shield LCD
int Leggi_Tasti()
{
    int valore;
    valore = analogRead(analogPin);    //Lettura ADC A0
    valore = get_key(valore); //Preleva ID tasto premuto in base ad A0
    return valore;
}

int get_key(unsigned int input)
{
    int k;
    for (k = 0; k < NUM_KEYS; k++)
    {
        if (input < adc_key_val[k])
        {
            return k;
        }
    }
    if (k >= NUM_KEYS)
        k = -1; //No tasto premuto
    return k;
}

//Aggiornamento info RDS sul display. Effettua il controllo ogni 16 secondi.
//Visualizza a display l'eventuale
//indicazione ricevuta.
void ControllorDS()
{
    if (countdownRDS==0)
    {
        radio.readRDSRadioStation(rdsname);
        Serial.println(rdsname);
        lcd.setCursor(0, 1);
        lcd.print(rdsname);
        countdownRDS=80;
    }
    countdownRDS=countdownRDS-1;
}
```

no tramite quattro bit (altrettante linee di I/O di Arduino) e richiede un'unica linea analogica della stessa board per leggere lo stato dei 6 pulsanti di bordo, connessi a una scala di resistenze.

REALIZZAZIONE PRATICA

Il circuito fa largo uso di componentistica SMD, ragion per cui bisogna avere un po' di manualità e l'attrezzatura di base, consistente in un saldatore da non più di 20 W con punta molto fine, del filo di lega saldante non più grande di 0,5 mm di diametro, della trecciola dissaldante e della pasta fluxante, una pinzetta (tipo quelle da garza) e una lente d'ingrandimento per verificare le saldature e aiutarvi nel posizionamento degli elementi più

piccoli. Il circuito stampato è a doppia faccia e si può ottenere per fotoincisione a partire dalle tracce lato rame scaricabili dal nostro sito www.elettronica.in insieme ai file del progetto. Il montaggio del circuito, essendo quasi totalmente SMD, richiede una certa attenzione perché i componenti utilizzati sono piccoli e vanno posizionati con l'aiuto della pinzetta e saldati dopo aver depositato della pasta fluxante sulle rispettive piazzole; la pasta agevola lo scioglimento della lega saldante e il suo posizionamento sulle piazzole e sui contatti dei componenti. Dunque, una volta ottenuto lo stampato, disponete i componenti iniziando con gli integrati, che vanno centrati nelle rispettive piazzole

e poi saldati partendo da un terminale su uno degli angoli, quindi procedendo un terminale di un lato per volta; sistemate poi condensatori e resistenze, le bobine, i MOSFET e il quarzo (in contenitore tubolare), che non è per SMD ma va montato sdraiato sul lato componenti, come fosse un componente SMD.

Per le connessioni con l'esterno (altoparlanti, MUTE) dovete infilare nei rispettivi fori dei pin-strip a passo 2,54 mm e saldarli. Montate anche gli strip femmina a pin lunghi per la connessione ad Arduino.

Verificate, a montaggio ultimato, di aver correttamente orientato i componenti polarizzati, ossia l'integrato e i MOSFET.

Ricordate che se volete inserire le resistenze d'ingresso degli SSM2305 dovete tagliare (interrompere) le piste-ponticello dello stampato e montare R13, R14, R15, R16. Per valutare di farlo è opportuno dire che tali resistori sono preposti all'impostazione del guadagno in tensione dell'amplificatore e rappresentano le resistenze di ingresso di stadi amplificatori configurati in modo invertente, dato che la retroazione di ciascun canale di IC2 e IC3 è realizzata come mostra la **Fig. 2**; siccome la resistenza di retroazione interna (R_f) è di valore fisso, pari a 296 kohm giochiamo sul guadagno (dato dal rapporto R_f/R_i , dove R_i è la resistenza posta in serie all'ingresso che si considera) scegliendo l'opportuno valore di R1 ed R2.

In realtà l'integrato dispone già internamente di una rete con valori di R_f ed R_i pari rispettivamente a 296 e 37 kohm, il che fissa, in assenza di resistori in serie agli ingressi, il guadagno in tensione a 18 dB, corrispondenti a poco meno di 9 volte; ciò significa che per ottenere la

massima potenza d'uscita su 4 ohm ciascun canale deve ricevere in ingresso una tensione di 350 mVeff.

Il guadagno predefinito può essere ridotto inserendo resistenze, che nel nostro caso sono le predette R13÷R16; inserendole (quindi aprendo i ponticelli) il guadagno si riduce a:

$$G = 296k\Omega / (37k\Omega + R)$$

dove R è il valore di R13, R14, R15 o R16.

Va comunque ricordato che la resistenza d'uscita della fonte di segnale audio che si intende amplificare influisce sul guadagno, perché si trova in serie agli ingressi, quindi l'esatta amplificazione dipende anche da essa:

nello specifico, dall'impedenza d'uscita BF dell'integrato radio, che comunque è trascurabile rispetto a quelle integrate nell'SM2305.

Ricordate che se collegate all'uscita degli amplificatori degli altoparlanti, dovete sceglierli da 8 ohm, perché a 4 ohm l'assorbimento di corrente diverrebbe rilevante e insopportabile dalla linea 5V di Arduino.

Per far funzionare la radio occorre collegare al contatto ANTENNA uno spezzone di filo in rame (in guaina) lungo circa 75 cm, ovvero un'antenna a stilo per FM.

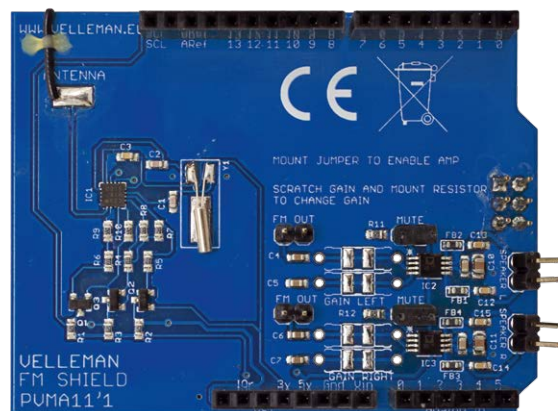
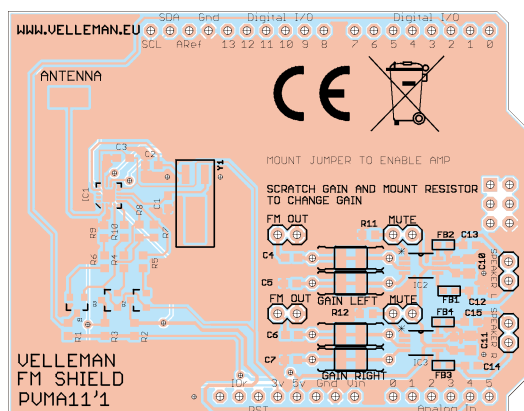
IL FIRMWARE

Passiamo adesso alla parte firmware: come accennato, possiamo ascoltare la radio FM con

Arduino anche senza lo shield con display e pulsanti: basta utilizzare la libreria in dotazione allo shield FM, che permette di controllare via seriale la radio, andando sia a scegliere la stazione radio da sentire, il volume e se avere informazioni dall'RDS, oltre che gestire i canali "preferiti". Per comandare la nostra radio bisogna collegare Arduino a un PC via USB, assegnarle una COM e avviare un programma di terminale anche semplice, che supporti la comunicazione su COM (Hyperterminal, ma anche il Serial Monitor dell'IDE di Arduino).

Appena aperta la porta seriale (la sessione di comunicazione va impostata per una velocità di 9.600 bps) appare il menu di gestione che vi aiuterà nella scelta delle

[piano di MONTAGGIO]



Elenco Componenti:

R1, R2, R3, R11, R12: 1,5 kohm (0603)
R4÷R6: 330 ohm (0603)
R7÷R10: 4,7 kohm (0603)
R13÷R16: vedi testo
C1, C2: 22 pF ceramico (0603)
C3: 22 nF ceramico (0603)
C4÷C7: 220 nF ceramico (0603)
C8, C9: 100 nF ceramico (0603)

C10, C11: 10 µF ceramico (0805)
C12÷C15: 500 pF ceramico (0603)
FB1÷FB4: Ferrite BLM18PG121SN1D
Y1: Quarzo 32,768 kHz (AB26TRB-32.768KHZ-T)
Q1÷Q3: BSS138
IC1: Si4703-C19-GM
IC2, IC3: SSM2305RMZ-R2

Varie:

- Strip maschio 2 vie (6 pz.)
- Jumper (2 pz.)
- Strip Maschio/Femmina 8 vie (2 pz.)
- Strip Maschio/Femmina 10 vie
- Strip Maschio/Femmina 2x3 vie
- Circuito stampato (69 x 54mm)

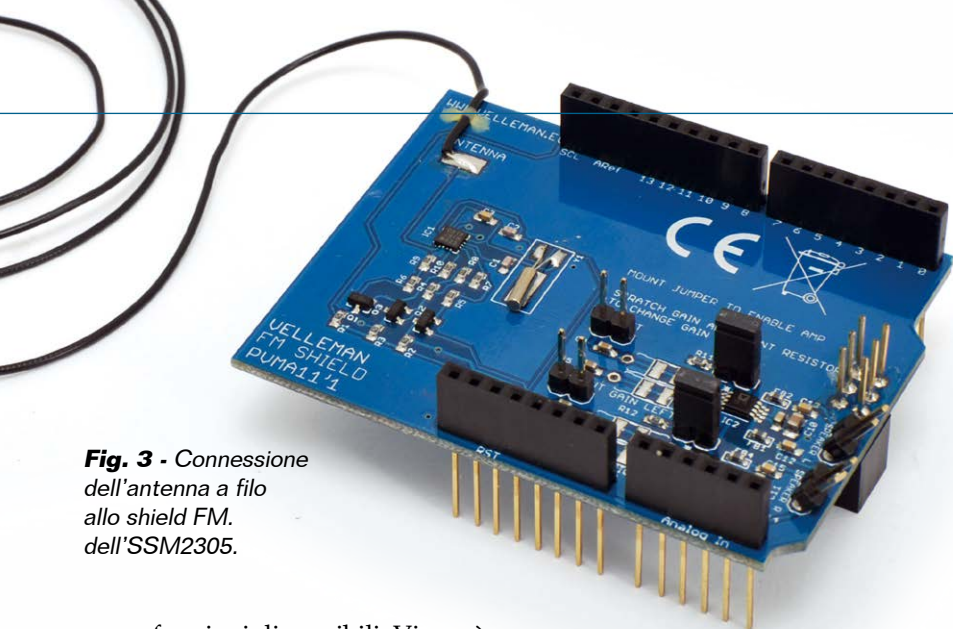


Fig. 3 - Connessione dell'antenna a filo allo shield FM dell'SSM2305.

funzioni disponibili. Vi verrà mostrato il seguente menu:

VMA11 Test Sketch

```
=====
a b  Favourite stations
+ -  Volume (max 15)
u d  Seek up / down
r    Listen for RDS Data
    (15 sec timeout)
R    Listen for RDS Data
    (15 sec timeout)
Send me a command letter.
```

Digitando il carattere desiderato seguito dalla pressione del tasto "Invio", verrà eseguito il comando corrispondente.

Se invece volete utilizzare la radio in maniera autonoma, ovvero avete aggiunto lo shield LCD, dovete caricare in Arduino lo sketch di esempio del Listato 1. Di segui-

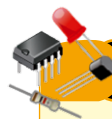
to illustriamo la corrispondenza tra i pulsanti dello shield con i comandi impartiti per la gestione della radio, ottenuta dal codice del predetto listato (notate che i comandi a, b ed R non sono stati gestiti):

- Up > +
- Down > -
- Right > u
- Left > d
- Select > r

Sul display verrà mostrata nella prima riga la frequenza dell'emittente radiofonica sintonizzata, che potrà essere cambiata tramite i pulsanti Right e Left, mediante la funzione di sintonizzazione automatica, mentre i tasti Up e

Down controlleranno il volume. Il tasto Select forza la ricerca delle informazioni RDS, sebbene queste vengano già ricercate ogni 15 secondi in modo automatico tramite la funzione **void ControllorDS()** dello sketch. Le informazioni RDS vengono mostrate nella seconda riga del display in modo automatico e periodicamente, anche se non lo richiedete.

Sia il volume d'ascolto che la stazione radio vengono salvate, in modo che ogni volta che riaccendete la vostra radio vengano reimpostate in automatico. Bene, con questo abbiamo concluso: crediamo di avervi coinvolti abbastanza in questo progetto di radiorecettore, seppure non sia una "old fashion radio" valvolare o dei primordi dell'era dei transistor. Buona costruzione e buon ascolto!

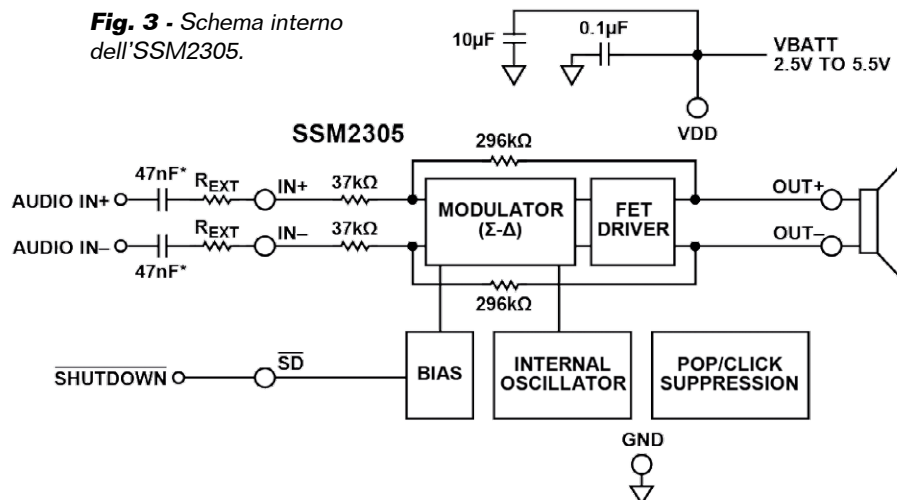


per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono disponibili presso Futura Elettronica. La scheda ARDUINO UNO REV3 (cod. ARDUINOUNOREV3) è in vendita a Euro 24,50, lo SHIELD RADIO FM per Arduino (cod. VMA11) costa Euro 39,90. Lo SHIELD LCD Keypad per Arduino (cod. SHIELD-LCD) è disponibile a Euro 19,00 e l'altoparlante in miniatura 2W/8 Ohm - Ø 66 mm (cod. MLS3) costa Euro 2,90. Al posto della board ARDUINO UNO REV3 è possibile utilizzare la scheda FISHINO UNO (cod. FISHINO UNO) in vendita a Euro 36,00.

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

Fig. 3 - Schema interno dell'SSM2305.



L'ACQUARIO CONNESSO

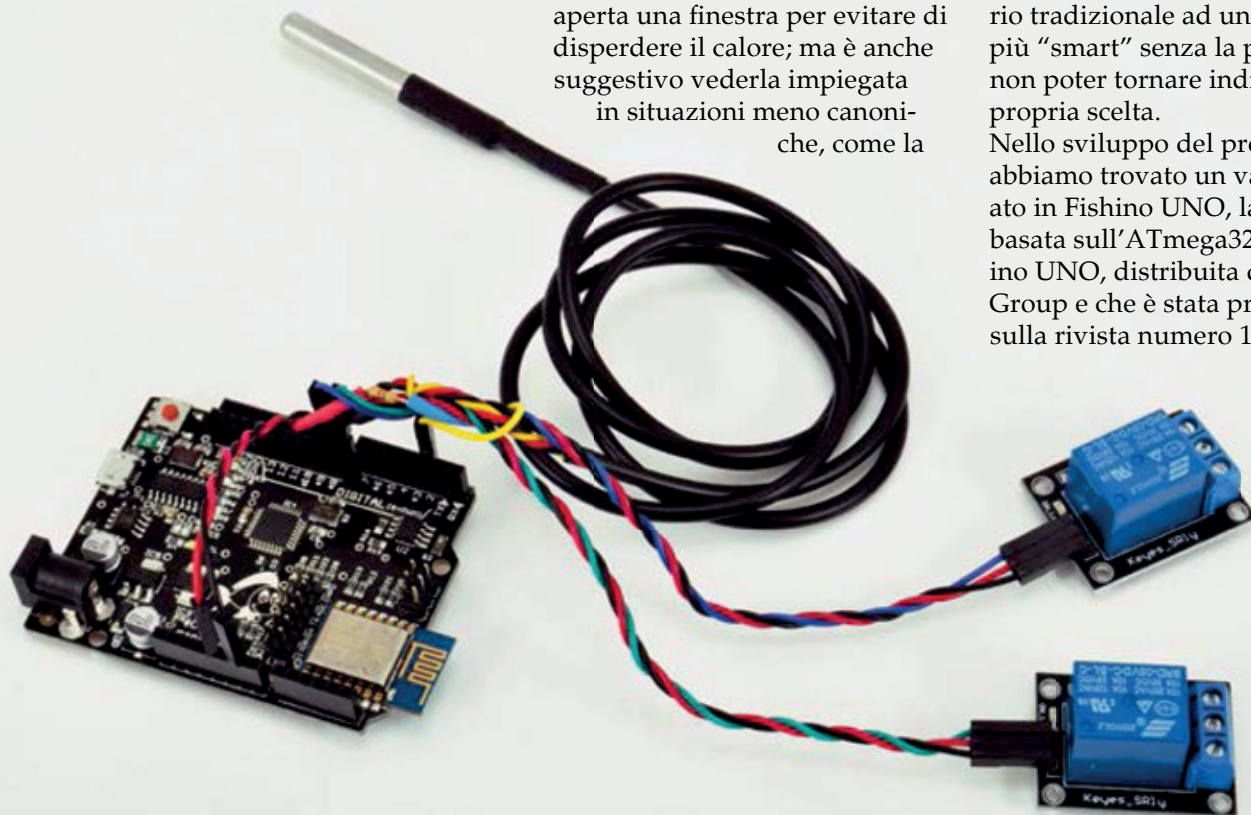
di SIMONE DALL'ASTA

Rendiamo smart un acquario esistente e inseriamolo nel mondo dell'IoT grazie alla scheda Fishino.

La tecnologia si è fatta strada ormai anche nella nostra vita quotidiana e la domotica diviene ogni giorno più familiare: è comodo pensare, ad esempio, ad un sistema di riscaldamento che ci avvisa di aver dimenticato aperta una finestra per evitare di disperdere il calore; ma è anche suggestivo vederla impiegata in situazioni meno canoniche, come la

gestione di un acquario proposta in queste pagine. Il progetto che vi presentiamo consente infatti di gestire un acquario senza però modificarne la struttura o l'aspetto elettrico. L'idea è quindi quella di passare da un acquario tradizionale ad una versione più "smart" senza la paura di non poter tornare indietro nella propria scelta.

Nello sviluppo del progetto abbiamo trovato un valido alleato in Fishino UNO, la scheda basata sull'ATmega328 di Arduino UNO, distribuita da Futura Group e che è stata presentata sulla rivista numero 198; la scel-



CARATTERISTICHE TECNICHE

- Alimentazione: 5 Vcc
- Basato su Fishino UNO (WiFi, RTC e lettore di microSD a bordo)
- Connessione con un database MySQL
- Acquisizione della temperatura dell'acqua tramite la sonda DS18B20
- Possibilità di monitorare la temperatura dell'acquario in un grafico ovunque ci si trovi tramite qualsiasi dispositivo con connessione a Internet
- Invio automatico di e-mail ad un indirizzo impostato in caso di temperature critiche
- Personalizzazione tramite microSD
- Attivazione o disattivazione automatica di aeratore, diffusore di anidride carbonica, plafoniere di illuminazione (se già presenti nell'acquario) tramite due relé

ta è stata dettata da specifiche funzionalità di cui la scheda dispone: una connessione WiFi per permettere di monitorare a distanza la temperatura dell'acqua utilizzando qualsiasi dispositivo dotato di connessione Internet e un orologio interno (RTC, Real Time Clock) con backup a batteria per temporizzare le fasi di funzionamento al posto dei soliti timer analogici (che in caso di blackout dovrebbero essere rimessi in passo). Infine, tramite una scheda microSD, è possibile definire con semplicità vari aspetti legati alla vita nell'acquario, come le ore di luce, quanta anidride carbonica erogare e se attivare o meno un diffusore di ossigeno. Per implementare le stesse funzioni con una classica Arduino UNO avremmo dovuto

dotarla di shield RTC, SD e WiFi, con evidente aumento dei costi e degli ingombri.

La scelta di Fishino UNO è poi naturale, in quanto il nome ...si addice al compito assegnatole!

IL SISTEMA

Vediamo dunque di cosa si tratta riferendoci alla Fig. 1; in essa vedete che il cuore pulsante del sistema è Fishino UNO, grazie al quale è possibile dotare l'acquario delle caratteristiche che lo rendono "smart" e che quindi lo distinguono da un acquario

tradizionale. Il funzionamento del dispositivo è semplice: Fishino UNO "apprende" gli orari di accensione e spegnimento delle plafoniere di illuminazione, dell'erogatore di anidride carbonica e dell'aeratore tramite i file contenuti nella scheda microSD. Dopo questa operazione sarà in grado di attivare o disattivare i dispositivi a seconda dell'orario. Il tempo è scandito dal modulo RTC (Real Time Clock) innestato in Fishino UNO e, anche in caso di alimentazione assente, la batteria di backup non fa perdere l'orario programmato. Grazie alla sonda impermeabile contenente un DS18B20 della Maxim è possibile rilevare la temperatura dell'acqua e, ogni mezz'ora, questo dato viene inviato in automatico tramite WiFi per essere poi memorizzato su un database MySQL (vedremo come). Successivamente sarà possibile, tramite un qualsiasi dispositivo dotato di connessione ad Internet, prendere visione

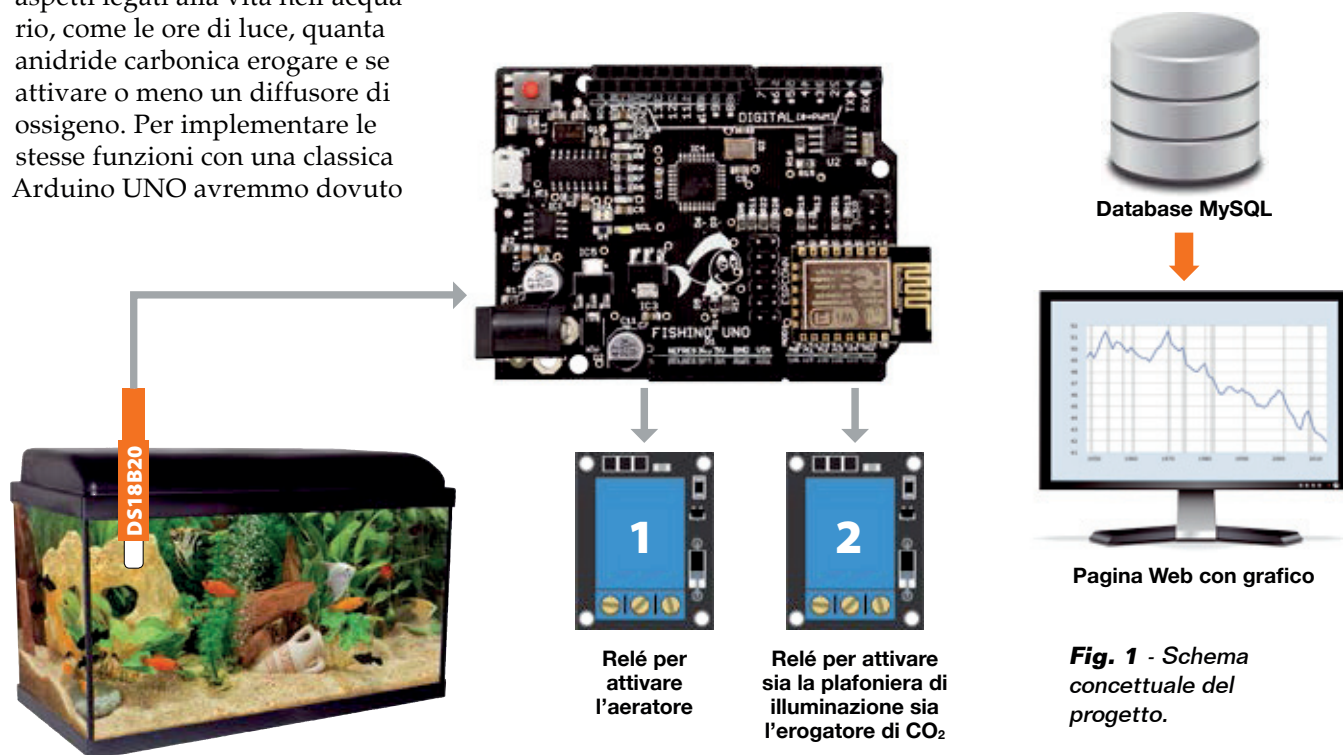


Fig. 1 - Schema concettuale del progetto.

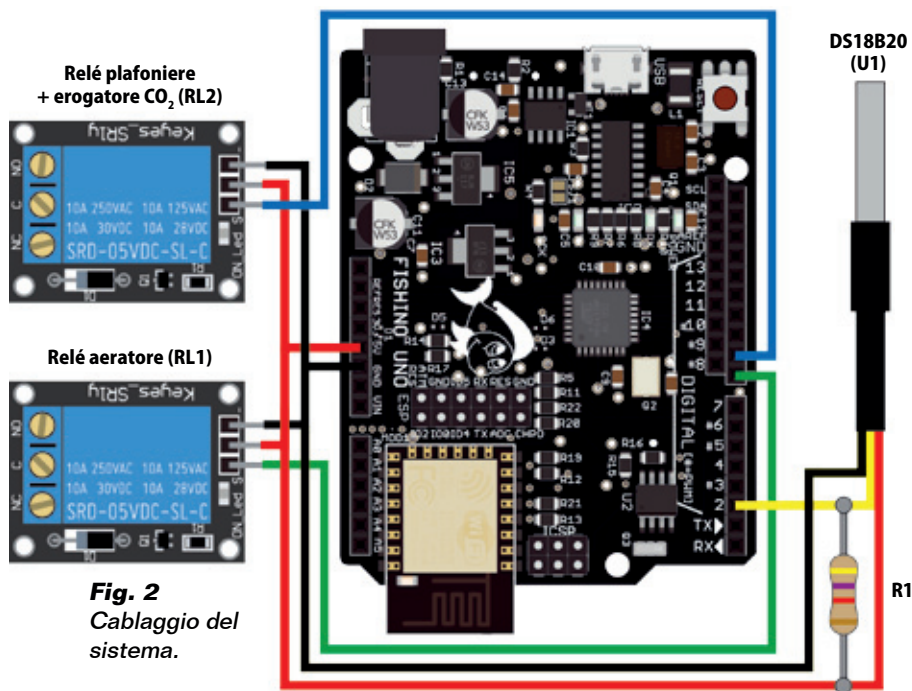


Fig. 2
Cablaggio del sistema.

della temperatura su un comodo grafico per sapere, ad esempio, la minima o la massima temperatura raggiunta dall'acqua. La temperatura rilevata, oltre ad esser salvata sul database, è usata anche per inviare, in caso di temperature critiche (troppo caldo o troppo freddo), una e-mail ad un indirizzo impostato così da essere avvisati e poter agire per tempo.

SCHEMA ELETTRICO

Nella rivista numero 198 abbiamo spiegato l'hardware di Fishino UNO, quindi qui ci limiteremo a spiegare la parte di circuito aggiuntiva richiesta dal progetto, il cui schema di cablaggio è mostrato dalla Fig. 2. Vediamo che la temperatura dell'acqua è acquisita da Fishino tramite una sonda di temperatura waterproof in acciaio inossidabile, realizzata con il sensore digitale di temperatura DS18B20 della Maxim (U1). U1 consente di misurare con precisione la temperatura in ambienti umidi, non solo se immerso in liquidi, con un range che va da -55°C a +125°C; naturalmente per un acquario questi

limiti sono più di quel che serve. Ogni sensore è dotato di un codice seriale a 64 bit memorizzato in una ROM al suo interno; ciò, unito al fatto che è richiesto un solo pin di comunicazione (1-Wire), permette di collegare più di un sensore a bus, in modo da misurare la temperatura in più punti dell'acquario ed ottenere così indicazioni più precise sulle condizioni dell'ambiente. La temperatura è misurata con una precisione di $\pm 0,5^\circ\text{C}$ (nel campo tra -10°C e $+85^\circ\text{C}$). La sonda si alimenta con una tensione che va da 3V a 5,5V.

La connessione della sonda a Fishino UNO si fa così:

- il filo nero va al pin GND;
- il filo rosso va al pin 5V;
- il filo giallo (canale dati) va al pin digitale 2 e gli si deve collegare una resistenza (R1) da 4,7 k Ω verso il 5V (pull-up).

Per quanto riguarda RL1 e RL2, abbiamo utilizzato due schede dotate di relé monoscambio a 240V/10A, pilotabili con segnali TTL; ciascun modulo relé richiede un'alimentazione di 5 Vcc, prelevata da Fishino UNO (cui si connette anche la massa, con il filo nero), e rende accessibili i contatti N.A. (Normalmente Aperto - NO), COM (Comune) e N.C (Normalmente Chiuso). I moduli con RL1 e RL2 si pilotano con Fishino UNO tramite i pin digitali 8 e 9: il livello logico alto corrisponde a relé eccitato e lo zero a relé a riposo. Lo scambio dei relé va usato da interruttore, quindi C ed NO interromperanno un filo dell'alimentazione del carico per poi chiuderlo quando il relé viene attivato. Abbiamo previsto di collegare ad RL1 l'aeratore e di gestire con RL2 delle plafoniere di illuminazione e l'erogatore di CO₂. Per quanto riguarda l'apporto di anidride carbonica e di ossigeno siete liberi di usare il metodo con il quale siete abituati: potete, ad esempio, collegare al relé un'elettrovalvola normalmente chiusa che apre e chiude il tubo dentro il quale passa la CO₂ dopo aver attraversato il riduttore di pressione posto sopra la bombola. I carichi di plafoniere ed erogatore di anidride carbonica possono essere gestiti distintamente usando un relé aggiuntivo, ma, a nostro avviso, è più semplice usarne uno soltanto poiché, come ben saprete, l'apporto di CO₂ è necessario soprattutto durante i

Listato 1

```
CREATE TABLE acquario (
  Temperatura FLOAT NOT NULL,
  Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP PRIMARY KEY
)
```

The diagram illustrates a data processing cycle. At the top left, a computer monitor displays a web page with a table of data. Below it, the text 'Pagina Web' is followed by '(caricamento dati + invio e-mail)'. An orange arrow points from this monitor to a central database icon, which is a stack of three cylinders labeled 'Database'. Another orange arrow points from the database to a second computer monitor on the top right. This monitor displays a line graph, with the text 'Pagina Web (grafico)' below it.

SOFTWARE

database relazionale MySQL, che ne permette poi il recupero per costruire un grafico sulla base della temperatura rilevata ogni mezz'ora. Potremmo vedere le due parti seguendo il percorso logico che seguirà il dato della temperatura dal momento dell'acquisizione a quello del

salvataggio fino ad arrivare all'output nel grafico. Conviene però cominciare dal fondo, ovvero da come avviene il salvataggio del dato su database, perché ciò vi aiuterà a capire il funzionamento dello sketch.

PAGINA WEB E DATABASE RELAZIONALE MYSQL

Per tenere monitorata la temperatura dell'acqua usando qualsiasi dispositivo dotato di connessione a Internet, è necessario rendere disponibile al mondo esterno ciò che Fishino UNO rileva; il modo più semplice per farlo è usare un sito web personale attraverso cui sarà possibile avere una sorta di finestra condivisa tra voi e Fishino. In **Fig. 3** potete osservare il percorso effettuato dal dato della temperatura on-line: una prima pagina web si occupa della ricezione del dato e del salvataggio su database, mentre una seconda servirà per prelevare i dati dal database e mostrarli nel grafico.

Crea tabella

Nome tabella:

Struttura

Campo	Tipo	Lunghezza/Set*	Predefinito ²	Collation	Attributi	Null	Indice	AUTO_INCREMENT	Commenti
Temperatura	Float		Nessuno			☐	---	☐	
Timestamp	Timestamp		CURRENT_TIMESTAMP			☐	PRIMARY	☐	

#	Campo	Tipo	Collation	Attributi
1	Temperatura	float		
2	<u>Timestamp</u>	timestamp		

Fig. 4
Campi
della
tabella
"acquario".

Fig. 5 - Contenuto della tabella dopo la creazione.

Fig. 6 - Il contenuto della tabella "acquario" dopo aver inserito sei valori della temperatura.

		Temperatura	Timestamp
☐	Modifica Copia Elimina	27	2015-11-08 11:51:01
☐	Modifica Copia Elimina	27.9	2015-11-08 11:51:05
☐	Modifica Copia Elimina	27.91	2015-11-08 11:51:08
☐	Modifica Copia Elimina	27.31	2015-11-08 11:51:13
☐	Modifica Copia Elimina	26.31	2015-11-08 11:51:18
☐	Modifica Copia Elimina	25.61	2015-11-08 11:51:25

Per prima cosa va quindi creato il sito web, se già non ne avete uno.

Per fare questo vi potete affidare ai tanti servizi di hosting disponibili a pagamento o gratuitamente; l'importante è che nel servizio sia inclusa la possibilità di disporre anche di un database MySQL. Una volta creato il sito web e attivata la base di dati, recatevi nella sezione nella quale sarà possibile gestire il vostro database (se ne avete più di uno sceglietene uno qualsiasi).

Una volta dentro, create una tabella (è comodo usare lo strumento di creazione sicuramente offerto) dal nome *acquario* con i seguenti campi (**Fig. 4**) rispettando maiuscole e minuscole:

- Temperatura
 - Tipo: FLOAT;
- Timestamp
 - Tipo: TIMESTAMP;
 - Predefinito: CURRENT_TIMESTAMP;
 - Indice: PRIMARY.

In questo modo avete creato una tabella con due parametri dove *Timestamp* (che, se non definito diversamente, è impostato automaticamente alla data e all'ora del server su cui è ospitato il database) è la chiave primaria, ovvero non potranno mai esserci due righe della tabella con lo stesso Timestamp. Se non volete usare lo strumento per la creazione delle tabelle, potete usare

la console (di solito la si trova sotto la voce *SQL*) e inserire il codice del **Listato 1**.

Poi cliccate su *Esegui* oppure *Salva*, a seconda dei casi. Il risultato sarà come quello mostrato in **Fig. 5**.

Per quanto riguarda la pagina web che verrà usata da Fishino per caricare la temperatura sul database, può essere utilizzato uno script PHP. La scelta è ricaduta sul PHP poiché, essendo un linguaggio di scripting server-side, si presta perfettamente al compito da svolgere: Fishino opera come client, si connette al server, invia la temperatura che viene elaborata e salvata su database. Per scrivere la pagina è sufficiente un editor di testo come il Blocco Note di Windows. Il file dovrà essere salvato, ad esempio come *in.php* (vi consigliamo di cambiare nome ma non sceglietene uno troppo lungo), e successivamente caricato sul vostro spazio web (preferibilmente nella directory radice) usando, ad esempio il pannello utente (o gestore file) del servizio di hosting. Vediamo insieme le parti di cui è composto lo script nel file *in.php*. Cominciate

aprendo il tag `<?php` (ricordatevi di chiuderlo con `?>` alla fine); da qui in poi seguirà il vostro script. Dovrete poi riportare (**Listato 2**) i parametri di connessione al vostro database (li trovate solitamente nel pannello di controllo): `$username`, `$password`, `$db`, `$tablename`, `$host`. I nomi delle variabili (ricordiamo che nel linguaggio PHP si indicano con il simbolo `$` davanti) possono essere anche cambiati, ricordandosi però di mantenere il nome scelto per tutto lo script. Alcuni servizi di hosting consigliano di lasciare vuoti i campi di username, password e host per due motivi: innanzitutto sono recuperati automaticamente, poi lo svantaggio del PHP è che spesso, in caso di errori di sintassi o di altro genere, viene stampato a schermo tutto il codice, compresi questi parametri personali. In ogni caso, se non fate errori, non c'è alcun rischio. La variabile `$tablename` rappresenta il nome della tabella; noi la abbiamo chiamata "acquario" ma se avete scelto un nome differente allora dovrete sostituirlo con quello da voi impostato.

Per fare interpretare al server i dati che gli verranno passati da Fishino UNO, usiamo il metodo GET (avremmo potuto usare anche il metodo POST) con il quale si fanno richieste ad un Web Server. Il metodo GET consiste nell'accodare i dati che ci interessano all'indirizzo della pagina richiesta. Per farlo si deve aggiungere, dopo l'indirizzo della pagina, un punto in-



Fig. 7 - Esempio di e-mail inviata a seguito di una temperatura critica.

Listato 2

```
<?php
//-----
// ----- TODO -----
$username="username";
$password="password";
$db="database";
$tablename = "acquario";
$host="host";
//-----
```

Listato 3

```
$temperatura_acquario = $_GET['t'];

echo "Temperatura acquario = " . $temperatura_acquario;
```

Listato 4

```
if ($con = mysqli_connect($host, $username, $password))
{
    mysqli_select_db($con,$db) or die ("Unable to select database");
    $query = "INSERT INTO $tablename VALUES ($temperatura_acquario, now())";
    $result = mysqli_query($con,$query);
}
else
{
    echo('Unable to connect to database.');
```

terrogativo seguito dalle coppie *nome = valore* dei dati di interesse. Se vi è più di una coppia, esse vanno separate dal simbolo &. Il risultato sarà una variabile globale `$_GET` contenente l'array di tutte le coppie campo=valore che potremo usare nello script. Nel nostro caso avremo solo una coppia formata dalla temperatura con relativo valore. Vorremmo quindi passa-

re, ad esempio, la coppia *temperatura=25.6* (con il punto, non la virgola) al Web Server in modo tale da farglielo interpretare e, successivamente, caricare sul database. Per fare questo si dovrà assegnare alla variabile `$temperatura_acquario` il valore del campo *temperatura* passato con l'array `$_GET`. Se, ad esempio, chiamate il campo della temperatura con la

lettera t, allora si potrà selezionare questo specifico elemento dell'array rappresentato dalla variabile `$_GET` scrivendo `$temperatura_acquario = $_GET['t']` (ricordando gli apici prima e dopo t). In questo modo, digitando nella barra degli indirizzi del browser `www.miosito.it/in.php?t=25.6`, come risultato avrete che la variabile `$temperatura_acquario` sarà pari a 25,6. Se poi voleste visualizzare nel browser il dato appena inserito, basterà mostrarlo in output con la funzione `echo` (o `print`, o una funzione con lo stesso compito). È da notare che se vi dovessero essere errori di sintassi, di battitura o di altra natura nella query MySQL, questi non verrebbero notificati ma, semplicemente, non verrebbe caricato alcun dato sul database. In caso di errori vedreste semplicemente la temperatura in output ma, controllando nei dati contenuti nel database, vedreste una tabella vuota. Per verificare di non aver commesso errori, dopo aver fatto la prova appena descritta (inserire la temperatura di 25,6), controllate che il dato sia stato effettivamente salvato nel database.

Se tutto è avvenuto senza errori allora la tabella non sarà vuota ma conterrà i dati appena inseri-

Listato 5

```
//Invio mail per temperature critiche
if ($temperatura_acquario < 23)
{
    $messaggio = "La temperatura dell'acquario e' troppo BASSA! Rilevata temperatura di " .
    $temperatura_acquario . "°C in data: " . date("Y-m-d H:i:s");
    mail('mail@mail.com','Temperatura acquario CRITICA',$messaggio);
}
if ($temperatura_acquario > 29)
{
    $messaggio = "La temperatura dell'acquario e' troppo ALTA! Rilevata temperatura di " .
    $temperatura_acquario . "°C in data: " . date("Y-m-d H:i:s");
    mail('mail@mail.com','Temperatura acquario CRITICA',$messaggio);
}

?>
```

ti. Vi consigliamo di provare ad inserire più di un dato, così da popolare la tabella come mostrato in **Fig. 6**.

Questo tornerà utile quando dovrete testare il corretto funzionamento dello script che mostrerà il grafico (ne parleremo più avanti). È giunto il momento di salvare i dati nel database (**Listato 4**).

A questo scopo, dovrete innanzitutto usare la funzione `mysqli` per far connettere la vostra applicazione Web al Database Manager MySQL e, successivamente, selezionare il database di interesse. Con `if ($con = mysqli_connect($host, $username, $password))` viene tentata la connessione e, in caso di fallimento, viene mostrato l'errore *Unable to connect to database*.

Se invece va a buon fine, viene tentato il collegamento al database con `mysqli_select_db($con,$db)`; anche qui è previsto un messaggio di errore in caso di fallimento.

La funzione `mysqli` accetta anche il nome del database come quarto parametro; potreste quindi usare un'unica funzione

Listato 6

```
<html>
<head>
<title>Acquario</title>
</head>
<body>
<h1 align="center">ACQUARIO </h1>
<p align="center">- based on Fishino UNO - </p>

<?php

    $max_value = 100; //Quanti valori mostra al massimo nel grafico

    //----- TODO -----
    // -----
    $username="username";
    $password="password";
    $db="database";
    $tablename = "acquario";
    $host="host";
    //-----
```

Listato 7

```
$con = mysqli_connect($host, $username, $password);
mysqli_select_db($con, $db) or die('Errore connessione...');
```

per far connettere l'applicazione al Database Manager MySQL e selezionare un database: `mysqli_connect($host, $username, $password, $db)`.

La variabile `$con` è l'identificativo di connessione che servirà per comunicare con il database. Una volta avvenuta la connessione verrà mandata una query (una interrogazione di un data-

base) al database con la funzione `mysqli_query(identificativo, query)`, dove l'identificativo di connessione è `$con` (se mancante, verrà scelto l'ultimo identificativo usato) e la query è indicata come stringa in `$query`. Con questa query specifichiamo di inserire nella tabella `$tablename` (nel nostro caso, `acquario`) i valori di temperatura e orario nei campi

Listato 8

```
$q6 = "SELECT Temperatura, Timestamp FROM $tablename ORDER BY Timestamp DESC";

$i = 0;

$r6 = mysqli_query($con,$q6);
$n_rilevazioni = mysqli_num_rows($r6);
while((list($temp_water[$i], $temp_time[$i]) = mysqli_fetch_row($r6)) and $i<$max_value)
    $i = $i+1;

$i = $i - 1;

?>
```

Listato 9

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
```

```

<script type="text/javascript">
    google.load('visualization', '1', {packages: ['corechart', 'line']});
    google.setOnLoadCallback(drawBackgroundColor);

    function drawBackgroundColor() {
        var data = new google.visualization.DataTable();
        data.addColumn('date', 'Time');
        data.addColumn('number', 'Temp');

        data.addRows([
            <?php
                $j = 0;
                while($i >= 0 and $temp_water[$i] != 0)
                {
                    echo "[";
                    echo "new Date(";
                    $temp_date = date("Y,m-l,d,H,i,s", strtotime($temp_time[$i]));
                    echo $temp_date;

                    echo ")";
                    echo ",";
                    echo $temp_water[$i];
                    echo "]";
                    if ($i > 0)
                    {
                        echo ",";
                    }
                    $j = $j + 1;
                    $i = $i - 1;
                }

            ?>

        ]);

        var options = {
            hAxis: {
                title: 't'
            },
            vAxis: {
                title: 'gradi'
            },
            focusTarget: 'category',
            curveType: 'function',
            lineWidth: 3,
            title: 'Temperatura acquario degli ultimi giorni',
            backgroundColor: 'white'
        };

        var chart = new google.visualization.LineChart(document.getElementById('chart_div'));
        chart.draw(data, options);
    }

</script>

<!-- Portions of this code are modifications based on work created and shared by Google and used according to
terms described in the Creative Commons 3.0 Attribution License. Code samples are licensed under the Apache
2.0 License. -->

```

del database. Con la funzione **now()** viene ritornato l'orario attuale del Web server nel formato YYYY-MM-DD HH:MM:SS. Siccome nella definizione della tabella del database avete impostato per il campo Timestamp **CURRENT_TIMESTAMP** come valore predefinito, anche passando **NULL** al posto di **now()**

non sorgerebbero problemi. Potreste aggiungere la possibilità di ricevere una e-mail in caso di temperature critiche (un esempio è in **Fig. 7**). Questo può essere fatto con la funzione mail, la cui sintassi è: *bool mail (string \$a , string \$oggetto , string \$messaggio [, string \$header_addizionali [, string \$parametri_addizionali*

]]). Tralasciando i parametri opzionali (tra parentesi quadre) è evidente come questa funzione permetta di inviare automaticamente il messaggio *\$messaggio* all'indirizzo *\$a* con oggetto della e-mail *\$oggetto*. La funzione può essere chiamata quando la temperatura è fuori da un certo range (nell'esempio, sotto i 23°C

Listato 11

```
<p align="right"><?php echo "Dati rilevati: #" . $n_rilevazioni; ?> </p>
<p>&nbsp;</p>

<?php
    echo "<br /><br /><strong>Ultima temperatura rilevata: " . $temp_water[0] . "&deg;C in data: " .
    $temp_time[0] . "</strong>";
?>

</body>
</html>
```

e sopra i 29°C).

GRAFICO DELLA TEMPERATURA

Per il grafico sul quale visualizzare le ultime temperature rilevate, la soluzione più semplice è sfruttare le molte librerie reperibili sul web; quella da noi utilizzata è la soluzione offerta da Google, che mette a disposizione gratuitamente un gran numero di grafici ottimizzati per i vari dispositivi, personalizzabili in molti aspetti e facili da utilizzare. I *Google charts*, la cui documentazione è reperibile all'indirizzo web <https://developers.google.com/chart/>, sono liberamente utilizzabili e forniti sotto licenza Creative Commons Attribution 3.0, mentre gli esempi di codice sono concessi sotto licenza Apache 2.0. Il grafico che vi mostriamo nel nostro esempio è scelto dalla

categoria "Line Charts", con sfondo bianco, linea del grafico blu e con andamento curvilineo. Sull'asse delle ascisse rappresentiamo l'andamento del tempo, mentre sull'asse delle ordinate vi sarà la temperatura. Ogni singolo dato avrà poi un'etichetta, con data e temperatura in formato testuale, che comparirà al passaggio del cursore del mouse (o toccando lo schermo, per i dispositivi touch). Anche per visualizzare il grafico dovrete realizzare uno script PHP, chiamando il file *acquario_grafico.php*. Iniziate (**Listato 6**) con i tag di apertura di una normale pagina HTML (html, head, title, body) e aprite poi il tag PHP con *<?php*. Con la variabile *\$max_value* definite quanti dati mostrare nel grafico: sceglietene il valore considerando che verrà inviato e caricato un dato ogni

30 minuti quindi; per vedere il grafico di un giorno intero, il valore della variabile dovrà essere almeno pari a 48. Nel nostro esempio poniamo questo valore pari a 100, tanto da coprire poco più di 2 giorni.

Infine inserite i parametri di connessione al database come già avete fatto per il file *in.php*. Allo stesso modo facciamo collegare l'applicazione web al database: ciò si può fare come nel file *in.php* oppure in modo più abbreviato, come mostrato nel **Listato 7**.

Per prelevare le temperature dal database dovremo impostare la query *\$q6*; stavolta specifichiamo che vogliamo recuperare le temperature con relative date dalla tabella *acquario*, ordinate a partire dalla più recente. Dopo aver interrogato il database con la funzione *mysql_query*

File	Dimensione	Data
acquario_grafico.php	3 KB	08/11/2015
in.php	1 KB	08/11/2015

Fig. 8 - Contenuto del "Gestore file" del sito web.

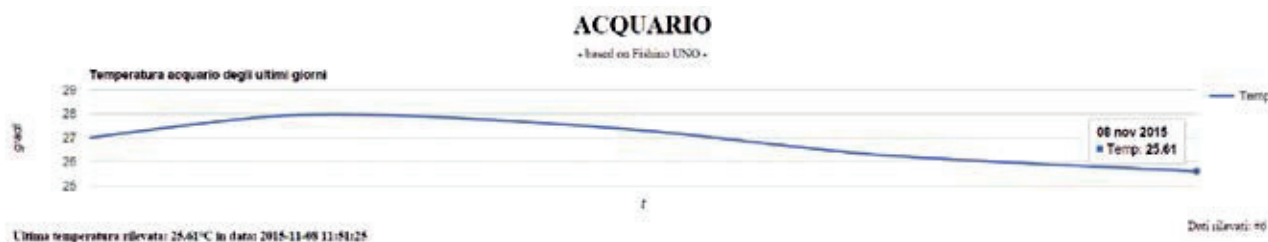


Fig. 9 - Aspetto del grafico con focus su uno dei dati.

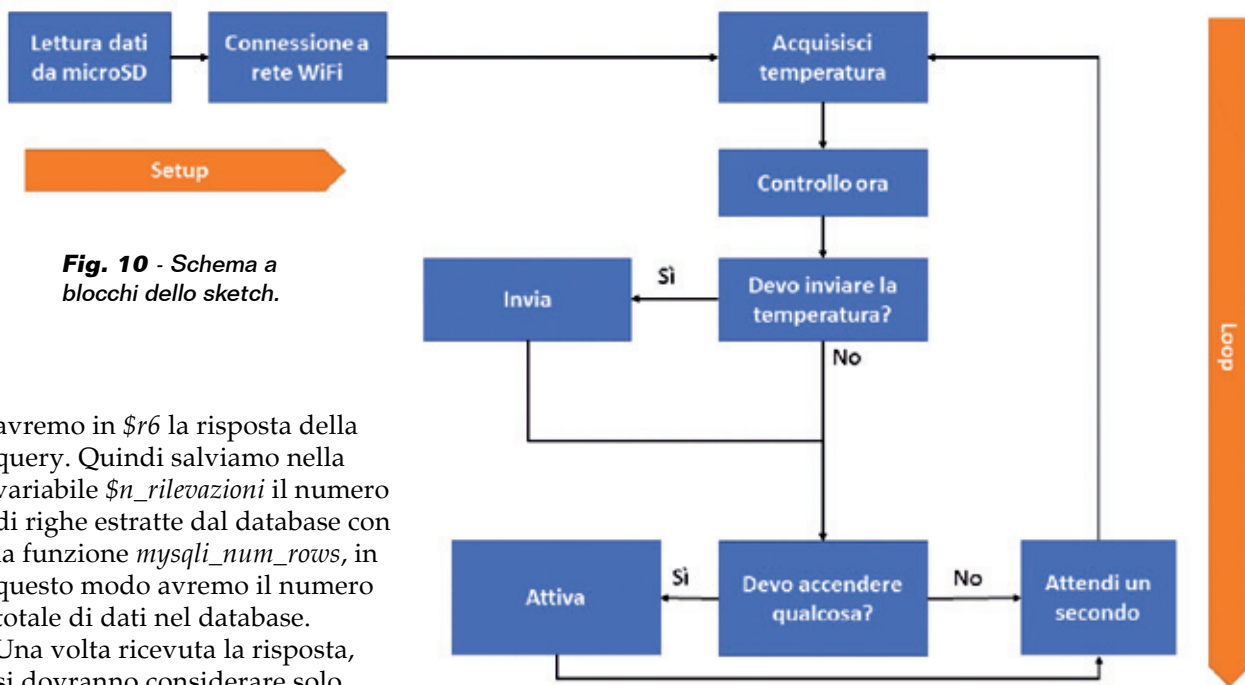


Fig. 10 - Schema a blocchi dello sketch.

avremo in `$r6` la risposta della query. Quindi salviamo nella variabile `$n_rilevazioni` il numero di righe estratte dal database con la funzione `mysql_num_rows`, in questo modo avremo il numero totale di dati nel database.

Una volta ricevuta la risposta, si dovranno considerare solo le prime 100 righe di ciò che il database ha restituito. Per fare ciò, usate la funzione `mysql_fetch_row(risultato)` che ha come parametro il risultato dell'interrogazione del database (la variabile `$r6` nel nostro caso). Questa funzione carica in un array una riga del risultato associato all'identificativo specificato, oppure restituisce NULL se non ci sono più righe.

Ciascun elemento di questo array sarà formato dalla coppia temperatura-data. Usate il costrutto `list()` per assegnare il valore dei due campi a due variabili rappresentate ciascuna da una cella di un array (`$temp_water[$i]` e `$temp_time[$i]`, dove `$i` è una variabile inizializzata a 0 e incrementata di 1 ad ogni ci-

clo). Questo costrutto è usato per assegnare valori a una lista di variabili in una sola operazione. Con il ciclo `while` nel **Listato 8** viene specificato che, per ciascuna riga restituita dal database, si deve salvare la temperatura e la data nei due array indicati rappresentanti rispettivamente il vettore delle temperature e

quello delle date. Se il numero di righe restituite è maggiore del massimo degli elementi voluti, il ciclo termina quando si sono esauriti gli elementi; se invece il numero di righe è maggiore, allora il ciclo termina quando la variabile contatore `$i` raggiunge il valore massimo. Una volta fuori dal ciclo `while`,

Listato 12

```

File myFile;

//Pin DATA di DS18B20: pin 2
OneWire ds(2);

// Initialize the Fishino client library
FishinoClient client;

//Credenziali rete WiFi
#define MY_SSID    "your_network"
#define MY_PASS    "your_password"

/* Definisco i pin di attivazione dei relé.
 * BOLLE (Aeratore) = Pin 8
 * PLAFO (Plafoniera + CO2) = Pin 9
 */
#define BOLLE 8
#define PLAFO 9

//Numero massimo di fasce orarie nel file contenuto nella microSD
#define MAX_VAL 4

RTC_DS1307 RTC;

//Matrice usata per memorizzare gli orari letti da microSD
int orari_bolle_plaf[MAX_VAL][7];
  
```

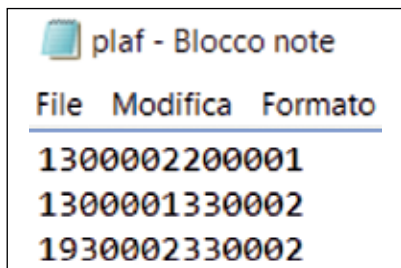


Fig. 11 - Esempio di file `plaf.txt`.

Listato 13

```
void setup() {
  Serial.begin(115200);

  //Inizializzo gli elementi della matrice a 0
  for (int t = 0; t < MAX_VAL; t++)
    for (int j = 0; j < 7; j++)
      orari_bolle_plaf[t][j] = 0;

  //Leggo il file nella microSD e inserisco gli orari nella matrice
  leggi_SD();

  //Relé non eccitati
  pinMode(PLAFO, OUTPUT);
  pinMode(BOLLE, OUTPUT);
  digitalWrite(PLAFO, LOW);
  digitalWrite(BOLLE, LOW);

  // only for Leonardo needed
  // necessario solo per la Leonardo
  while (!Serial)
    ;
  // initialize SPI
  // inizializza il modulo SPI
  SPI.begin();
  SPI.setClockDivider(SPI_CLOCK_DIV2);

  // reset and test WiFi module
  // resetta e testa il modulo WiFi
  while (!Fishino.reset())
    Serial << F("Fishino RESET FAILED, RETRYING...\n");
  Serial << F("Fishino WiFi RESET OK\n");

  // go into station mode
  // imposta la modalità stazione
  Fishino.setMode(STATION_MODE);

  // Initialize the Wifi connection.
  // Inizializza la connessione WiFi
  Serial << F("Setting up the Wifi connection...\n");

  // try forever to connect to AP
  // tenta la connessione finchè non riesce
  while (!Fishino.begin(MY_SSID, MY_PASS) == STATION_GOT_IP)
    Serial << F("ERROR CONNECTING TO AP, RETRYING...\n");

  Serial << F("SUCCEFULLY CONNECTED TO AP\n");

#ifdef IPADDR
  Fishino.config(ip);
#else
  Fishino.startDHCP();
#endif
  delay(1000);
}
```

sarà necessario decrementare di 1 la variabile *\$i* in modo da avere l'indice dell'array *\$temp_water* riferito all'ultimo valore inserito (se le righe restituite sono più di 100, ovviamente il valore di *\$i* sarà 100). In questo modo avete due array: uno contenente tutte le temperature da rappresentare nel grafico, l'altro con le date relative all'acquisizione. È il momento di inserire questi valori in un grafico; per sfruttare quello

offerto da Google aggiungete le librerie necessarie (**Listato 9**). Successivamente basta impostare qualche parametro legato all'aspetto del grafico e fornire i dati da rappresentare. L'aspetto più interessante è il passaggio dei dati, che avviene sotto forma di tabella dove le colonne costituiscono i campi da rappresentare nell'ordine ascisse e ordinate, mentre le righe sono i dati veri e propri.

Nel **Listato 10** vediamo infatti la variabile *data*, rappresentante i dati, che permette di aggiungere colonne con il metodo *addColumn(tipo, nome)* e righe con il metodo *addRows*.

Per evitare di dover aggiungere manualmente una riga per ogni elemento si può sfruttare un ciclo che aggiunga iterativamente righe separate da virgola fino a che non arriva al penultimo elemento, dopo il quale la virgola non è più necessaria. Alla fine del ciclo si avrà un risultato del tipo: *data.addRows([[data, temperatura], [data, temperatura], [data, temperatura]])*. Associando il grafico al contenitore *chart_div*, sarà possibile mostrarlo in output:

```
<div id="chart_div"></div>
```

Prima di vedere il risultato finale vi mostriamo come sia possibile visualizzare in output il numero totale di dati nel database e l'ultima temperatura rilevata con il relativo orario (**Listato 11**).

Una volta salvato il file (ricordatevi di chiudere tutti i tag aperti), caricatelo sul vostro sito web tramite il gestore file (**Fig. 8**). Il grafico sarà poi visibile (**Fig. 9**) digitando ad esempio *www.mio-sito.it/acquario_temperatura.php*.

LO SKETCH

Passiamo ora a schematizzare in pochi passi la logica dello sketch e lo facciamo con l'aiuto della **Fig. 10**. Inizialmente si includono le necessarie librerie:

```
#include <Wire.h>
#include <RTCLib.h>
#include <Flash.h>
#include <FishinoUdp.h>
#include <FishinoSockBuf.h>
#include <Fishino.h>
#include <SPI.h>
#include <SD.h>
```


Listato 14

```

/* Funzione che legge il file plaf.txt nella microSD
 * e ne estrae gli orari inserendoli nella matrice
 * degli orari.
 * CS - pin 4*/
void leggi_SD()
{
    String orari = "";
    if (!SD.begin(4)) {
        return;
    }
    int i;
    char c;
    myFile = SD.open("plaf.txt");
    if (myFile) {
        i = 0; //Numero di righe trovate nel file
        while (myFile.available()) {
            c = myFile.read();
            orari += c;
            if (c == '\n') //C'è ancora una riga
                ++i;
        }
        myFile.close();
    }
    int x = 0;
    int cont = 0;
    for (int j = 0; j <= i; j++)
    {
        while (orari[cont] != '\n')
        {
            switch (x)
            {
                case 0:
                    orari_bolle_plaf[j][0] += (orari[cont] - 48) * 10; //ORE_ON
                    break;
                case 1:
                    orari_bolle_plaf[j][0] += (orari[cont] - 48);
                    break;
                case 2:
                    orari_bolle_plaf[j][1] += (orari[cont] - 48) * 10; //MINUTI_ON
                    break;
                case 3:
                    orari_bolle_plaf[j][1] += (orari[cont] - 48);
                    break;
                case 4:
                    orari_bolle_plaf[j][2] += (orari[cont] - 48) * 10; //SECONDI_ON
                    break;
                case 5:
                    orari_bolle_plaf[j][2] += (orari[cont] - 48);
                    break;
                case 6:
                    orari_bolle_plaf[j][3] += (orari[cont] - 48) * 10; //ORE_OFF
                    break;
                case 7:
                    orari_bolle_plaf[j][3] += (orari[cont] - 48);
                    break;
                case 8:
                    orari_bolle_plaf[j][4] += (orari[cont] - 48) * 10; //MINUTI_OFF
                    break;
                case 9:
                    orari_bolle_plaf[j][4] += (orari[cont] - 48);
                    break;
                case 10:
                    orari_bolle_plaf[j][5] += (orari[cont] - 48) * 10; //SECONDI_OFF
                    break;
                case 11:
                    orari_bolle_plaf[j][5] += (orari[cont] - 48);
                    break;
                case 12:
                    orari_bolle_plaf[j][6] += (orari[cont] - 48); //1 (plafoniere) o 2 (bolle)
                    break;
            }
            ++x;
            ++cont;
        }
        ++cont;
        x = 0;
    }
    orari = "";
}

```

```

#include <OneWire.h>
#include <avr/io.h>
#include <avr/wdt.h>

```

Molte di esse saranno già presenti se avete scaricato l'IDE di Arduino mentre altre, come quelle di Fishino (scaricabili dal sito www.fishino.it o www.fishino.com), RTCLib.h per usare il modulo Real Time Clock (scaricabile da <https://github.com/adafruit/RTCLib>) e OneWire (scaricabile ad esempio da <http://playground.arduino.cc/Learning/OneWire>) dovranno essere aggiunte nella directory *Libraries* di Arduino.

Successivamente andranno definite alcune variabili (**Listato 12**): va dichiarato, ad esempio, il pin al quale è collegata la sonda DS18B20 (nel nostro caso il 2), va inizializzata la libreria client di Fishino e va definito il nome della rete WiFi con relativa password (*MY_SSID* e *MY_PASS*). Si dovranno poi definire i pin con i quali si attiveranno e disattiveranno i due relé. Con *MAX_VAL* potete specificare quante fasce orarie massime volete considerare, non è consigliabile inserire un valore alto per poi inserire pochi orari. Ad esempio se vorrete accendere una sola volta la plafoniera di illuminazione e, magari, due volte l'aeratore, allora potrete mettere come *MAX_VAL* un valore poco più grande di 3 (ad esempio 4 o 5) così da non sprecare memoria e da poter aggiungere in futuro qualche fascia oraria senza ricompilare lo sketch ma semplicemente modificando il file sulla microSD.

Infine, sarà usata una matrice *orari_bolle_plaf[MAX_VAL][7]* per poter salvare gli orari contenuti nella microSD in modo da non doverla aprire ad ogni

Listato 15

```
// Funzione per calcolare la temperatura dell'acqua in gradi Celsius tramite il DS18B20
float getTemp() {
    byte data[12];
    byte addr[8];

    if ( !ds.search(addr)) {
        //no more sensors on chain, reset search
        ds.reset_search();
        return -1000;
    }

    if ( OneWire::crc8( addr, 7) != addr[7]) {
        return -1000;
    }

    if ( addr[0] != 0x10 && addr[0] != 0x28) {
        return -1000;
    }

    ds.reset();
    ds.select(addr);
    ds.write(0x44, 1); // start conversion, with parasite power on at the end

    byte present = ds.reset();
    ds.select(addr);
    ds.write(0xBE); // Read Scratchpad

    for (int i = 0; i < 9; i++) { // we need 9 bytes
        data[i] = ds.read();
    }

    ds.reset_search();

    byte MSB = data[1];
    byte LSB = data[0];

    float tempRead = (MSB << 8) | LSB; //using two's compliment
    float TemperatureSum = tempRead / 16;

    return TemperatureSum;
}
```

Listato 16

```
/* Funzione per decidere se il tempo attuale sia in un intervallo o meno.
 * Accetta 6 parametri:
 * - Prima ora
 * - Primo minuto
 * - Seconda ora
 * - Secondo minuto
 * - Ora attuale
 * - Minuto attuale
 * Restituisce true o false se il calcolo ha esito affermativo o meno*/
boolean ora_giusta(int ora1, int minuto1, int ora2, int minuto2, int ora_ adesso, int minuto_ adesso)
{
    //Considero subito una risposta affermativa
    boolean risposta = true;

    //Vado a cercare se sono fuori dall'intervallo
    if (ora_ adesso < ora1 || ora_ adesso > ora2)
        risposta = false;
    if (ora1 == ora2 && minuto1 == minuto2)
        risposta = false;
    if (ora_ adesso == ora1 && minuto_ adesso < minuto1)
        risposta = false;
    if (ora_ adesso == ora2 && minuto_ adesso >= minuto2)
        risposta = false;
    if (ora1 > ora2 || (ora1 == ora2 && minuto1 > minuto2))
        risposta = false;

    return risposta;
}
```

ciclo, ma solo una volta sola. Questa matrice avrà tante righe quante sono le fasce orarie (le restanti saranno poste a zero) e le colonne indicheranno rispettivamente:

1. ora di accensione;
2. minuto di accensione;
3. secondo di accensione;
4. ora di spegnimento;
5. minuto di spegnimento;
6. secondo di spegnimento;
7. carattere per distinguere tra illuminazione ed erogatore di anidride carbonica o aeratore.

SETUP()

Nel **Listato 13** vediamo l'inizializzazione di Fishino nel *Setup*: si può notare come Fishino sia in modalità stazione. La matrice degli orari è inizialmente riempita di zeri ma, tramite la funzione **leggi_SD()**, viene progressivamente popolata dagli orari definiti da microSD.

LEGGI_SD()

La microSD è letta tramite la funzione **leggi_SD()** che non necessita di alcun parametro e si limita ad aprire il file *plaf.txt* (**Fig. 11**) contenuto nella microSD. All'interno di questo file saranno contenuti gli orari di accensione e spegnimento dei due relé nel formato *HHMMSSHHMMSSn* dove *n* è un carattere 1 o 2 a seconda che si tratti di un orario riferito al relé dell'illuminazione e dell'anidride carbonica oppure di quello dell'aeratore. Se, ad esempio, volessimo accendere l'aeratore tra le 10 e le 11 del mattino, dovremmo scrivere: *1000001100001*. Aggiungete una nuova riga di testo se e solo se volete aggiungere un nuovo orario, non lasciate altrimenti righe vuote. Una volta letti tutti i caratteri del file *plaf.txt* e averli inseriti nella stringa *orari*, questa viene analizzata e, in base alla posizio-

```

void loop() {
  //Imposto il WatchDog a 8 secondi
  wdt_enable(WDTO_8S);
  // server address:
  char server[] = "www.miosito.it";
  float temperatura = 0.0;
  while (temperatura < 1 || temperatura > 84) //Evita di stampare dati impossibili
    temperatura = getTemp();
  //Inizializza stringa da inviare al server
  String temperature = "";
  //Data attuale
  DateTime now = RTC.now();
  //Queste variabili non sono necessarie, sono usate solo per comodità
  int ore = now.hour();
  int minuti = now.minute();
  int secondi = now.second();
  int anno = now.year();
  //Ad ogni inizio ora e ogni mezza...
  if ((minuti == 0 || minuti == 30) && (secondi == 0 || secondi == 1))
  {
    // close any connection before send a new request.
    // This will free the socket on the WiFi module
    // chiude ogni ventuale connessione prima di inviare una nuova richiesta
    // quest libera il socket sull modulo WiFi
    client.stop();
    delay(1000);
    //Preparo la stringa da inviare al server
    temperature = "GET /in.php?t=";
    temperature += temperatura;
    temperature += " HTTP/1.1";
    // if you get a connection, report back via serial:
    // se ottiene una connessione, stampa i dati sulla seriale
    if (client.connect(server, 80))
    {
      //Serial.println("connected to server");
      // Make a HTTP request:
      client.println(temperature);
      //TODO
      client.println("Host: www.miosito.it");
      client.println("User-Agent: ArduinoWiFi/1.1");
      client.println("Connection: close");
      client.println();
    }
  }
  /* Variabili per decidere se accendere o spegnere i relé:
  * false = spento
  * true = acceso
  * Inizialmente li considero spenti*/
  boolean bolle_on = false, plaf_on = false;
  /*Scorro la matrice degli orari per controllare se deve essere acceso qualcosa
  * (nel caso si trovi qualcosa da accendere pongo a true le variabili bolle_on o plaf_on)*/
  for (int k = 0; k < MAX_VAL; k++)
  {
    if (orari_bolle_plaf[k][0] + orari_bolle_plaf[k][1] + orari_bolle_plaf[k][3] + orari_bolle_plaf[k][4] != 0)
    {
      if (ora_giusta(orari_bolle_plaf[k][0], orari_bolle_plaf[k][1], orari_bolle_plaf[k][3], orari_bolle_plaf[k][4], ore, minuti) == true)
      {
        switch (orari_bolle_plaf[k][6])
        {
          case 1:
            plaf_on = true;
            break;
          case 2:
            bolle_on = true;
            break;
        }
      }
    }
  }
  //Accesi (true) o spenti (false)?
  if (plaf_on == false)
    digitalWrite(PLAFO, LOW);
  else
    digitalWrite(PLAFO, HIGH);
  if (bolle_on == false)
    digitalWrite(BOLLE, LOW);
  else
    digitalWrite(BOLLE, HIGH);
  //Resetto il Fishino una volta al giorno
  if (ore == 23 && minuti == 59 && (secondi >= 0 || secondi <= 5))
    delay(15000); //Lascio scadeere gli 8 secondi del WatchDog
  //Attendo un secondo
  delay (1000);
  //Resetto il contatore del WatchDog
  wdt_reset();
}

```


ne del carattere in esame, viene inserito in una cella della matrice il valore calcolato differenziando tra decine e unità (**Listato 14**). Ad esempio la stringa "34" viene trasformata nel numero 34 moltiplicando il 3 per 10, aggiungendolo al 4, il tutto dopo aver convertito da *char* a *intero*.

GETTEMP()

La funzione **getTemp()**, nel **Listato 15**, non è stata modificata dalla funzione che si trova anche nella libreria OneWire. Il suo scopo è quello di calcolare la temperatura in base a quanto misurato dal sensore DS18B20. La funzione restituisce un float rappresentante la temperatura in gradi Celsius.

ORA_GIUSTA

La funzione **ora_giusta** (**Listato 16**) accetta come parametri tre orari formati da ore e minuti e controlla se il terzo orario si trova tra i primi due. La funzione non controlla se i due orari sono invertiti ma si potrebbe aggiungere tale opzione. In ogni caso è sufficiente rispettare l'ordine con il quale li si inserisce. Inizialmente, nella funzione, la variabile *risposta* è posta a true. Successivamente si verifica se l'orario in esame si trovi fuori dall'intervallo analizzato, in caso affermativo la variabile *risposta* è posta a false. Alla fine dei confronti viene restituito il valore di *risposta*. Se tale valore è rimasto a true allora significa che l'orario analizzato si trova tra gli altri due.

LOOP()

Passiamo ora ad analizzare il corpo principale dello sketch, la funzione **Loop()** (**Listato 17**). All'inizio del ciclo viene impostato il WatchDog a 8 secondi così

da resettare Fishino in caso di blocchi o cicli infiniti causati da errori imprevedibili.

Viene poi definito il server al quale connettersi, basta indicare in *server[]* l'indirizzo del vostro sito Web sul quale avete caricato il file *in.php*.

In seguito viene letto il valore della temperatura dalla sonda, evitando errori di misura con un ciclo while.

Ogni 30 minuti deve avvenire la connessione al server per inviare la temperatura, quindi si chiudono le eventuali connessioni ancora aperte e si prepara la stringa da inviare.

La stringa sarà composta dal GET, che abbiamo già trattato, e dal percorso del file *in.php* sul sito Web (se lo avete messo nella directory radice sarà sufficiente inserire */in.php*) seguito dal campo *t* (quello che verrà usato dallo script PHP per recuperare il valore della temperatura) con relativo valore. Infine viene effettuata la connessione al Web server e inviata la temperatura con il metodo GET.

Ricordatevi di cambiare anche l'indirizzo host in *client.println("Host: www.miosito.it")* inserendo, ancora una volta, l'indirizzo del server.

Successivamente si scorre la matrice degli orari per decidere se i relé devono essere accesi o spenti e, per far questo, si usano due variabili *bolle_on* e *plaf_on* inizialmente poste a false che, se la funzione *ora_giusta* ritorna true, allora vengono poste a true. Alla fine del ciclo vengono eccitati solo i relé la cui variabile booleana relativa è posta a true, altrimenti vengono disattivati. Per sicurezza, una volta al giorno (alle 23:59) Fishino è resettato, così da azzerare tutti i contatori e non incorrere in blocchi inutili. Il reset è ottenuto facendo attende-

re 15 secondi così da far scadere il WatchDog.

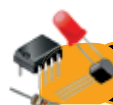
Questa parte può ovviamente essere evitata.

Viene poi resettato il contatore del WatchDog dopo aver atteso un secondo. Infine viene rieseguito il ciclo di loop().

Se non avete mai impostato l'orario dell'RTC di Fishino potete farlo facilmente caricando lo sketch *Softrtc* che trovate negli esempi della libreria *RTCLib*.

SVILUPPI FUTURI

Il sistema presentato svolge solo alcune delle funzioni che potrebbe compiere nel settore dell'acquariofilia; con qualche sensore in più e alcune modifiche al software, potreste ad esempio utilizzare il valore della temperatura per far gestire a Fishino l'aspetto del raffreddamento e del riscaldamento. Sulla base del file *plaf.txt*, potreste inserire un ulteriore file nella microSD e usarlo per definire il colore di strisce LED RGB per dare un tono allegro all'ambiente. ■



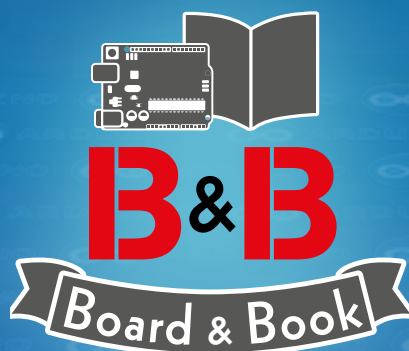
per il MATERIALE

Tutto il materiale necessario per realizzare questo progetto è disponibile presso Futura Elettronica. La board Fishino (cod. FISHINOONO) viene fornita montata e collaudata al prezzo di Euro 36,00. La sonda di temperatura waterproof con DS18B20 (cod. SONDADS18B20) è in vendita a Euro 7,00. Il modulo 1 relé 5 Vdc 10A viene fornito montato al prezzo di Euro 5,00. 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>

Vuoi entrare nell'affascinante mondo di **ARDUINO**
e imparare a utilizzarlo al meglio?

Allora non perderti questa
FANTASTICA OFFERTA!



Acquista insieme libro & scheda:
avrà uno **sconto di 4,50 euro!**



IVA INCLUSA

€ **29,90**

anziché

~~€ 34,40~~

cod. ARDUKITBOOK1



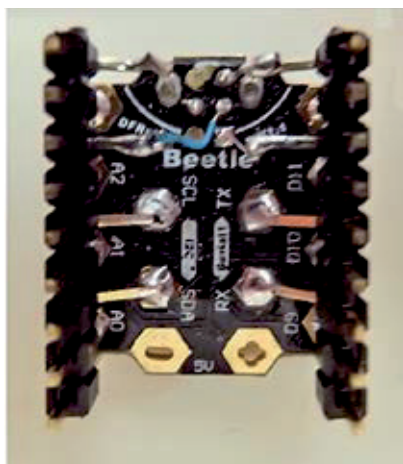
Evidenziamo un oggetto di cristallo o un brillante stimolandone suggestivi riflessi di luce grazie a tre spot a LED gestiti da Beetle, la più piccola board compatibile con Arduino.

LUCCICHINO: METTI IN VETRINA ARDUINO

di Ettore Massimo Albani

Quando si vuol mettere in evidenza un oggetto di cristallo sfaccettato o lavorato (uno Swarovsky, per esempio...) ma anche un gioiello in vetrina, l'effetto più suggestivo consiste nell'illuminarlo con una luce bianca pulsata, alla quale reagisce quasi prendendo vita ed emettendo un luccichio. Volevamo realizzare qualcosa del genere con dei semplici LED, tuttavia l'effetto, per risaltare, richiede l'illuminazione dell'oggetto da parte di più fonti luminose

poste con un'angolazione idonea rispetto all'area da illuminare. Inizialmente si pensava di usare i classici spot a luce alogena o a LED, ma il problema è che il cono di luce deve avere un angolo di diffusione molto piccolo, allo scopo di concentrare la luce sul target. Infatti, considerando la classica apertura di 20° dei LED con lente in plastica, già dopo solo un metro il cono di luce avrebbe un diametro di ben 35 cm; per non parlare della diminuzione di illumina-



La scheda Beetle modificata.

mento (i famosi lux...) che decresce, a parità di flusso luminoso, all'aumentare dell'area su cui la luce viene diffusa. L'alternativa consisteva nell'usare un sistema di focalizzazione con lenti, ma risultava piuttosto costosa, perciò si è scelto di avvicinare lo spot all'oggetto e di impiegare spot a LED a stretto angolo di diffusione della luce.

Per l'elettronica di controllo abbiamo deciso di utilizzare l'immancabile Arduino, ma non la solita scheda UNO, bensì la più piccola derivata da Arduino oggi in commercio: la Beetle.

ARDUINO BEETLE

L'azienda americana DFROBOT (http://www.dfrobot.com/wiki/index.php/Beetle_SKU:DFR0282) che

l'ha ideata, si è basata sul microcontroller ATMEL ATmega32U4, lo stesso usato sulle schede Arduino Leonardo, Micro ed altre. Le dimensioni sono molto contenute (20 x 22 x 3,8 mm) e la sua forma ricorda quella di un insetto, da cui il nome Beetle, che in inglese significa maggiolino. Per contenere le dimensioni, il produttore ha ridotto il numero di pin a disposizione, ma, in compenso, nella Beetle è stata preservata la caratteristica principale delle schede tipo Leonardo e cioè la comunicazione USB con il PC, senza rinunciare alla porta UART del microcontroller.

Quindi, i pin accessibili oltre quelli di alimentazione sono 10, con la particolarità che i pin 9 e 10 sono anche configurabili come ingressi analogici A9 e A10. Tuttavia, i pin SCL, SDA, TX, RX, MISO, MOSI, SCK ed RST sono posti nella parte posteriore, il che rende difficile l'accesso, anche perché non hanno fori passanti. Si nota anche l'assenza di un regolatore di tensione a 5V, ma la Beetle è stata pensata per essere alimentata via USB e quindi per prelevare i 5 volt da tale connessione.

Comunque è anche possibile alimentare la scheda a batterie:

per esempio con tre 3 stilo da 1,5 volt, perché la minima tensione che ne garantisce il funzionamento è 4,5 V. La Beetle può essere alimentata anche con quattro batterie ricaricabili NiMH da 1,2 V in serie (per un totale di 4,8V). L'assorbimento del microcontrollore è attorno ai 15 mA, quindi è garantita una lunga autonomia. Un'altra opzione per l'alimentazione consiste nella serie di quattro stilo alcaline, che però forniscono 6 volt; considerato che la Beetle può funzionare fino a un massimo di 5,5 volt, è richiesta l'interposizione di un diodo al silicio (tipo 1N4004 o 1N4007), la cui caduta minima è 0,6 volt. La ridotta quantità di I/O accessibili sul bordo esterno e le piccole dimensioni della Beetle, nonché il basso costo, fanno intuire che la scheda è stata pensata per l'utilizzo in gadget e giocattoli tecnologici. Tuttavia non è precluso l'utilizzo in applicazioni più impegnative, perché con un piccolo sforzo è possibile dotare la Beetle di pin-strip per poterla saldare agevolmente su un PCB: lo abbiamo fatto in questo progetto e tra breve vi spiegheremo come. Nell'apportare questa modifica, ne abbiamo approfittato per aggiungere ben quattro

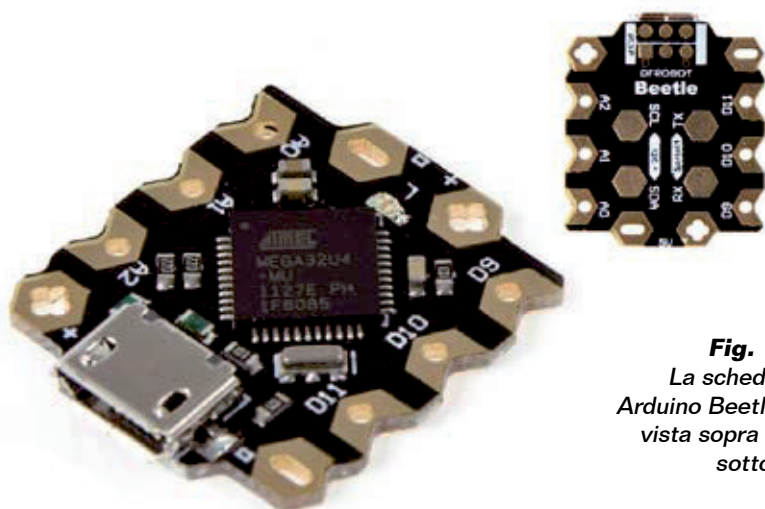


Fig. 1
La scheda Arduino Beetle vista sopra e sotto.

Tabella 1 - Caratteristiche della Beetle.

Microcontrollore	ATmega32u4
Frequenza di clock	16 MHz
Tensione d'alimentazione	5V (4,5 ÷ 5,5V)
Assorbimento	15 mA
Digital I/O	10
canali PWM	4
Input analogici	5
UART	1
I ² C-Bus	1
USB	1 (micro USB)
Flash Memory	32 kB (4 kB usati dal bootloader)
SRAM	2,5 kB
EEPROM	1 kB

pin digitali e un'uscita PWM (normalmente lasciati scollegati nella Beetle originale), collegando opportunamente i pin alle piazzole difficilmente accessibili cui accennavamo.

IL BEETLE "ELABORATO"

Nella **Tabella 2** trovate i segnali della Beetle confrontati con quello della Leonardo; in azzurro abbiamo evidenziato quelli da noi aggiunti con i pin-strip. Si noti che il pin 13, essendo collegato al LED blu sulla scheda, può essere utilizzato solo come uscita digitale, lo stesso limite di altre schede Arduino. Notate anche la presenza del pin AUX, che non è collegato ad alcunché, essendo una piazzola in più; tuttavia, sarebbe possibile collegarlo a un pin del microcontrollore per ottenere un ulteriore segnale: ad esempio un ingresso analogico a scelta tra A3 (pin 39), A4 (pin 40), A5 (pin 41).

Nella versione ufficiale della Beetle, le piazzole ICSP sono disposte in orizzontale (come mostrato a sinistra nella Fig. 3), ma esiste una nuova versione in cui le piazzole sono disposte in verticale (Fig. 3 a destra). In entrambe, la piazzola quadrata individua il pin 1 (MISO).

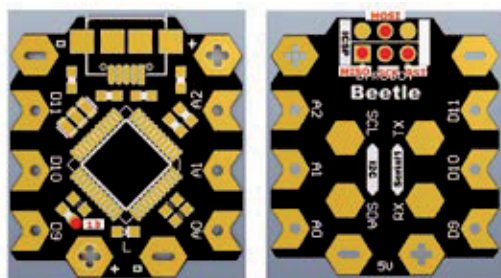


Fig. 3

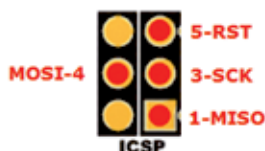


Fig. 2 - Schema elettrico della Beetle.

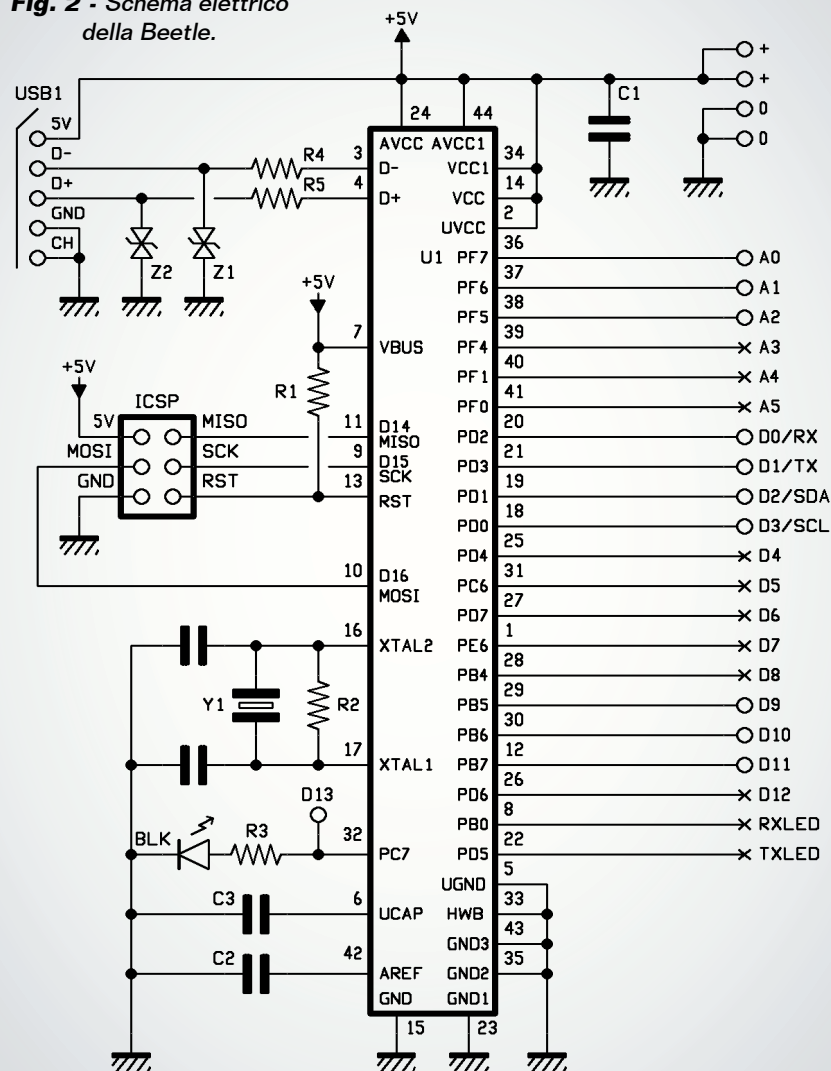


Tabella 2 - I/O della Beetle con evidenziati in azzurro quelli da noi aggiunti.

Beetle	Leonardo	PWM			Note
Pin	Pin	Timer	Register	Hz	
RX	0				Serial1
TX	1				
SDA	2				I ² C
SCL	3	3	OC3A	490	
D9	9-A9	1	OC1A	490	
D10	10-A10	1	OC1B	490	
D11	11	0	OC0A	977	
A0	A0				
A1	A1				
A2	A2				
D13	13	4	OC4A	490	*Solo come uscita ICSP pin 1 ICSP pin 3 ICSP pin 4 ICSP pin 5 *Difficile da saldare
MISO	14				
SCK	15				
MOSI	16				
RST	RESET				
AUX					

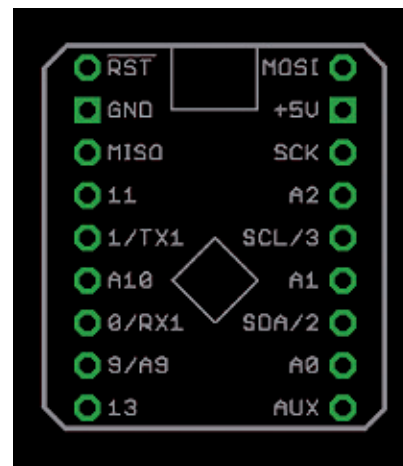
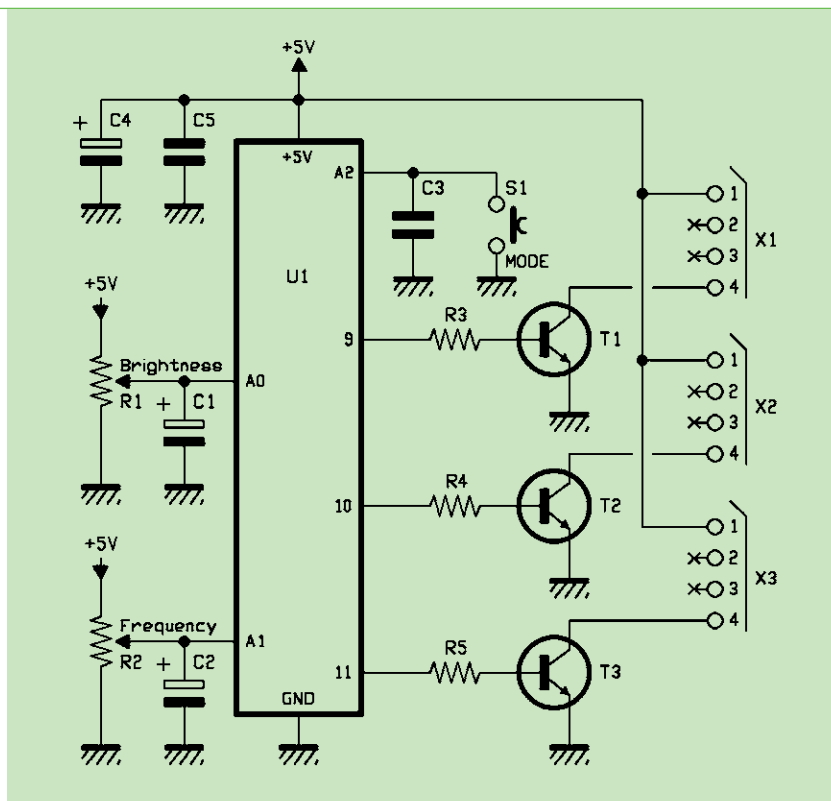


Fig. 4

Apportando le nostre modifiche (aggiunta e connessione dei pin-strip) la pin-out dell'Arduino Beetle diventa quella mostrata nella Fig. 4. Allo scopo occorre preparare due strip maschio/maschio da 9 pin con passo 100 mil (2,54 mm); si possono usare quelli standard (un lato lungo ed uno corto) oppure quelli con entrambi i lati lunghi: i primi sono indicate per la saldatura diretta sul PCB, mentre le seconde sono indicate se si intende usare due strip femmina da 9 pin con contatti a tulipano, per una facile rimozione del modulo.

Si noti che vengono montate al contrario di quanto normalmente si farebbe, cioè con il lato più lungo dei pin verso Beetle e quello più corto verso il PCB.

Poiché il passo delle piazzole della Beetle è 200 mil (5 mm), dobbiamo ripiegare all'interno alcuni pin. I pin ripiegati e corrispondenti alle piazzole SCL, SDA, TX e RX possono essere saldati direttamente. Per quelli corrispondenti ai pin ICSP (MISO, MOSI su un lato e SCK, RESET sull'altro)

bisogna aggiungere quattro spezzoni di filo sottile, ad esempio filo rigido da Wire Wrap da 30 AWG (sezione 0,5 x 0,25 mm). Adesso viene la parte più difficile: per il collegamento del pin 13 bisogna saldare uno spezzone di filo tra il pin 13 dello strip ed il

resistere SMD sul lato frontale di Beetle. Attenzione: quest'ultima saldatura risulta più impegnativa rispetto alle precedenti, a causa della estrema vicinanza tra loro dei pin del micro e si consiglia di farle solo se si ha esperienza col le saldature SMD.

Prima di dare tensione, assicuratevi con una lente che le saldature siano state fatte alla regola dell'arte e non ci siano sbavature di stagno; verificate anche che i pin-strip siano ben dritti e che la loro altezza sia almeno 15 mm.

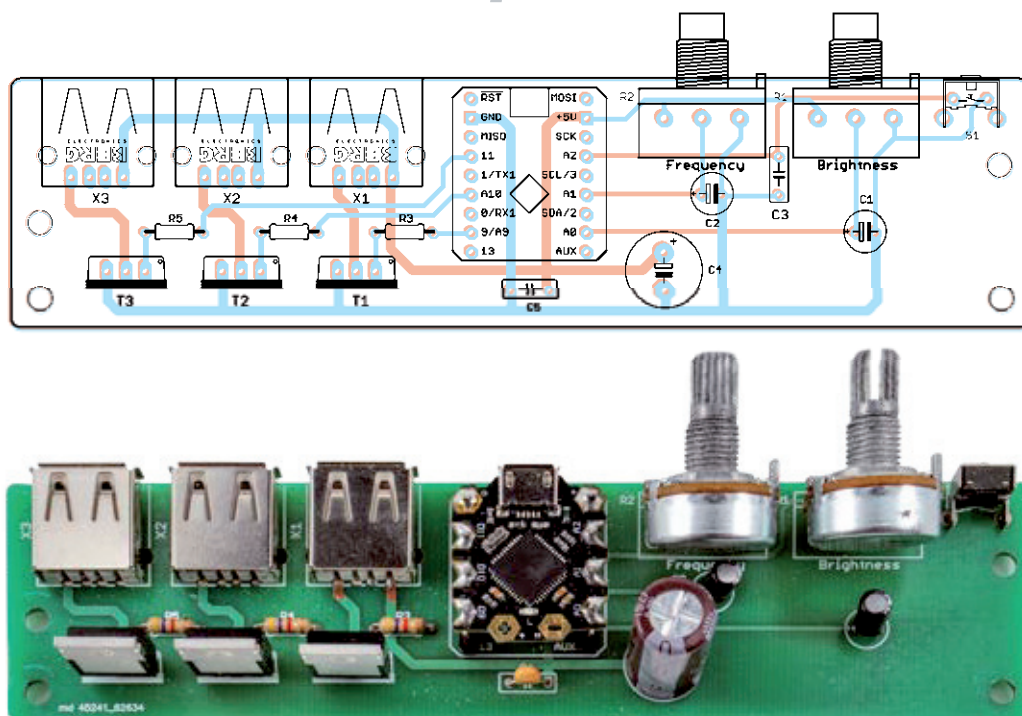
Fig. 5 - Esempi di spot a LED alimentati da USB.



Tabella 3 - Modalità di accensione dei LED.

Modalità	Lampeggi	Descrizione
Off	0	Tutti e tre i canali spenti.
Smooth	1	Come la modalità Normal, ma con rampa di accensione e spegnimento.
Normal	2	Due canali attivi alla volta che si alternano alla frequenza scelta.
Alternate	3	Solo un canale attivo per volta.

[piano di **MONTAGGIO**]



Elenco Componenti:

R1: Potenzimetro 10 kohm
 R2: Potenzimetro 10 kohm
 R3÷R5: 4,7 kohm
 C1: 10 μ F 63 VL elettrolitico
 C2: 10 μ F 63 VL elettrolitico

C3: 470 μ F 16 VL elettrolitico
 C4: 100 nF 63 VL poliestere
 C5: 100 nF 63 VL poliestere
 U1: Beetle DFR0282 (MF1238)
 T1÷T3: TIP122

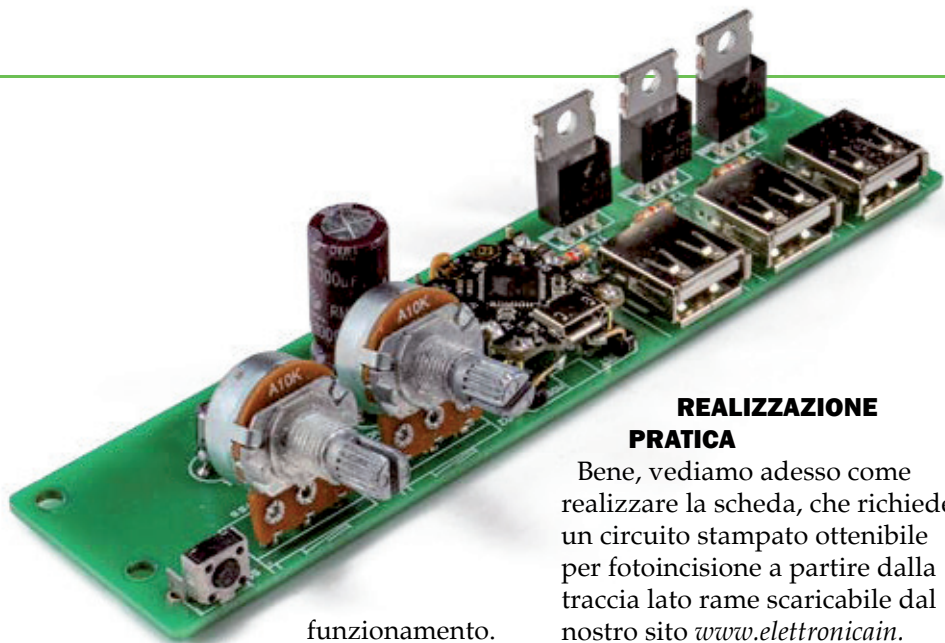
Varie:
 - Connettore USB-A
 Femmina Orizzontale
 da CS (3 pz.)
 - Circuito stampato S1238

SCHEMA ELETTRICO

Passiamo adesso alla nostra applicazione, dove la Beetle (siglata U1 nello schema elettrico, dove viene raffigurata come fosse un integrato) viene usata per la modulazione PWM della tensione fornita da tre canali che a loro volta pilotano dei piccoli spot a LED. Per l'esattezza, i piedini 9, 10 e 11 della scheda pilotano ciascuno -con un segnale PWM generato dal microcontrollore- la base di un Darlington NPN (di tipo TIP122) mediante un resistore di limitazione; i transistor sono collegati a emettitore comune ed hanno come carico ciascuno un LED di potenza. Per le connessioni di uscita abbiamo utilizzato tre prese USB di tipo

A (siglate X1, X2, X3), in quanto, per semplificare la realizzazione del sistema, abbiamo optato per delle semplici ed economiche lampade USB con braccio flessibile, del tipo usato per illuminare la tastiera dei PC notebook. Siccome i piedini del connettore USB che portano l'alimentazione sono i due esterni, il collettore di ogni transistor è collegato al 4 (corrispondente al negativo dei 5 volt nello standard USB) mentre all'1 (corrispondente al positivo) è connessa la linea di alimentazione della nostra scheda, che porta i 5 volt sia alle prese, sia alla Beetle. Ogni volta che uno dei transistor riceve sulla base l'impulso (1 logico) del segnale rettangola-

re PWM, passa dall'interdizione alla saturazione facendo scorrere corrente nel proprio collettore, in quanto la tensione Vce (collettore-emettitore) praticamente si annulla (diventa poche centinaia di millivolt) e quasi l'intera tensione di alimentazione si ritrova tra i piedini 1 e 4 del connettore USB corrispondente e quindi ai capi dello spot a LED. Il circuito è completato da due potenziometri siglati R1 e R2, alimentati dai 5 volt e collegati con i cursori, il primo al pin A0 e il secondo all'A1; la tensione fornita dai cursori, filtrata da un elettrolitico per ciascuna linea, viene letta dall'ADC del microcontrollore contenuto nella Beetle e imposta la modalità di



REALIZZAZIONE PRATICA

Bene, vediamo adesso come realizzare la scheda, che richiede un circuito stampato ottenibile per fotoincisione a partire dalla traccia lato rame scaricabile dal nostro sito www.elettronica.in.it. Inciso e forato lo stampato, disponetevi i componenti a partire da quelli a più basso profilo e terminate con la scheda Arduino Beetle, opportunamente modificata e provvista di pin-strip. Ricordate di far aderire bene al c.s. i tre connettori USB, stagnandone abbondantemente le alette di ancoraggio, dato che essi dovranno sostenere gli spot a LED, con tutto il loro peso. I Darlingston vanno in piedi, con il lato metallico rivolto al bordo della scheda e non richiedono

funzionamento.

Più esattamente, R1 permette di regolare la luminosità (Brightness) dei LED e R2 di registrare la frequenza del lampeggio (Frequency); la luminosità viene variata regolando il duty-cycle dei tre segnali PWM che alimentano i LED, mentre la frequenza è quella dei PWM. Alla linea A2, utilizzata in questo caso come input digitale con pull-up interno al microcontrollore, abbiamo connesso il pulsante S1, la cui funzione è consentirci di passare da uno all'altro tra gli effetti luminosi previsti dal firmware; il condensatore C3 permette l'antirimbombo, ossia evita che quando i contatti non sono precisi o cominciano ad usurarsi, il pulsante trasmetta più impulsi con una sola pigiata.

L'alimentazione per tutto il circuito viene prelevata dal connettore microUSB della Beetle e dal suo pin +5V passa al resto dei componenti; sulla scheda, i 5 volt vengono filtrati dai condensatori C4 e C5, che impediscono il rientro dei disturbi originati dalla commutazione dei transistor sugli spot a LED nella Beetle. Filtrare la linea di alimentazione a 5V ha anche lo scopo di eliminare l'eventuale ripple residuo dell'alimentatore switching e fornire una "riserva" di energia per la commutazione sui LED.

alcun dissipatore, perché sono capaci di sopportare una corrente di collettore di 5A ma qui vengono utilizzati per commutare un massimo di 500 mA, quindi dissipano una potenza che non li surriscalda neanche se lavorano privi di dissipatore.

Il circuito, con su montata la Beetle, va racchiuso in un contenitore a console o comunque che abbia un piano per appoggiare gli oggetti da illuminare. Passiamo adesso alla costruzione meccanica del nostro apparato, che, proprio perché destinato a una vetrina, abbiamo voluto inserito in un contenitore dotato di un ampio piano; si tratta di un box per elettronica, del tipo a leggio, realizzato in ABS e avente le dimensioni di 189 mm x 134 mm x 32/55 mm, con un ampio piano che permette di appoggiare anche gioielli di grandi dimensioni. La superficie del contenitore può essere ricoperta con un rettangolo di gomma antiscivolo oppure di velluto

Listato 1 – Porzione della Routine Loop dello sketch

```
// ----- LOOP - Digital Input -----

if (ButtonDebounce(ModePin, LOW, 50)) { // push button pressed

    Mode ++;
    Mode = Mode % 4; // 0 = off, 1 = smooth, 2 = normal, 3 = alternate
    digitalWrite(PWMBPin, LOW);
    digitalWrite(PWMBPin, LOW);
    digitalWrite(PWMBPin, LOW);
    delay(500);

    for (byte I = 0; I < Mode; I++) { // all LED Flash 200ms

        digitalWrite(PWMAPin, HIGH);
        digitalWrite(PWMBPin, HIGH);
        digitalWrite(PWMCPin, HIGH);
        delay(200);
        digitalWrite(PWMAPin, LOW);
        digitalWrite(PWMBPin, LOW);
        digitalWrite(PWMCPin, LOW);
        delay(200);

    }

    delay(500);
}
```

colorato, non solo per l'estetica, ma anche per impedire che il gioiello possa scivolare. È consigliabile usare un colore scuro (nero opaco o simili), per evitare che il riflesso della luce pulsata generata dai LED dia fastidio. Per il nostro prototipo abbiamo scelto un rettangolo di moquette rasata nera usata per il rivestimento di casse acustiche ed incollata con nastro biadesivo, in modo da facilitarne la sostituzione, se si rendesse necessaria. La scatola a leggio deve essere lavorata nella parte posteriore in modo da lasciar affacciare all'esterno le tre prese USB; richiede anche una finestrella per la presa micro USB della Arduino Beetle, due fori per gli alberini dei due potenziometri e, infine, un foro per il piccolo pulsante.

Contrariamente al nostro prototipo, consigliamo un modello di pulsante con un alberino di azionamento sporgente di almeno 10 mm, per risparmiarci l'uso di cacciaviti o stuzzicadenti quando dobbiamo premerlo.

Per quanto riguarda gli spot a LED, dobbiamo dire che di lampade USB del genere scelto (per illuminare la tastiera dei notebook) ce ne sono vari tipi, di solito con parecchi LED per avere un intenso flusso luminoso, ma noi abbiamo scelto un modello con un solo LED a luce concentrata e dotato di braccetto flessibile, in modo che non sia troppo vistoso né pesante; il peso è un fattore determinante perché se è eccessivo o se il braccetto è poco flessibile, si vanno a sforzare i connettori USB della scheda, tanto da strapparli dal PCB, con le ovvie conseguenze.

Durante le prove abbiamo avuto modo di verificare la qualità di prodotti di vario tipo e riscontrato che, tra campioni dello stesso modello e del medesimo costrut-

tore, esistono delle differenze di tonalità nel bianco e di intensità luminosa dei LED. Purtroppo i prodotti in commercio, tutti "made in China", riservano queste sorprese perché i LED utilizzati non sono particolarmente selezionati. Comunque i siti di e-commerce -ormai numerosissimi sul web- vi propongono una vastissima scelta di luci USB snodate, quindi valutate un po' voi i prodotti migliori o comunque quelli che ritenete più adatti alla vostra realizzazione. Rammentate solo che ogni luce non può assorbire oltre 500 mA.

L'alimentatore necessario a far funzionare il circuito dev'essere di quelli dotati di uscita USB: nel nostro caso è un 5V - 1,5A dotato di presa USB A, al quale va applicato un di cavetto USB/micro USB.

IL FIRMWARE

Il duty-cycle del segnale PWM, che decide la luminosità dei LED, viene controllato dal valore di tensione sull'ingresso analogico A0, fornito dal potenziometro R1. La frequenza di lampeggio (1÷50 Hz) viene controllata dal valore di tensione sull'ingresso analogico A1 generato dal potenziometro R2.

Il pulsante S1 commuta la modalità di funzionamento, ossia di combinazione dell'accensione dei tre spot a LED: ogniqualevolta lo si preme, si passa da una modalità alla successiva. Il firmware prevede tre modalità, oltre alla Off, nella quale la Beetle non genera alcun PWM. Il passaggio da una modalità all'altra è evidenziato dal numero di lampeggi di tutti e tre il LED, secondo quanto esposto nella **Tabella 3**. Si noti che alla partenza, il programma si avvia eseguendo la modalità Smooth e che la modalità Alternate presenta una

luminosità dimezzata rispetto alle altre, a causa dell'unico canale d'uscita attivo per volta. Il LED della Beetle, collegato all'uscita digitale 13, viene acceso per 10 millisecondi ogni volta che il ciclo inizia, allo scopo di verificare il funzionamento del programma. L'effetto migliore si ottiene in modalità Smooth, con una frequenza di lampeggio di circa 10 Hz.

Per controllare i valori, si può usare il Serial Monitor di Arduino, predisposto per una velocità di 9.600 baud. Si noti che l'attivazione del Serial Monitor rallenta un po' l'esecuzione del ciclo (allunga il tempo di ciclo). Inoltre, il Serial Monitor sull'IDE dovrebbe essere attivato solo per eseguire delle prove, perché a causa di un bug, se si apre il Serial Monitor od un altro emulatore di terminale e dopo lo si chiude, dopo un po' si ha un forte rallentamento del loop, che termina solo con un reset del microprocessore o togliendo l'alimentazione. Insomma, è come se Arduino rimanesse impegnato da Serial Monitor anche disattivandolo. ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista. La scheda Beetle montata (cod. DFR0282) è disponibile a 13,00 Euro presso Futura Elettronica. Il prezzo si intende 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>

GESTISCI E CONTROLLA LA TUA CASA CON I TELECONTROLLI GSM

ATTIVA E CONTROLLA DA REMOTO QUALSIASI DISPOSITIVO
ELETTRICO O ELETTRONICO: DALLA CENTRALINA ANTIFURTO
ALL'IMPIANTO DI IRRIGAZIONE, DALLE LUCI DI CASA
AL SISTEMA DI CONDIZIONAMENTO E RAFFRESCAMENTO.
GESTIONE TRAMITE CHIAMATA VOCALE O MESSAGGI SMS.

cod. TDG133

€ 79,00

Telecontrollo GSM bidirezionale 2 IN / 2 OUT

- 2 uscite a relè con funzionamento monostabile o bistabile e 2 ingressi d'allarme optoisolati.
- Dimensioni 103 x 67 x 28 mm
- Alimentazione 9÷32 Vdc/1 A
- Completo di antenna GSM esterna



**Disponibili
anche moduli
con funzione di:**

APRICANCELLO

cod. TDG134 • € 75,00

GSM



Tutti i telecontrolli vengono forniti già montati e collaudati e sono facilmente configurabili mediante SMS o in locale tramite PC (per il collegamento al PC è necessario il modulo opzionale d'interfaccia USB cod. FT782M • € 13,00).



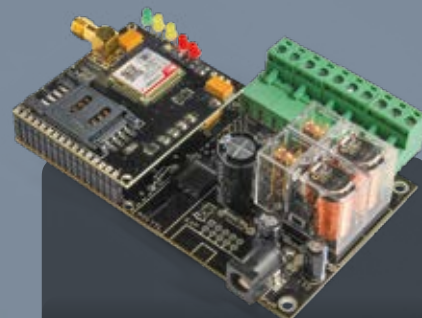
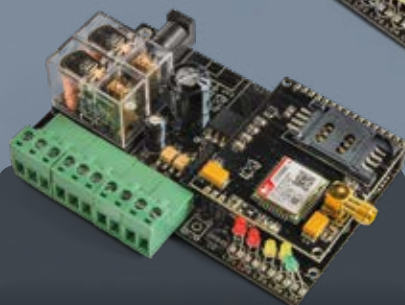
TERMOSTATO CON CONTROLLO GSM

cod. TDG139 • € 79,00



cod. TDG138 • € 79,00

TELEALLARME GSM CON ANTI JAMMER



cod. TDG140 • € 88,00

TELECONTROLLO GSM CON COMANDI DTMF

Contenitore in ABS
appositamente forato per
ospitare i telecontrolli GSM:
TDG133, TDG138 e TDG140.
Cod. BOXTDG • € 16,90

Disponibile anche per TDG134.
Cod. BOXTDG134 • € 16,90

Disponibile anche per
TDG139.
Cod. BOXTDG139 • € 16,90



Prezzi IVA inclusa.

**FUTURA
ELETTRONICA®**

Futura Group srl
Via Adige, 11 • 21013 Gallarate (VA)
Tel. 0331/799775

Caratteristiche tecniche
e vendita on-line su:

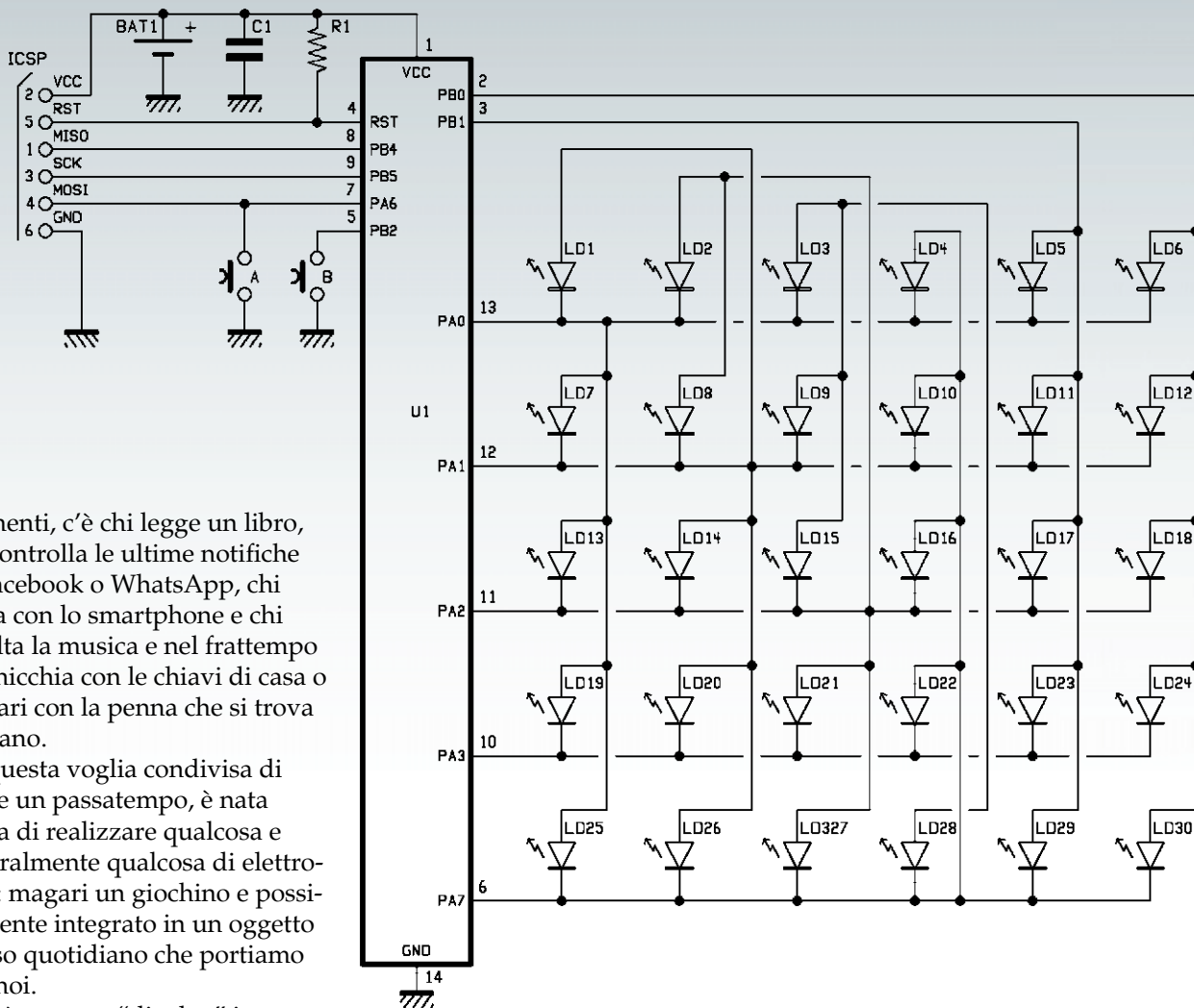
www.futurashop.it

Basato su un'architettura Arduino, realizza giochi elettronici e animazioni pilotando una matrice di LED secondo una speciale modalità di multiplex.

KEY CHAININO: PORTACHIAVI MILLEUSI

di ALESSANDRO MATERA

Nel mondo "accelerato" in cui viviamo, i momenti per fare ciò che vorremmo bisogna ritagliarli con abilità da equilibristi tra i mille impegni della giornata. Siamo sempre di fretta e gli unici attimi di relax che ci concediamo sono i tempi morti durante l'attesa del bus, mentre siamo in treno, aspettiamo la metropolitana o siamo in coda in banca o alla Posta. In questi



momenti, c'è chi legge un libro, chi controlla le ultime notifiche di Facebook o WhatsApp, chi gioca con lo smartphone e chi ascolta la musica e nel frattempo giochicchia con le chiavi di casa o magari con la penna che si trova in mano.

Da questa voglia condivisa di avere un passatempo, è nata l'idea di realizzare qualcosa e naturalmente qualcosa di elettronico: magari un giochino e possibilmente integrato in un oggetto di uso quotidiano che portiamo con noi.

Così è nato un “display” interamente realizzato con una matrice a LED controllata da un piccolo microcontrollore, compatibile con Arduino, integrato in un circuito stampato che ha forma e dimensioni di un portachiavi. Dopo aver preso forma, la nostra idea ha preso anche un nome: KeyChainino. Il nome è già indicativo di ciò che il circuito è: un portachiavi (keychain, in inglese...) basato su architettura Arduino. Si tratta infatti di un portachiavi elettronico programmabile con Arduino, dove chiunque può realizzare il proprio gioco-passatempo e non solo. KeyChainino è composto da una matrice di 6x5 LED e due pulsanti, il tutto controllato dal piccolo ATtiny84A, un microcontrollo-

re Atmel ad 8 bit in esecuzione SMD, che ha solamente 14 pin, quindi è sufficientemente miniaturizzato per i nostri scopi. Per pilotare tutti i LED della matrice, tuttavia, ci servirebbe un numero maggiore di I/O, cui vanno aggiunti quelli necessari a leggere lo stato dei pulsanti di gestione del gioco. Abbiamo però risolto questo problema con un artificio che vi spiegheremo tra breve, analizzando il circuito elettronico alla base del progetto.

SCHEMA ELETTRICO

Il tutto fa capo al microcontrollore Atmel, che vediamo (nello schema elettrico) circondato da

una matrice di LED collegata in un modo originale: non più a righe e colonne ma LED collegati in cascata tra due piedini del microcontrollore.

Questa particolare configurazione, che spiegheremo bene nei prossimi paragrafi, consente con soli 14 pin di comandare 30 LED, grazie a una tecnica nota come Charlieplexing; con il tradizionale multiplex il comando di 30 diodi avrebbe richiesto una matrice di 5x6, quindi $5+6=11$ piedini. Qui ce ne bastano 7.

Il micro funziona con l'oscillatore interno ed è per questo che nello schema elettrico non vedete alcun componente di temporizzazione come quarzi o condensatori.

tori; ciò ha consentito la massima semplificazione del circuito, che si riduce al microcontrollore, alla matrice di LED, alla resistenza di pull-up applicata al piedino RST (che a riposo -ossia nel normale funzionamento del micro- deve trovarsi a livello alto, ma che in programmazione, attraverso il connettore ICSP, deve poter essere commutata dal programmatore) e ad un condensatore di filtro dell'alimentazione, che in questo caso è di soli 3 volt ed è assicurata dalla pila a bottone CR2032. I pulsanti collegati alle linee PA6 (che in programmazione è la linea dati MOSI) e PB2 servono a gestire le varie funzionalità del gioco; i due I/O dispongono di un resistore di pull-up interno al microcontrollore che viene attivato dal firmware in fase di inizializzazione degli I/O.

Siccome la linea di alimentazione del microcontrollore è collegata a Vpp del connettore ICSP, quando si programma l'ATTiny84A bisogna togliere la pila dal suo alloggiamento, giacché la tensione che il programmatore fornirà sarà sicuramente maggiore. Detto ciò, passiamo al discorso più "succoso" su questo circuito, ossia il Charlieplexing: questa tecnica, che è una specie di multiplex, si basa sul fatto che un certo LED della matrice funziona soltanto quando la tensione ai suoi capi è di polarità tale da polarizzarlo direttamente; giocando su un opportuno collegamento e disponendo i LED come vedete nello schema elettrico, sfruttando il fatto che i pin del nostro microcontrollore possono assumere tre stati logici (ALTO/HIGH, BASSO/LOW, Alta impedenza/

Three-state) si riesce a comandare con ogni I/O più di un LED. Lo stato Three-state viene raggiunto quando programiamo il pin affinché funzioni come ingresso digitale, in quanto la linea di I/O assume uno stato di alta impedenza, che quindi limita fortemente il passaggio di corrente risultando quasi interrotta. Per farvi capire come funziona il Charlieplexing, proviamo ora ad analizzare lo schema di **Fig. 1**, che permette di pilotare sei LED con sole tre linee di I/O del microcontrollore; considerando che X1, X2 e X3 siano i pin interessati, ovvero tre di Arduino, e proviamo a far accendere il LED1. Per fare ciò, in Arduino, configuriamo i pin X1 e X2 -che sono collegati al LED che ci interessa accendere- come OUTPUT; portiamo X1 a livello logico ALTO

Fig. 1

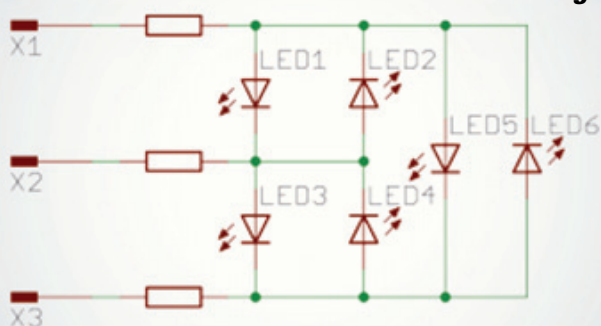


Fig. 2

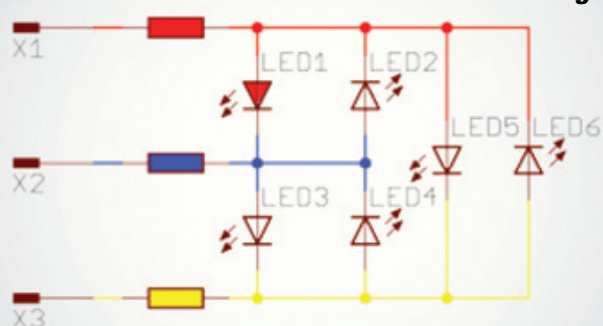


Fig. 3

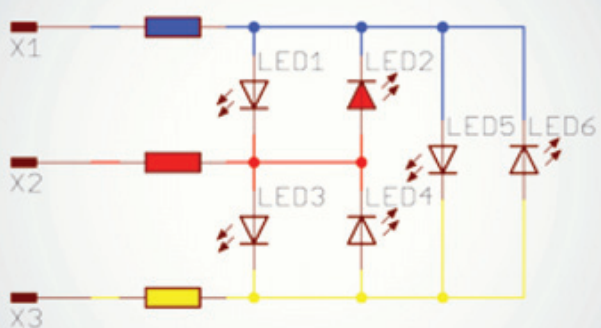
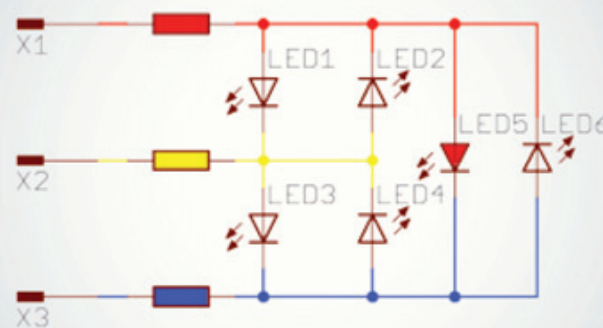


Fig. 4





Lo smartphone si trasforma in oltre 40 shield

È nato ed è stato proposto su Kickstarter **1Sheeld**, uno shield polivalente per Arduino che interfacciato con uno smartphone Apple o Android permette di emulare il funzionamento di tantissimo hardware; equipaggiato con un modulo Bluetooth Low Energy, può utilizzare i sensori e in generale l'hardware dello smartphone per emulare il funzionamento di tantissimi shield. L'hardware è composto da accelerometro e giroscopio, altoparlante e microfono, vibrecall, fotocamere (con cui realizzare il riconoscimento d'immagine e di colore), RTC, memoria (per realizzare dei Data Logger), pulsanti e touch-screen (utilizzabile come sensore di prossimità e dispositivo di input) display, NFC, ricevitore GPS, eventuale magnetometro ed altro per realizzare ad esempio applicazioni Text To Speech e di riconoscimento vocale, riproduttori musicali ecc. **1Sheeld** consente inoltre di interagire con le e-mail, Facebook, Foursquare. Lo shield contiene un modulo Bluetooth HM-10 e supporta smartphone con sistema operativo sia iOS (dal 9) che Android (dal 4.3). Lo shield si può acquistare dal sito www.futurashop.it.

e imponiamo per X2 il livello logico BASSO. Quanto a X3, lo configuriamo come INPUT:

```
pinMode(X1, OUTPUT);
pinMode(X2, OUTPUT);
pinMode(X3, INPUT); //Three-State
digitalWrite(X1, HIGH);
digitalWrite(X2, LOW);
```

In questo modo il diodo luminoso LED1 si accende, mentre gli altri LED rimangono spenti. Da notare che il LED2, che è collegato agli stessi pin del LED1, non si accende perché connesso in anti-parallelo a LED1 (**Fig. 2**), quindi può accendersi solamente

quando la polarità si inverte (in questo caso si spegne LED1). Ora, per spegnere il LED1, ci basta impostare tutti i pin in Three-State e quindi configurarli come INPUT:

```
pinMode(X1, INPUT);
pinMode(X2, INPUT);
pinMode(X3, INPUT);
```

Se volessimo accendere il LED2, per esempio, ci basta configurare i relativi pin connessi all'anodo e al catodo del LED, quindi X2 e X1, e configurarli come OUTPUT, imponendo per X2 la condizione logica HIGH e per X1 quella

LOW. Il pin X3 lo lasciamo in condizione three-state:

```
pinMode(X1, OUTPUT);
pinMode(X2, OUTPUT);
pinMode(X3, INPUT); //Three State
digitalWrite(X1, LOW);
digitalWrite(X2, HIGH);
```

Quindi la corrente scorre dal pin X2 e attraversa il LED2, che è polarizzato correttamente e va a chiudere il circuito nel pin X1. Invece il pin X3 non interferisce, in quanto è configurato come INPUT (**Fig. 3**).

Proviamo ora ad accendere il LED5: per fare ciò, dobbiamo impostare i pin X1 e X3 come OUTPUT, quindi X1 come HIGH e X2 come LOW, e configurare il pin X2 come Three-state, ossia INPUT (**Fig. 4**):

```
pinMode(X1, OUTPUT);
pinMode(X2, INPUT); //Three State
pinMode(X3, OUTPUT);
digitalWrite(X1, HIGH);
digitalWrite(X3, LOW);
```

Adesso il LED 5 è acceso, però a questo punto viene da domandarsi come mai LED 1 e LED3 non si accendono anche se la tensione ai loro capi è polarizzata correttamente; ebbene, il fatto è che questi due LED sono connessi in serie e quindi, per poterli accendere, il microcontrollore dovrebbe fornire una tensione doppia rispetto alla tensione di soglia del singolo LED. Dato che il microcontrollore fornisce sempre la stessa tensione di uscita, i due LED non si accendono e questo ci consente di mantenerli collegati come nell'esempio.

Adesso che abbiamo capito come accendere molti LED con pochi pin, dobbiamo però considerare un limite del Charlieplexing: con questa specifica tecnica di pilotaggio dei LED, è possibile accendere un solo LED del gruppo alla volta. Questo ci limita parecchio, infatti come facciamo

a “disegnare” nella matrice di LED di KeyChainino uno “smile”, per esempio?

Ebbene, ciò è fattibile perché in realtà quello che ci serve è visualizzare i LED accesi nello stesso istante, non tenere accesi i LED nello stesso istante: è una differenza sottile -un gioco di parole, se volete- ma si spiega perfettamente con quella che viene chiamata POV - acronimo di “persistence of vision”. Per capire la cosa sappiate che il nostro occhio non riesce a distinguere immagini che si succedono oltre una certa velocità, che la Scienza ha dimostrato essere 24 fotogrammi al secondo; questo sia per la persistenza della retina (cioè il tempo impiegato da coni e bastoncelli all'interno del nostro occhio ad adeguarsi alle variazioni di luce) sia per la velocità di elaborazione del cervello. Ne deriva che se accendiamo e spegniamo un LED molto più veloce di quanto l'occhio umano possa percepire, non riusciamo a percepire come vengono composte le immagini da parte della matrice a LED di KeyChainino e vediamo in realtà il disegno composto. Nel caso specifico, il

firmware fa accendere e spegnere i LED tanto rapidamente da comporre molte immagini al secondo e quindi i LED ci appaiono sempre accesi anche se in realtà pulsano velocemente.

Questa tecnica si può affiancare al Charlieplexing ed è ciò che abbiamo fatto: accendiamo un LED alla volta ma, eseguendo un refresh molto veloce -dell'ordine dei 60Hz- il nostro occhio vedrà tutti i LED accesi nello stesso momento.

Il vero problema è la gestione della matrice di LED senza doversi preoccupare di come sono collegati e senza pensare di dover aggiornare la matrice molto velocemente affinché il nostro cervello non veda i LED accendersi e spegnersi.

Per risolvere questo problema abbiamo ideato una funzione che, tramite l'utilizzo dell'interrupt del timer, aggiorna lo stato dei LED in “background”.

Quello che dovremo fare nello sketch sarà solamente cambiare il valore del LED che vogliamo

accendere, nel vettore multidimensionale “matrixState”. Se ad esempio volessimo accendere il LED nella posizione [Y][X] della matrice, ci basterà scrivere:

`matrixState[Y][X] = 1;`

Per spegnerlo, basterà settare tale valore a 0. In questo modo possiamo accendere e spegnere contemporaneamente tutti i LED della matrice 6x5 senza doverci preoccupare di nulla. Sarà la funzione associata all'interrupt del timer 1 ad andare a verificare quale LED accendere e come configurare i pin in base alla connessione del LED stesso.

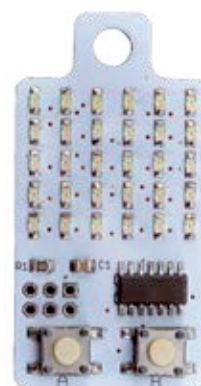
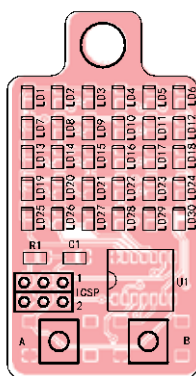
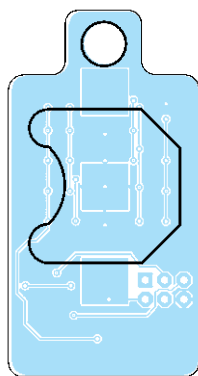


Lato posteriore di KeyChainino: ospita il portapila. La CR2032 va rimossa durante la programmazione, che va effettuata utilizzando il connettore ICSP che in questa foto non è montato (va nelle sei piazzole in basso).

[piano di MONTAGGIO]

Elenco Componenti:

R1: 10 kohm (0805)
C1: 100 nF ceramico (0805)
LD1÷LD30: LED rosso (0805)
U1: ATTINY84A-SSU
A: Microswitch SMD
B: Microswitch SMD
BAT1: Porta Batteria Bottone



Guarda Key Chainino in funzione su YouTube!



Il portachiavi tecnologico basato su Arduino è capace di tanti effetti e vari giochi.

Se vuoi vederlo in funzione collegati alla pagina https://youtu.be/NB_cfv-GEUg. Dal nostro canale YouTube <https://www.youtube.com/user/ElettronicaIN/videos> potrai anche vedere i video di moltissimi altri nostri progetti proposti in passato.



Un altro vantaggio del Charlieplexing è il ridotto consumo energetico che consente; è per questo che lo abbiamo voluto in KeyChainino, che è alimentato da una batteria CR2032, la quale ha una capacità di circa 225mAh. Il consumo di elettricità è un fattore determinante perché se anche riuscissimo ad accendere tutti i 30 LED contemporaneamente, considerando un assorbimento di 10 mA per LED, avremmo un prelievo di corrente dall'alimentazione pari a $30 \times 10 \text{ mA} = 300 \text{ mA}$.

Ammesso e non concesso che la batteria possa avere uno spunto tale da alimentare i LED contemporaneamente, essa si esaurirebbe in appena $225/300 = 0,75$ ore (45 minuti).

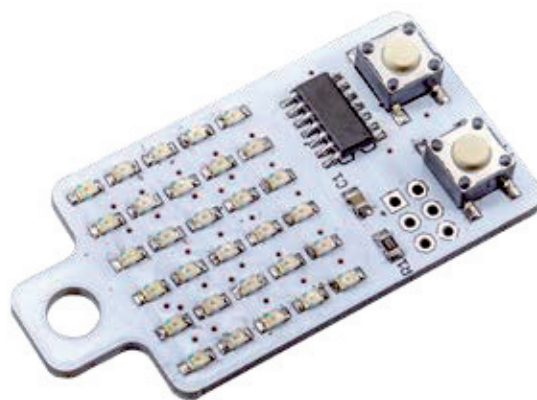
Con il Charlieplexing, invece, proprio perché viene alimentato soltanto un LED per volta, il consumo si riduce solamente a 10 mA per tutti i 30 LED accesi,

il che si traduce in un'autonomia -fornita dalla pila a bottone CR2032- di ben $225/10 = 22,5$ ore: 30 volte maggiore rispetto al caso precedente!

Naturalmente i LED sono accesi soltanto per 1/30 del tempo, quindi è come se pilotassimo i LED con la tecnica PWM con un duty-cycle del 3,33% che comporta una diminuzione dell'intensità luminosa del LED.

Un altro aspetto da considerare del Charlieplexing è l'utilizzo della resistenza di limitazione (è posta in serie...) della corrente nei LED. Normalmente tale resistenza viene calcolata dividendo la differenza tra la tensione fornita al bipolo resistenza-diodo e la caduta di tensione diretta del diodo stesso, per la corrente diretta nel LED. Nel nostro caso, però, dato che ogni LED è connesso a due pin, la resistenza calcolata, deve essere divisa per due e collegata in serie al rispettivo piedino del microcontrollore, anziché in serie a ciascun LED. In questo modo, ciascun LED è sempre connesso in serie tra due resistenze la cui somma è quella necessaria al funzionamento ottimale del LED stesso. Questo vale in linea generale per il pilotaggio in Charlieplexing, tuttavia in certe condizioni limite è possibile rinunciare alla resistenza; è il nostro caso, perché in KeyChainino si è preferito omettere la resistenza in quanto i LED utilizzati nel circuito hanno una tensione diretta molto vicina a quella della batteria, quindi la caduta di tensione sui MOSFET che costituiscono lo stadio di uscita di ciascun I/O del microcontrollore può compensare.

Poi c'è da dire che, data l'elevata velocità di refresh, ciascun LED viene alimentato solamente per qualche istante e questo permet-



te di alimentarlo ad una tensione istantanea di poco superiore senza danneggiare né esso né il microcontrollore.

In ogni caso, se la tensione di alimentazione è più alta di quella di alimentazione dei LED, conviene inserire la resistenza. KeyChainino è quindi un ottimo strumento per imparare la sorprendente tecnica del Charlieplexing, scoprire come si devono utilizzare gli interrupt e imparare a gestire il risparmio energetico; il tutto in un gadget unico e vecchio stile, a portata di chiavi! Creare giochi per KeyChainino è molto semplice: basta partire dallo sketch base scaricabile da GitHub all'indirizzo web https://github.com/alessandromatera/KeyChainino/blob/master/Sketch/KeyChainino_basic/KeyChainino_basic.ino. Sempre per imparare a pro-



per il MATERIALE

Il portachiavi programmabile KeyChainino (cod. KEYCHAININO) viene venduto montato ed è disponibile presso Futura Elettronica al prezzo Euro 13,00. Il prezzo si intende 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>

grammare KeyChainino potete seguire i tutorial che trovate sul sito web del progetto, realizzato dall'autore, alla pagina web <http://www.keychainino.com/tutorial-how-to-make-the-walled-game-step-by-step-part-1/>. Oppure potete partire dai vari giochi sviluppati dall'autore e da altri utenti di KeyChainino disponibili su www.keychainino.com/keychainino-sketches/ e modificare i rispettivi sketch a piacimento. Sul sito web del progetto, www.keychainino.com, trovate altro materiale in continuo aggiornamento.

REALIZZAZIONE PRATICA

Passiamo adesso dalla teoria alla pratica e vediamo come realizzare questo gadget: per contenere le dimensioni del circuito abbiamo fatto ricorso esclusivamente a componenti SMD e a uno stampato a doppia faccia a fori metallizzati o interconnessioni realizzate con sottili spezzoni di filo in rame nudo; lo stampato si ottiene per fotoincisione ricavando le pellicole dalle tracce lato rame che trovate sul nostro sito web www.elettronica.in.it, insieme agli altri file del progetto.

La costruzione richiede quindi l'attrezzatura del caso (un saldatore da 20 W a punta finissima, del filo di lega saldante da 0,5 mm di diametro, della pasta fluxante di media densità, una lente d'ingrandimento e una pinzetta per posizionare i chip più piccoli), oltre che un po' di pazienza e mano ferma.

Il montaggio di KeyChainino va iniziato posizionando per primo il microcontrollore, ben centrato nelle proprie piazzole, e stagnandone due terminali sugli angoli per fermarlo; poi procedete a stagnare i terminali uno per lato, magari dopo averli spalmati col fluxante per evitare che la lega saldante si attacchi tra due piedini unendoli. In questa operazione aiutatevi con la lente d'ingrandimento, perché il passo dei piedini è molto ridotto ed è facile realizzare cortocircuiti. Sistemato il microcontrollore, disponete sullo stampato i componenti più piccoli come resistenze e condensatori, quindi i LED, il quarzo, l'elettrolitico e i pulsanti. Completate il montaggio con il portapila e il connettore ICSP, che è un doppio strip maschio a passo 2,54 mm da 3 poli, da rimuovere dopo la programmazione in-circuit del microcontrollore; infatti le punte sono abbastanza alte e renderebbero il dispositivo piuttosto spesso, mentre per la sua natura di portachiavi è opportuno sia il più snello possibile.

Completato e programmato, KeyChainino va inserito in un contenitore in plastica (trasparente dal lato dei LED) che potete scegliere in base al vostro gusto; potete anche valutare la possibilità di coprire il lato dei LED con una resina trasparente (qualche mano di vernice poliuretanica, per esempio) e il retro con un coperchio, sempre in materiale isolante.

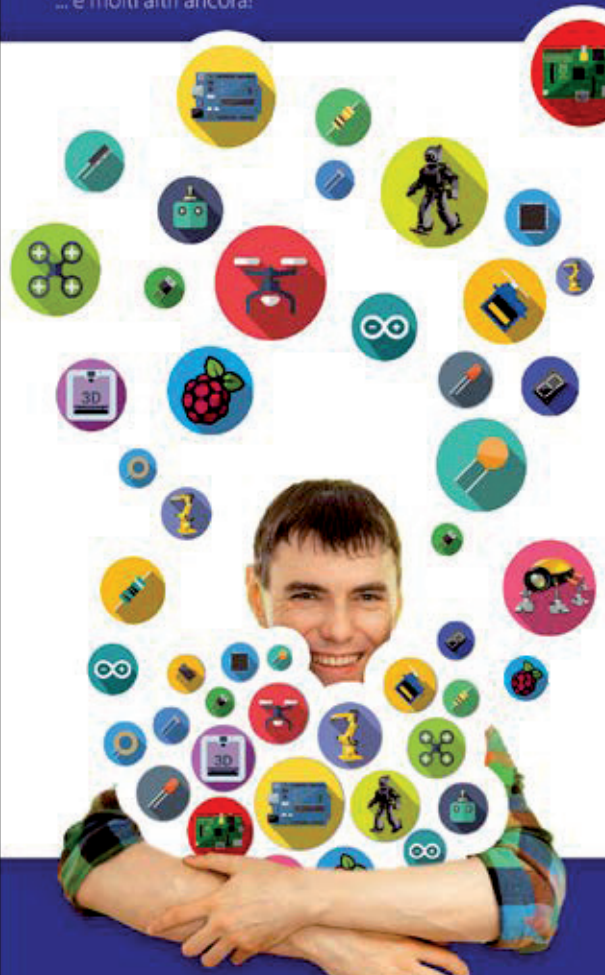
CORSI FUTURA ACADEMY



Puoi scegliere tra numerosi argomenti:

- Stampa 3D
- Raspberry Pi
- Droni
- Arduino
- Robotica
- Eagle
- Elettronica
- Fotovoltaico

... e molti altri ancora!



NON ASPETTARE.
Se ti iscrivi in anticipo avrai diritto ad uno sconto!

INFORMAZIONI DETTAGLIATE E ISCRIZIONI SU
www.futura.academy

Elettronica In - la più prestigiosa rivista di progettazione elettronica



**TUTTI i MESI
in EDICOLA!**

- Progetti pratici
- Articoli didattici
- Corsi hardware e software
- News dal mondo dell'elettronica
- ...e molto altro ancora.

ABBONATI! SOLO PER TE ALTRI VANTAGGI:

- + La DISCOUNT CARD che ti consentirà di usufruire di uno sconto del 10% su tutti i prodotti acquistati direttamente e on-line da Futura Elettronica.
- + Un volume a scelta della collana "L'Elettronica per tutti".
- + La possibilità di leggere la versione digitale della rivista direttamente da PC.

TI SEI PERSO DEGLI ARTICOLI INTERESSANTI? CREA LA TUA RACCOLTA ARRETRATI SU SUPPORTO USB

Crea con pochi clic una raccolta digitale di arretrati personalizzata. Riceverai, direttamente a casa tua, una card contenente una pen drive USB da 2GB nella quale troverai i pdf dei numeri di Elettronica In che hai scelto.

Le raccolte sono disponibili in tre differenti formati: da 10, da 25 e da 50 arretrati.



PER ABBONARTI COLLEGATI AL SITO

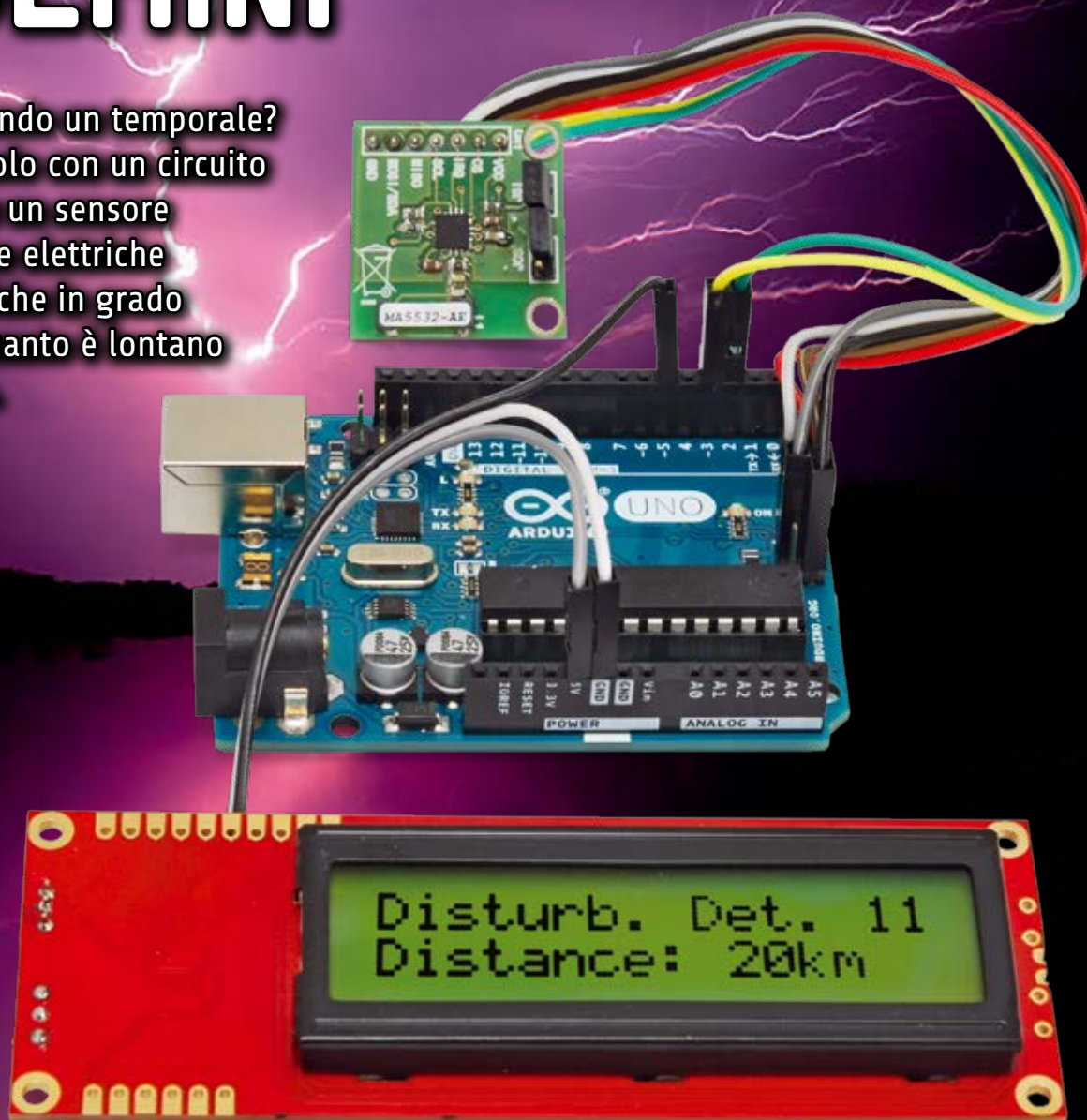
www.elettronica.in.it

**Futura
Group**
Edizioni

FUTURA GROUP srl
Via Adige 11
21013 Gallarate (VA)
TEL.: 0331 752668
Fax: 0331 792287

SENSORE DI FULMINI

Sta arrivando un temporale? Scopriamolo con un circuito basato su un sensore di scariche elettriche atmosferiche in grado di dirci quanto è lontano e intenso.



di BORIS LANDONI

I centri meteorologici effettuano il monitoraggio, oltre che dei parametri atmosferici allo scopo di stimare la probabilità di precipitazioni e la loro consistenza, anche dei temporali e dei fenomeni elettrostatici loro correlati, come i fulmini. In alcune nazioni esposte a consistenti fenomeni elettrici

atmosferici e ai rischi che comportano, tale attività assume particolare rilievo. Senza scomodare la cinematografia, che non ci ha fatto mancare lungometraggi del genere catastrofico con fulmini "assassini", possiamo comprendere come le saette che vediamo solcare il cielo o abbattersi

CARATTERISTICHE TECNICHE

- **Tensione di alimentazione:** 9V
- **Corrente assorbita:** 300 mA
- **Distanza di rilevamento:** 40 km
- **Settaggio per l'utilizzo sia all'interno che all'esterno**
- **Visualizzazione su display di numero e distanza dei fulmini**

possano essere pericolose, per l'energia notevole che possono cedere precipitando dal cielo (dove si formano) a terra, su edifici, alberi e purtroppo anche sull'uomo. Una pericolosità che diviene ancora maggiore in quelle aree dove possono verificarsi fulmini quasi a ciel sereno, ovvero in maniera inaspettata. Senza contare i danni che i fulmini caduti vicino alle linee di distribuzione elettrica e di telecomunicazioni causano, sotto forma di sovratensioni indotte, agli apparati elettrici e soprattutto elettronici.

L'attività di analisi dei fenomeni elettrici atmosferici viene svolta con varie metodiche (la più antica delle quali fu l'aquilone di Benjamin Franklin) basate sulla valutazione preventiva degli accumuli di elettricità statica nell'atmosfera e sull'intercettazione delle scariche elettriche, con apparati complessi e costosi, tuttavia si può condurre anche in maniera amatoriale, a casa propria, grazie a dispositivi come il sensore basato sull'integrato AS3935 della Austriamicrosystems, del quale in queste pagine vi facciamo vedere un'applicazione. Il componente utilizzato riesce a captare i disturbi radioelettrici causati dallo scoccare di archi elettrici (e i fulmini sono tali) e, tramite un'opportuna elaborazione, fornisce un segnale che corrisponde al rilevamento di un fulmine e della sua intensità.

UN PO' DI TEORIA

I fulmini sono scariche elettriche atmosferiche aria-aria o aria-terra che si innescano a causa dell'ac-

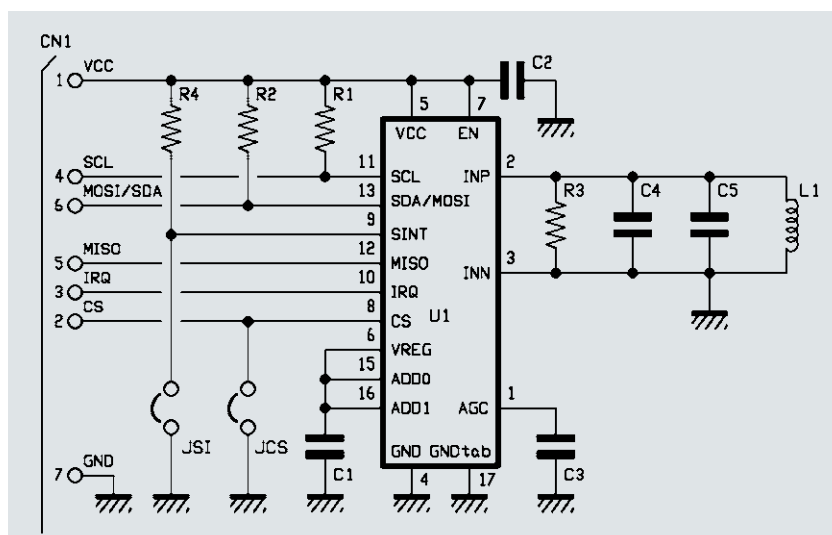
cumulo di una consistente carica elettrica, ovvero quando si crea tra due punti dell'atmosfera o tra l'aria e la terra (ma anche il mare o i corsi d'acqua) un forte scompenso di carica elettrica; nel momento in cui la differenza di potenziale dovuta allo scompenso di carica è tale da superare la rigidità dielettrica del mezzo interposto (l'aria, di solito, che sia più o meno umida...) scocca l'arco elettrico, vale a dire la saetta, che comporta un flusso di corrente attraverso un mezzo per sua natura isolante.

La luce che ne deriva è causata dalla ionizzazione dei gas componenti l'aria compresa tra i punti tra i quali scocca l'arco, ovvero dal ritorno nei legami atomici degli elettroni asportati forzatamente dal fortissimo campo elettrico durante la scarica; l'energia restituita dall'elettrone prima del rientro nel legame produce un fotone, ossia una particella di luce, che nei fulmini

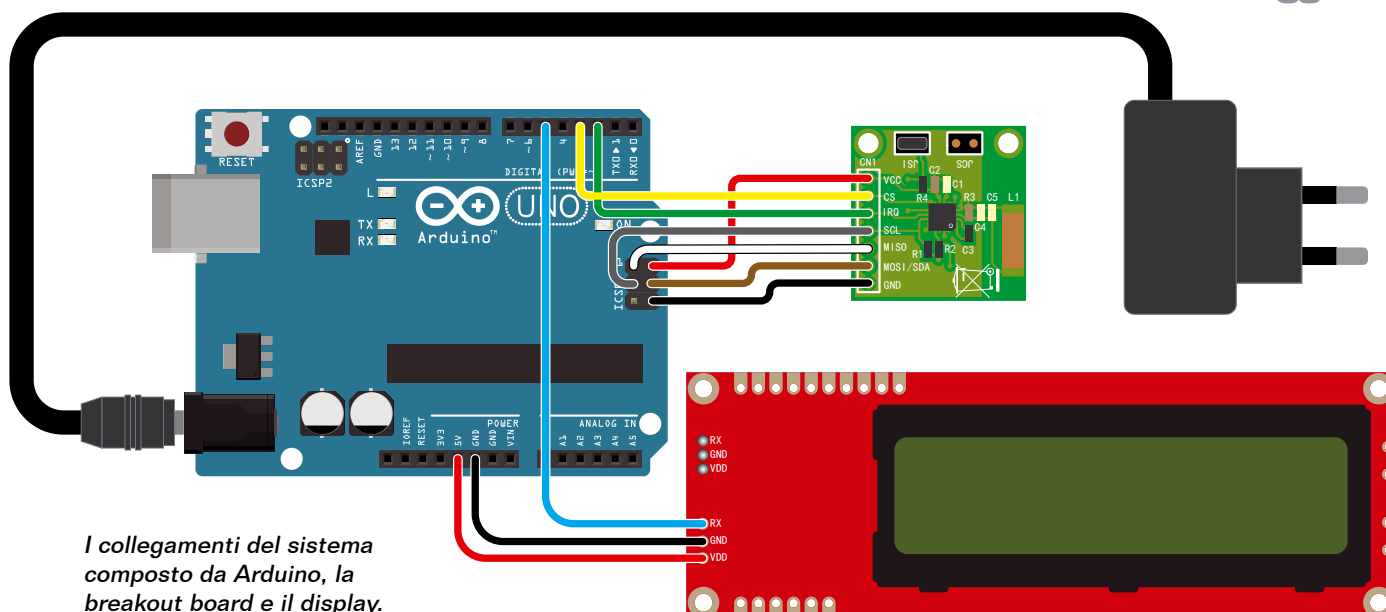
è in buona parte collocata nello spettro del visibile.

Tale attività elettrica è percepibile a distanza come fosse un segnale radioelettrico trasmesso da un'emittente: lo dimostrò l'esperimento di Hertz, il quale fece scoccare delle scintille e rilevò a distanza con un dipolo la debole tensione che ne risultava. Infatti possiamo immaginare lo scoccare di un fulmine come fosse un'oscillazione che stimola elettricamente l'aria e fa propagare una carica elettrica a una distanza che dipende dalla composizione dell'aria e dalle sue condizioni, nonché dalle prestazioni dell'antenna ricevente e del dispositivo utilizzato per la ricezione.

Utilizzando un sensibile ricevitore RF ed elaborando il segnale ricevuto mediante un algoritmo proprietario, l'integrato AS3935 determina se l'attività elettrica captata è causata da un fulmine, ovvero stima la probabile distanza sulla base dell'intensità del segnale; per distanza si intende verosimilmente quella dall'inizio del temporale o tempesta nell'ambito del quale scoccano i fulmini e il chip può determinarla con sufficiente approssimazione tra un massimo di 40 km e



[schema **ELETTRICO**]



I collegamenti del sistema composto da Arduino, la breakout board e il display.

un minimo di 1km. L'algoritmo serve a discernere, con una certa approssimazione, tra fulmini (scariche atmosferiche) e scariche elettriche prodotte ad esempio da spinterogeni e candele di motori a benzina, motori elettrici a spazzole, forni a microonde.

IL SENSORE DI FULMINI

Per rendere più facilmente utilizzabile il sensore di scariche atmosferiche abbiamo realizzato una breakout board contenente l'integrato AS3935; ne trovate lo schema elettrico in queste pagine. Il cuore è chiaramente l'integrato, del quale portiamo all'esterno l'alimentazione, il bus per la comunicazione con circuiti esterni a microcontrollore, oltre che i principali segnali di controllo. Tutte le linee da collegare all'esterno o gestire dall'esterno sono riportate a una fila di piazzole a passo 2,54 mm poste su un lato, alle quali è possibile applicare un pin-strip. La connessione dati con l'esterno avviene serialmente tramite I²C-Bus o SPI; in quest'ultimo caso impieghiamo le linee:

- MISO (Master Input Slave Output) che è l'uscita della

breakout board con la quale vengono inviati ad Arduino i dati relativi ai fulmini rilevati;

- MOSI (Master Output Slave Input) che è l'uscita dati di Arduino con la quale vengono comunicati alla breakout board messaggi di impostazione o di conferma;
- CS (Chip Select) ovvero la linea di uscita di Arduino che abilita la breakout board all'interazione sul bus SPI; il CS serve perché sull'SPI possono affacciarsi vari dispositivi e per evitare collisioni dei dati occorre che ad ogni sessione di comunicazione ne venga abilitato uno solo per volta.
- SCL (clock) che corrisponde al segnale con cui viene sincronizzata la comunicazione sul bus.

I predetti segnali vengono riportati al CN1; per la linea di clock SCL e per il MOSI sono state inserite le resistenze di pull-up. Notate che il piedino 8 dell'AS3935, vale a dire il CS, in assenza di diversa impostazione viene mantenuto a livello alto dal proprio resistore di pull-up

interno, ma può essere forzato a zero logico dal jumper JCS laddove non si preveda la gestione dall'esterno.

Quanto alla modalità di comunicazione seriale, nella breakout board si può selezionare tramite un secondo jumper, siglato SI, che interviene sul piedino 9 (Select Interface); l'integrato AS3935 funziona:

- in modalità SPI se il piedino 9 è posto a zero logico;
- in modalità I²C se il Select Interface è posto a VDD.

Nella nostra applicazione abbiamo optato per la prima, perciò vedete che il jumper è stato chiuso.

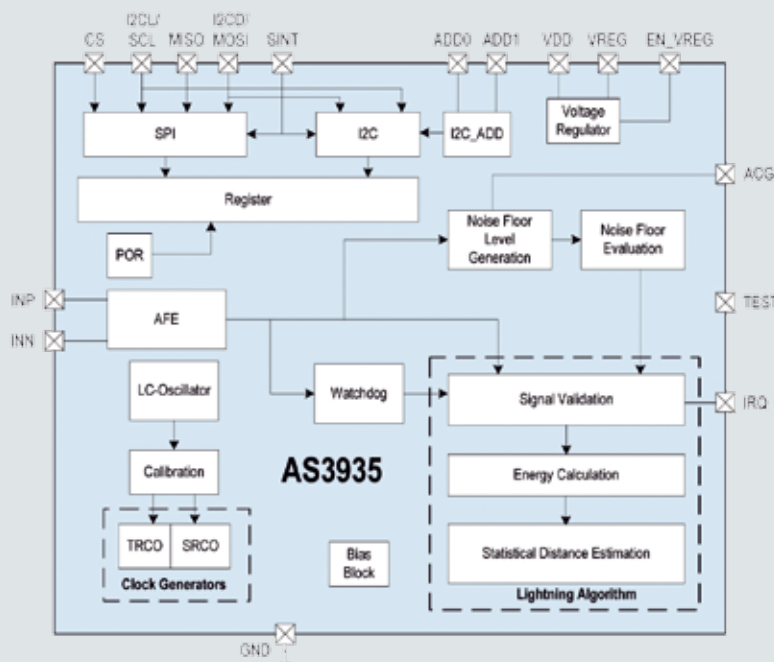
Abbiamo poi il piedino IRQ (10), riportato al contatto 3 del connettore CN1, che comunica al microcontrollore quando IRQ è un'uscita che l'AS3935 porta a livello logico alto quando viene intercettato un fulmine e scritto in uno dei registri interni (REG0x03[3:0]) il valore corrispondente.

Procediamo con l'analisi dello schema elettrico passando allo stadio d'ingresso, che è quello attraverso cui captiamo il di-

L'integrato AS3935

Il nostro circuito rileva i fulmini grazie a un integrato della Austria-microsystems (<http://ams.com>) che si può considerare un radiorecettore a larga banda con front-end analogico (il blocco AFE nel disegno in questo riquadro...) accordato da un circuito antirisonante esterno da applicare ai pin INP - INN e una sorta di demodulatore che estrae il segnale. Questo, comprendente il rumore di fondo, viene poi instradato verso il blocco analizzatore dell'attività elettrica atmosferica, entrando in un blocco di validazione del segnale (Signal Validation) che, tenendo conto della risultanza del blocco analizzatore del livello di rumore di fondo, identifica con sufficiente approssimazione il segnale dovuto ai fulmini. Appurato che si tratta dell'attività elettrica di un fulmine, determina l'energia e da essa la distanza da cui proviene, scrivendo nel registro digitale (Register) un valore numerico corrispondente all'ampiezza e alla validità del rilevamento. Ogni rilevamento viene segnalato tramite un impulso a zero logico sul pin IRQ e il microprocessore con cui il chip va interfacciato può acquisire i dati corrispondenti attraverso il canale dati seriale (impostabile tramite il SINT tra I²C-Bus e SPI) di

cui l'integrato dispone. Nel caso si utilizzi la modalità I²C-Bus, l'address dell'integrato si imposta con i pin ADD0 (bit meno significativo) e ADD1 (bit più significativo). Nel registro, i valori possono spaziare tra 111111, corrispondente a nessun rilevamento, a 000001 che è il massimo e significa praticamente avere un fulmine nel cielo soprastante. Il ricevitore dell'AS3935 è del tipo a banda stretta, con una frequenza di centro-banda di 500 kHz e una larghezza di banda che, con i componenti da noi previsti nella breakout board, è circa 33 kHz. Il guadagno dell'AFE può essere impostato con un apposito comando seriale ottimizzandolo per il funzionamento indoor (circuito utilizzato al chiuso) o outdoor (circuito posto all'aperto). Nel nostro caso il firmware di Arduino imposta la seconda modalità. Un dettaglio interessante dell'integrato è che può rilevare sia i fulmini che scoccano dal cielo a terra, sia quelli che rimangono in cielo, entro una distanza di 40 km. Maggiori dettagli si trovano nel datasheet del componente, reperibile in Internet all'indirizzo <http://ams.com/eng/Products/Wireless-Connectivity/Wireless-Sensor-Connectivity/AS3935>.



sturbo radioelettrico corrispondente al fulmine; esso fa capo ai piedini 2 e 3, che sono l'input del radiorecettore presente nell'integrato, il quale richiede all'ingresso un circuito antirisonante allo scopo di sintonizzare una certa banda ed escludere disturbi a bassa frequenza e di natura estranea allo scopo.

Il circuito di sintonia è composto da una bobina (induttanza AF) acquistabile già pronta e da due condensatori in parallelo, che assicurano il valore di capacità richiesto per determinare la frequenza di centro banda voluta. La bobina fa anche da antenna ricevente ed assicura un'ottima sensibilità. Il resistore R3, posto in parallelo al circuito antirisonante, peggiora il fattore di merito dello stesso ma limita superiormente l'ampiezza del segnale introdotto nello stadio di sintonia; peggiorando il fattore di merito si allarga la banda sintonizzabile, il che ci va benissimo perché dobbiamo rilevare scariche elettriche che non hanno una precisa frequenza, ma cadono entro un ampio spettro.

La frequenza di centro banda dello stadio di sintonia è determinata dalla formula:

$$f = 1/6,28 \times \sqrt{LC}$$

dove L è l'induttanza della bobina e C la somma di C4 e C5. Il fattore di merito Q è invece dato dal rapporto:

$$Q = f/Bw$$

dove Bw è la larghezza di banda del circuito antirisonante.

Il radiorecettore è del tipo ad amplificazione diretta e dispone di AGC, per regolare il guadagno in funzione del livello atteso all'uscita; il condensatore C3

```

#include <SPI.h>
#include <AS3935.h>
#include <SoftwareSerial.h>
#define LCD_pin 5 // LCD data signal

const int NoDetect=30;
int counter;
int NumDisturber=0;

SoftwareSerial LCD = SoftwareSerial(0, LCD_pin);

void printAS3935Registers();

byte SPITransfer(byte sendByte);

void AS3935Irq();

AS3935 AS3935(SPITransfer,3,2); //change to AS3935(SPITransfer,9,3) if using slot #2

void setup()
{
  serLCDInit();
  backlightOn();
  clearLCD();
  lcdPosition(0,3);
  LCD.print("Lightning");
  lcdPosition(1,5);
  LCD.print("Sensor");
  delay(1500);
  clearLCD();

  Serial.begin(9600);
  SPI.begin();
  SPI.setDataMode(SPI_MODE1);
  SPI.setClockDivider(SPI_CLOCK_DIV16);
  SPI.setBitOrder(MSBFIRST);
  AS3935.reset();
  delay(10);
  AS3935.setOutdoors();
  AS3935.registerWrite(AS3935_NF_LEV,2); //write 2 in the Noise Level register
  //AS3935.registerWrite(AS3935_SREJ,0); //write 2 in the Noise Level register and run calibration
  // if lightning detector can not tune tank circuit to required tolerance, calibration function will return false
  if(!AS3935.calibrate())
  {
    Serial.println("Tuning out of range, check your wiring, your sensor and make sure physics laws have not changed!");
    // now we print the value in the calibration register TUN_CAP it is in the range 0 - 15
    tunecap=AS3935.registerRead(AS3935_TUN_CAP); //Internal calibration
    Serial.print("Tuning cap register is ");
    Serial.println(tunecap);
  }

  AS3935.enableDisturbers();
  printAS3935Registers();
  // AS3935IrqTriggered = 0;
}

```

(Continua)

determina il tempo di intervento dell'AGC.

Il segnale fornito viene quindi analizzato dall'apposito blocco interno al chip, che lo confronta con la media del rumore di fondo e tramite l'apposito algoritmo determina cosa può essere un fulmine e cosa va scartato, quindi il risultato produce una stringa di dati che viene depositata nell'apposito registro e, laddove il valore corrisponda a quello di un fulmine, genera un

impulso sull'IRQ (piedino 3 del connettore CN1).

Concludiamo l'analisi dello schema elettrico della scheda sensore con i piedini ADD0 e ADD1, che servirebbero per impostare l'indirizzo dell'integrato laddove per la comunicazione venisse utilizzato il bus I²C e che in questo caso, essendo inutilizzati, sono portati a livello logico alto collegandoli all'uscita VREG (piedino 6) del regolatore di tensione interno all'integrato;

la tensione fornita da tale pin viene filtrata dal ripple mediante il condensatore C1.

Ciò detto, passiamo a vedere come la breakout board è utilizzata nel circuito.

L'APPLICAZIONE

Il progetto che vi proponiamo di realizzare consta sostanzialmente di tre elementi: il sensore di fulmini, disponibile su breakout board, una scheda Arduino Uno o Fishino Uno, il cui compito è

Listato 1 - (segue)

```
void loop()
{
    delay(1000);
    Serial.println("Waiting...");
    if (counter==0)
    {
        NumDisturber=0;
        counter=NoDetect;
        clearLCD();
        lcdPosition(0,1);
        LCD.print("No lightning");
        lcdPosition(1,4);
        LCD.print("detected");
    }
    else
    {
        counter=counter - 1;
    }
}

void printAS3935Registers()
{
    int noiseFloor = AS3935.getNoiseFloor();
    int spikeRejection = AS3935.getSpikeRejection();
    int watchdogThreshold = AS3935.getWatchdogThreshold();
    Serial.print("Noise floor is: ");
    Serial.println(noiseFloor,DEC);
    Serial.print("Spike rejection is: ");
    Serial.println(spikeRejection,DEC);
    Serial.print("Watchdog threshold is: ");
    Serial.println(watchdogThreshold,DEC);
}

// this is implementation of SPI transfer that gets passed to AS3935
{
    return SPI.transfer(sendByte);
}

void AS3935Irq()
{
    int irqSource = AS3935.interruptSource();
    // returned value is bitmap field, bit 0 - noise level too high, bit 2 - disturber detected, and finally bit 3 - lightning!
    if (irqSource & 0b0001)
        Serial.println("Noise level too high, try adjusting noise floor");
    if (irqSource & 0b0100)
    {
        NumDisturber+=1;
        Serial.println("Disturber detected");
        clearLCD();
        lcdPosition(0,0);
        LCD.print("Disturb. Det: ");
        LCD.print(NumDisturber,DEC);
        counter=NoDetect;
    }
    if (irqSource & 0b1000)
    {
        // need to find how far that lightning stroke, function returns approximate distance in kilometers,
        // where value 1 represents storm in detector's near vicinity, and 63 - very distant, out of range stroke
        // everything in between is just distance in kilometers
        int strokeDistance = AS3935.lightningDistanceKm();
        if (strokeDistance == 1)
        {
            Serial.println("Storm overhead, watch out!");
            lcdPosition(1,1);
            Serial.println("Storm overhead");
            lcdPosition(1,3);
            Serial.println("WATCH OUT!");
            counter=NoDetect;
        }

        if (strokeDistance == 63)
        {
            Serial.println("Out of range lightning detected.");
        }
    }
}
```

(Continua)

Listato 1 - (segue)

```
    lcdPosition(0,2);
    Serial.println("Out of range");
    lcdPosition(1,0);
    Serial.println("lightning detect");
    counter=NoDetect;
}

if (strokeDistance < 63 && strokeDistance > 1)
{
    Serial.print("Lightning detected ");
    Serial.print(strokeDistance,DEC);
    Serial.println(" kilometers away.");
    lcdPosition(1,0);
    LCD.print("Distance: ");
    LCD.print(strokeDistance,DEC);
    LCD.print("km");
    counter=NoDetect;
}
}
```

analizzare il segnale fornito dal sensore ed elaborarlo per poi mostrare sul display LCD seriale (che è il terzo elemento circuitale) le informazioni che ne desume, vale a dire il numero di scariche elettriche rilevate che si possono ritenere "figlie" di un temporale e la distanza stimata.

Il tutto dev'essere collegato come mostrato nel disegno di cablaggio visibile in queste pagine, dove la breakout board sensore si interfaccia ad Arduino tramite il connettore dell'ICSP, grazie al fatto che quest'ultimo veicola il bus SPI completo, ossia a 4 linee. Per la gestione del display seriale utilizziamo una seriale software, emulata grazie alla libreria *SoftwareSerial.h*, il che vi consente di avere libera la seriale hardware per eventuali applicazioni che lo richiedano. La breakout board sensore è collegata, come detto, attraverso un bus SPI, grazie al quale riceve le impostazioni del caso e comunica i dati sul rilevamento dei fulmini; il chip a bordo provvede già da solo a misurare l'intensità e a rilevare l'andamento delle scariche elettriche captate dalla bobina montata sulla breakout board, quindi i dati che fornisce non richiedono una particolare elaborazione da parte di Arduino.

La connessione con la breakout board avviene sfruttando le linee del bus SPI riportate sul connettore ICSP; ci sono poi GND (la massa) e Vcc (collegato al +5V) con cui la scheda prende l'alimentazione da Arduino.

Il CS (attivo a livello logico zero) viene gestito dal digital I/O 3 di Arduino, impostato dallo sketch come uscita e l'IRQ della breakout board è invece interfacciato al digital I/O 2, inizializzato come input, dato che, nello specifico, IRQ avvisa Arduino o un eventuale microcontrollore interfacciato all'integrato o alla sua breakout board che può leggere il contenuto del registro. Quando IRQ torna a zero logico è prevista una pausa di almeno 2 ms prima della lettura del registro.

Per la gestione del sensore da Arduino allo scopo di leggere i dati rilevati, è stata utilizzata una libreria sviluppata da un appartenente alla community (Raivis Rengelis) e scaricabile da github all'indirizzo web <https://github.com/raivisr/AS3935-Arduino-Library>. La gestione è affidata a uno sketch il cui codice riportiamo nel **Listato 1**.

Quanto al display LCD seriale, è collegato ad Arduino tramite i contatti 5V e GND per l'alimentazione e il digital I/O 5,

impostato come uscita, che invia i dati serialmente al contatto RX dell'LCD.

Il progetto mostra come leggere i dati rilevati dalla breakout, in particolare viene rilevato un eventuale temporale mostrando l'indicazione su un display LCD a comunicazione seriale.

Normalmente viene mostrata l'indicazione del temporale rilevato con il numero di rilevamenti negli ultimi 60 secondi. Se entro 60 secondi non avvengono altri rilevamenti, allora viene mostrata l'indicazione a display "No lightning detected" (Fig. 1). Se nel registro dell'integrato è disponibile la distanza di rilevamento, allora viene mostrato anche questo dato nella seconda



I fulmini, questi “conosciuti”

Correva l'anno 1752 quando Benjamin Franklin, facendo volare un aquilone dotato di una punta, dimostrò al mondo che i fulmini erano fenomeni elettrici. All'altro capo del filo con cui teneva l'aquilone, Franklin raccolse sufficiente elettricità da produrre scariche tra due elettrodi, come si faceva in laboratorio con macchine in grado di generare alte tensioni. Oggi, grazie anche a Franklin, sappiamo che i fulmini sono scariche elettriche causate dall'accumulo di cariche elettriche di polarità opposta tra due zone dell'atmosfera o fra cielo e terra. I fulmini, come tutti i fenomeni causati dal campo elettrico, prediligono le punte, non a caso i parafulmini sono a punta. In generale la fuga o la ricezione di carica elettrica si verificano preferibilmente nelle zone sporgenti e

appuntite, ovvero assottigliate. Per questo anche gli alberi, specie gli abeti e i cipressi, attirano i fulmini a terra ed anche l'uomo, se si trova su un terreno pianeggiante umido (o in acqua) senza elementi appuntiti intorno, rischia di attirare le saette, con conseguenze tragiche. Trattandosi di elettricità, l'energia dei fulmini scaricata a terra predilige i percorsi a bassa resistenza, quindi strutture metalliche e piante (perché percorse dalla linfa). Quando un fulmine si scarica a terra può propagarsi per un tratto orizzontalmente, se il suolo è bagnato e crea un percorso a bassa resistenza dal punto di abbattimento: ciò causa pericolosissime differenze di potenziale anche molto elevate tra due punti, tali a volte da uccidere capi di bestiame.

to il pin-strip e i jumper, quindi per il montaggio occorre disporre di un saldatore da non oltre 20 W, a punta finissima, oltre che di filo di lega saldante del diametro di 0,5 mm, di una pinzetta per posizionare i componenti e del fluxante per facilitare l'adesione della lega saldante. Una lente d'ingrandimento aiuterà a posizionare i componenti più piccoli centrati nei rispettivi pad e identificare, a saldatura avvenuta, eventuali cortocircuiti provocati da eccessi di lega saldante fusa su terminali attigui. È auspicabile avere una certa manualità e un minimo di pratica con i componenti a montaggio superficiale. I componenti vanno disposti seguendo il piano di montaggio che trovate in queste pagine, dove potete vedere il giusto orientamento di quelli polarizzati; fate attenzione a non surriscaldare la bobina, onde evitare che si sciolga lo smalto e che venga pregiudicato l'isolamento. Tutti

riga del display, con un messaggio del tipo:

Disturb Det. x

dove al posto di “x” è indicato quante scariche sono state rilevate. Sotto, appare la scritta (Fig. 2):

Distance: n km

dove viene indicata, al posto di “n” la distanza stimata da dove è scoccato il fulmine. Se la distanza è inferiore al minimo, il display mostra nella prima riga *Storm overhead* e nella seconda *WATCH OUT!*

REALIZZAZIONE PRATICA

Bene, vediamo adesso come costruire il nostro rilevatore di fulmini: la prima operazione da compiere è la costruzione della breakout board (se non ve la sentite, sappiate che si trova già pronta presso Futura Elettronica, www.futurashop.it) partendo dal circuito stampato a doppia faccia del quale pubblichiamo sul

nostro sito www.elettronica.in.it, insieme agli altri file del progetto, le tracce lato rame. Incisa e forata la basetta dovete montare i componenti, che sono in SMD, eccet-



Fig. 1 - Condizione di riposo.



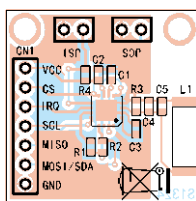
Fig. 2 - Rilevamento di 11 fulmini a una distanza stimata di 20 km.

Elenco Componenti:

R1 ÷ R4: 10 kohm (0603)
 C1: 1 μ F ceramico (0603)
 C2: 100 nF ceramico (0603)
 C3: 10 μ F ceramico (0603)
 C4: 680 pF ceramico (0603)
 C5: 270 pF ceramico (0603)
 L1: MA5532-AEB
 U1: AS3935-BQFT

Varie:

- Strip maschio 2 vie (2 pz.)
- Strip maschio 7 vie
- Jumper (2 pz.)
- Circuito stampato S1324 (26x26 mm)



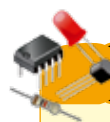
gli elementi vanno montati sul lato superiore. Il componente più critico è l'integrato AS3935, incapsulato in contenitore 16LD MLPQ (quadrato da 4x4 mm di lato) che a stretto rigore andrebbe montato con una stazione ad aria calda per il rework degli SMD, dopo aver stagnato e cosperso con il Flussante le rispettive piazzole; con un po' di attenzione e manualità è possibile saldarlo, una volta compiute le predette operazioni, mediante il classico saldatore, avendo cura di toccare i pad lateralmente (il risultato non sarà lo stesso ma funzionerà ugualmente).

Terminate le saldature e verificato che sia tutto a posto, chiudete il ponticello JSI e lasciate aperto JCS, giacché all'impostazione del Chip Select provvederà Arduino. Ora prendete in mano lo schema di cablaggio e, con dei cavetti jumper, collegate tra loro Arduino, il display e la breakout board secondo quanto indicato. Ricordate che l'alimentazione per l'insieme si fornisce in un unico punto, ossia sul connettore jack di Arduino Uno, mediante un alimentatore da 9 volt, capace di erogare 300 mA. Prima di dare alimentazione collegate alla presa USB di Arduino un cavo USB

A/B che dall'altro lato inserirete in una presa USB del vostro computer, nel quale avrete avuto cura di installare l'IDE Arduino (va bene qualsiasi versione tra quelle uscite negli ultimi due anni); aprite quindi il file contenente lo sketch, che dovrete aver copiato nella cartella dell'IDE insieme alla libreria, quindi effettuate il download in Arduino. Fatto ciò, staccate il PC e alimentate la vostra Arduino con l'alimentatore, verificando che il display si accenda e mostri i messaggi: inizialmente e in assenza di attività, apparirà il messaggio No lightning detected.

Se volete effettuare un test spicciolo del sistema, procuratevi un accendigas piezoelettrico (quelli a scintilla) e, mantenendolo a mezzo metro di distanza dalla breakout board premete più volte il pulsante per far scoccare qualche scintilla: il display dovrebbe mostrare il segnale Disturb. Det. con un numero che si aggiorna man mano che premete. Con questo avete collaudato, seppure in maniera empirica, il sistema e siete certi che funzionerà in campo. Per un utilizzo più pratico inserite il tutto in un contenitore in plastica, lavorato per far uscire il display e rendere

accessibile il jack di alimentazione di Arduino. La breakout board mettetela sotto al coperchio, fissandola con del silicone sigillante; evitate le viti, che potrebbero interferire, a meno di non utilizzare quelle in teflon. Non utilizzate contenitori metallici perché schermerebbero la bobina ricevente della breakout board, con le conseguenze immaginabili, ossia il circuito non potrà rilevare alcun fulmine. Ultima nota: lo sketch di Arduino prevede attualmente che per lo stadio RF sia impostato il guadagno corrispondente all'utilizzo all'esterno, come evidenziato nella riga di codice **AS3935.setOutdoors()**. Se desiderate la modalità indoor, perché utilizzate il circuito in casa, dovete cambiare tale riga in **AS3935.setIndoor()**. Bene, detto ciò possiamo concludere ricordandovi che il progetto, come tutti quelli open, può essere personalizzato da ognuno di voi al bisogno. ■



per il MATERIALE

Il materiale elencato di seguito è disponibile presso Futura Elettronica. La breakout con l'integrato AS3995 (cod. FT1324M) viene fornita montata e collaudata al prezzo di Euro 25,00, la board Arduino UNO REV3 (cod. ARDUINOUNO-REV3) è disponibile a Euro 24,50 e il display LCD seriale 16x2 nero/verde (cod. LCDSER16X2NV) è in vendita a Euro 34,00. Tutti i prezzi si intendono IVA compresa.

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

VUOI SVILUPPARE LE TUE APPLICAZIONI CON ARDUINO?

Da noi trovi tutto quello che ti serve!



L'ABC di ARDUINO

COD. ABCARDU

€ 9,90



Arduino e le tecniche di programmazione dei microcontrollori Atmel

COD. ATPROMA

€ 15,00



ARDUINO UNO Programmazione avanzata e librerie di sistema

COD. ARDUADVANCED

€ 22,00

Arduino Uno Rev3

Arduino Mega

Arduino Due



COD. ARDUINOUNOREV3

€ 24,50



COD. ARDUINOMEGAREV3

€ 43,00



ARDUINODUE

€ 44,00

Prezzi IVA inclusa

kit V5 con Arduino Uno REV3

Set contenente tutti i componenti necessari per realizzare gli esperimenti descritti nel libro "L'ABC di Arduino".



COD. ARDUKITV5

€ 59,00

Starter kit per robot con Arduino Uno

Set contenente le parti necessarie per realizzare il Robot descritto nel libro incluso.

€ 155,00



COD. ARDUKITV4

FUTURA ELETTRONICA®
www.futurashop.it

Futura Group srl • via Adige, 11 • 21013 Gallarate (VA)
Tel. 0331/799775 • Fax. 0331/792287

Caratteristiche tecniche di questi prodotti e acquisti on-line su www.futurashop.it