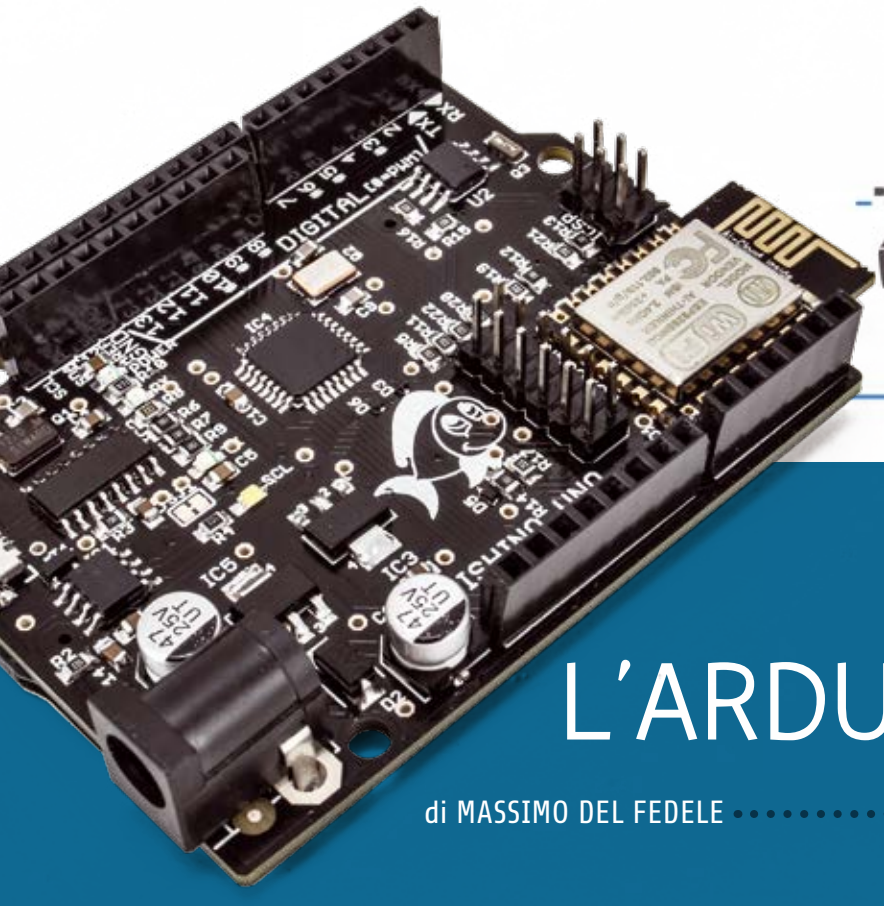
Nel firmamento Arduino brilla una nuova "scheda" simile alla UNO ma dotata di connettività WiFi e lettore per SD. Prima puntata.



FISHINO, L'ARDUINO DIVENTA WIRELESS

di MASSIMO DEL FEDELE

Il nome della scheda descritta in queste pagine è nato da un "pesce d'aprile" fatto dall'autore su un forum dove si presentava una fantomatica scheda denominata "**Fishino Zero**" dotata di caratteristiche mirabolanti, tra le quali un processore "Horata" di ultima generazione, tecnologia WiFi "Poisson", connessione avanzata WiFi "Fishnet" ed altre improbabili meraviglie. Il simbolo, piazzato tramite un programma di grafica sulla foto di una scheda esistente, era appunto il pesciolino che, finito lo scherzo, è diventato il logo della scheda che proponiamo in queste pagine.
Da scherzo, l'idea ha iniziato a prender forma fino alla realizzazione della board definitiva.
Il termine **UNO** è poi stato aggiunto sia per indicare la prima di una possibile serie di schede Arduino-compatibili, che per indicare la completa compatibilità con Arduino Uno, sia come connettività che come dimensioni.

UN ALTRO CLONE DI ARDUINO?

Si e no: con questa scheda abbiamo voluto realizzare un prodotto nuovo in grado di abbinare la semplicità d'uso e la sterminata quantità di librerie e shield di Arduino con la connettività Internet, una dotazione praticamente illimitata di memoria grazie alla scheda microSD e, ultimo ma non per importanza, un orologio interno con backup a batteria, il tutto ad una frazione del costo d'acquisto di una Arduino e degli shield relativi alle funzioni implementate, senza occupare prezioso spazio aggiuntivo; la scheda ha infatti un ingombro identico a quello dell'Arduino Uno, salvo la piccola sporgenza dell'antenna WiFi, di soli 7 mm.
L'integrazione delle periferiche descritte, a nostro avviso indispensabili nell'era dell'IoT, permette di realizzare tutta una serie di apparecchi sia controllabili via Internet che in grado di connettersi ad essa a richiesta e trasmettere dati rilevati in precedenza.

CARATTERISTICHE TECNICHE

- **Alimentazione: 12 Vcc o USB**
- **Completamente compatibile con Arduino Uno**
- **Scheda WiFi a bordo, con possibilità di funzionamento in modalità stazione, access point o entrambe contemporaneamente**
- **Interfaccia per schede di memoria MicroSD a bordo**
- **RTC (Real Time Clock) a bordo con batteria al litio di mantenimento**
- **Sezione di alimentazione a 3,3 V potenziata**
- **Connettore aggiuntivo sfalsato in modo da risolvere il problema dell'incompatibilità di Arduino con le schede millefori.**

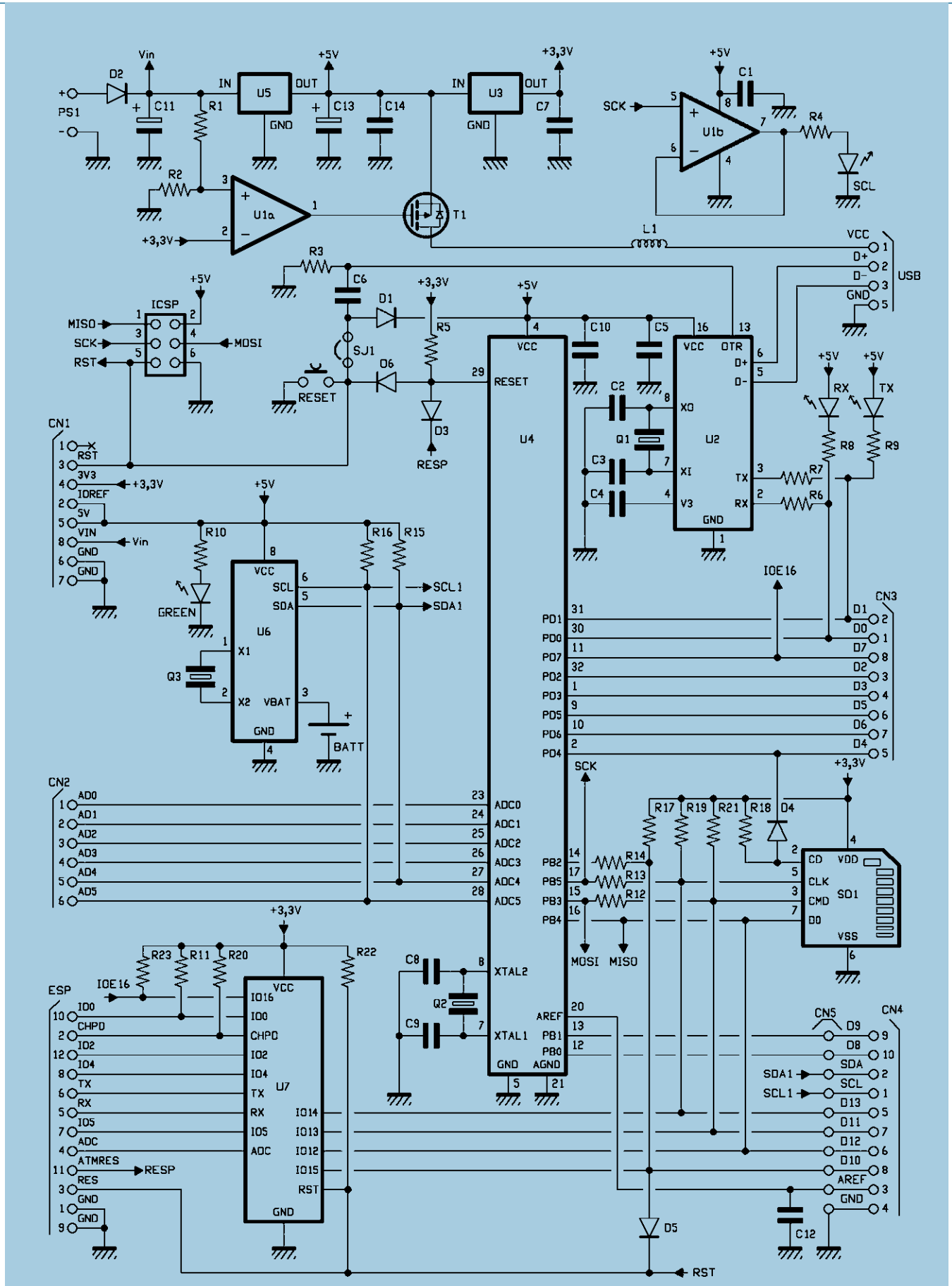
Tra le realizzazioni possibili ci sono, per esempio:

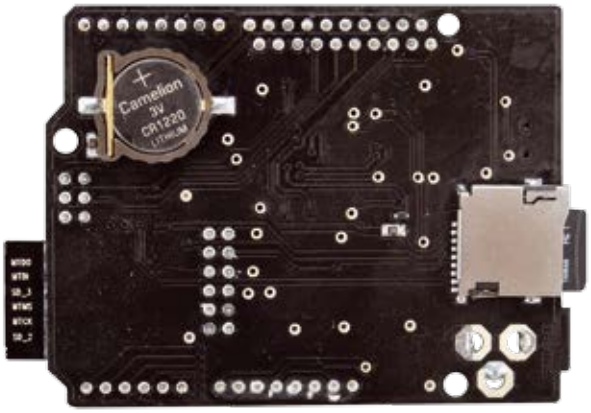
- sistemi di home automation gestibili via Internet tramite un Web Browser;
- data logger portatili in grado di connettersi e scaricare i dati sulla rete quando si entra nel campo di copertura di una rete WiFi;
- robot controllabili via rete e in grado di trasmettere tramite essa i dati rilevati da sensori.

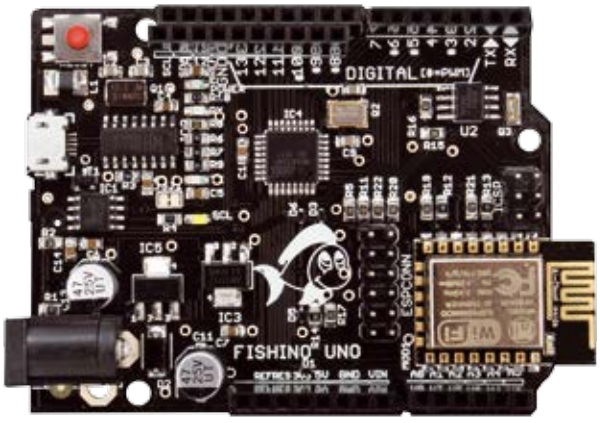
L'utilizzo di un modulo WiFi a basso costo ma con firmware da noi "hackerato" per ottenere prestazioni elevate e capace di funzionare anche come access point, ovvero senza la necessità di una struttura di rete WiFi esistente, permette il controllo via cellulare in qualsiasi momento, anche in assenza di copertura di rete, rendendo il dispositivo sempre interattivo. La possibilità di eseguire l'aggiornamento degli sketch via Internet, già prevista a livello hardware, permetterà inoltre di avere dispositivi sempre aggiornati senza la necessità di doverli collegare fisicamente ad un computer.

SCHEMA ELETTRICO

Il Fishino Uno, allo stesso modo di Arduino Uno, può essere alimentato tramite sia la porta USB che il connettore per l'alimentazione esterna. L'alimentazione viene automaticamente commutata su quella proveniente dal connettore esterno quando ai capi del medesimo (oppure all'ingresso Vin) viene applicata una tensione sufficiente al funzionamento del regolatore lineare U5.
La tensione giunge dal connettore di alimentazione al regolatore attraverso il diodo Schottky D2, utilizzato come protezione contro l'inversione della polarità; è stato scelto uno Schottky al posto di un più tradizionale diodo in silicio per la più bassa caduta di tensione che presenta: 0,3÷0,4 volt contro gli 0,7 circa dei diodi in silicio; questo, e l'aver utilizzato un regolatore lineare "low dropout" (a bassa caduta di tensione) permette di alimentare la scheda già con 6,6 volt (5 necessari ai circuiti + 0,4 di caduta sul diodo + 1,2 di caduta massima sul regolatore). La massima tensione di alimentazione ammissibile dipende invece dalla dissipazione del regolatore lineare utilizzato; si sconsiglia di superare i 12 V e, se possibile, di restare intorno ai 9 volt. Il regolatore dispone comunque di protezioni interne che lo disattivano quando la dissipazione risulta eccessiva.
La tensione in ingresso Vin (dopo il diodo di protezione) viene inoltre inviata all'operazionale U1A utilizzato per la commutazione dell'alimentazione tramite porta USB.
Quando la tensione Vin supera i 6,6 volt la tensione all'ingresso non invertente dell'operazionale, utilizzato qui come comparatore, dimezzata tramite il partitore costituito dalle resistenze R1 ed R2, supera quella di riferimento di 3,3 volt all'ingresso invertente e l'uscita commuta a livello positivo, provocando l'interdizione del MOSFET a canale P siglato T1.
Guardando da vicino quest'ultimo, il collegamento è apparentemente strano: l'alimentazione entra dal Drain ed esce dal Source, passando in contemporanea attraverso il diodo interno di clamping; ciò significa che la tensione proveniente dall'ingresso USB passa comunque attraverso il diodo e va ad alimentare il circuito anche se il MOSFET è interdetto.
A che serve quindi il MOSFET? Apparentemente con il solo diodo la commutazione sarebbe assicurata, perché se la tensione al catodo è superiore a quella dell'anodo (USBVCC) il diodo viene interdetto scollegando quindi l'alimentazione USB.
Il motivo della presenza del MOSFET (e circuiteria annessa) è da cercarsi sempre nella caduta di tensione provocata dal diodo, che farebbe diminuire l'alimentazione dai 5 volt della linea USB a 4,2÷4,6 volt circa, caduta evitata dal MOSFET stesso una volta entrato in conduzione.
A completare l'alimentazione è presente l'integrato U3, che fornisce in uscita la tensione a 3,3 V necessaria, tra l'altro, per la scheda SD ed il modulo WiFi.
A differenza dell'Arduino originale, che fornisce poche decine di mA sulla linea a 3,3 volt, nel Fishino sono utilizzabili circa 7÷800 mA a seconda del consumo sulla linea a 5 Volt.
L'interfaccia USB di Fishino è stata realizzata, diversamente dall'Arduino originale, tramite un chip CH340G (siglato U2) in sostituzione al più noto FT232 o altre soluzioni più o meno complicate.


PS1
D2
Vin
C11
R1
U5
IN
OUT
GND
+5V
C13
C14
U3
+3,3V
C7
SCK
U1b
C1
R4
SCL
R2
U1a
T1
L1
VCC
USB
D+
D-
GND
R3
C6
D1
SJ1
D6
R5
RESET
D3
RESP
ICSP
MISO
MOSI
RST
CN1
3V3
IOREF
5V
VIN
U4
C10
C5
C2
Q1
C3
C4
U2
XO
XI
V3
OTR
RX
TX
R8
R9
R7
R6
IOE16
CN3
PD1
PD0
PD7
PD2
PD3
PD5
PD6
PD4
D1
D0
D7
D2
D3
D5
D6
D4
R10
GREEN
U6
X1
X2
VBAT
Q3
BATT
R16
R15
SCL1
SDA1
CN2
AD0
AD1
AD2
AD3
AD4
AD5
ADC0
ADC1
ADC2
ADC3
ADC4
ADC5
R17
R19
R21
R18
R14
R13
R12
PB2
PB5
PB3
PB4
MOSI
MISO
SD1
CD
CLK
CMD
DO
VDD
VSS
R23
R11
R20
R22
ESP
IOE16
IO0
CHPD
IO2
IO4
TX
RX
IO5
ADC
ATMRES
RES
U7
IO16
IO14
IO13
IO12
IO15
RST
C8
C9
Q2
XTAL2
XTAL1
AREF
PB1
PB0
AGND
CN5
CN4
D9
D8
SDA
SCL
D13
D11
D12
D10
AREF
GND
C12
D5





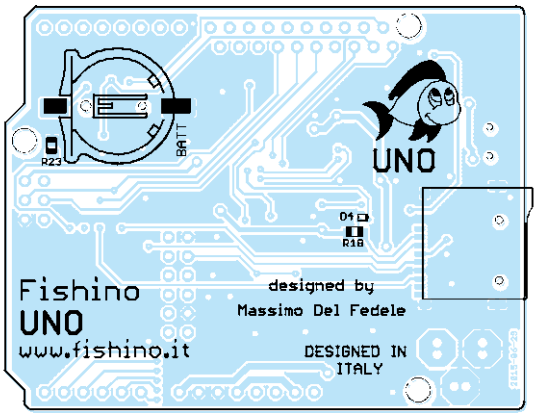
Elenco Componenti:

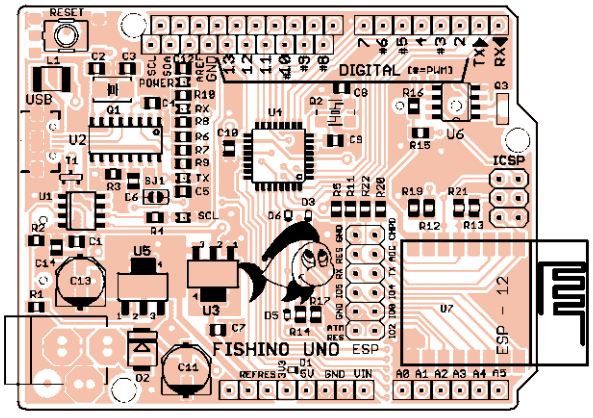
R1: 10 kohm (0805)
R2: 10 kohm (0805)
R3: 1 kohm (0805)
R4: 1 kohm (0805)
R5: 10 kohm (0805)
R6: 1 kohm (0805)
R7: 1 kohm (0805)
R8: 2,4 kohm (0805)
R9: 1 kohm (0805)
R10: 470 ohm (0805)
R11: 10 kohm (0805)
R12: 1 kohm (0805)
R13: 1 kohm (0805)
R14: 1 kohm (0805)
R15: 10 kohm (0805)
R16: 10 kohm (0805)
R17: 3,3 kohm (0805)
R18: 10 kohm (0805)
R19: 3,3 kohm (0805)
R20: 10 kohm (0805)
R21: 3,3 kohm (0805)
R22: 10 kohm (0805)
R23: 10 kohm (0805)
C1: 100 nF ceramico (0805)
C2: 22 pF ceramico (0805)
C3: 22 pF ceramico (0805)
C4: 1 μF ceramico (0805)
C5: 100 nF ceramico (0805)
C6: 1 μF ceramico (0805)
C7: 1 μF ceramico (0805)
C8: 22 pF ceramico (0805)
C9: 22 pF ceramico (0805)
C10: 100 nF ceramico (0805)
C11: 47 μF 16VL elettrolitico (Ø 6mm)
C12: 100 nF ceramico (0805)
C13: 47 μF 16VL elettrolitico (Ø 6mm)
C14: 100 nF ceramico (0805)
D1: RB521S
D2: M7
D3: RB521S
D4: RB521S
D5: CD1206-S01575
D6: RB521S
BATT: Porta batterie CH291-1220LF
U1: LMV358L
U2: CH340G
U3: NCP1117ST33T3G
U4: ATMEGA328P-AU
U5: NCP1117ST50T3G
U6: DS1307
U37: ESP12

La scelta è stata dettata principalmente da motivi di costo e di semplificazione circuitale a parità di prestazioni. L'integrato per funzionare necessita di pochissimi componenti esterni: un quarzo a 12 MHz (Q1), due condensatori per garantire la stabilità dell'oscillatore (C2 e C3) ed un condensatore di disaccoppiamento per il regolatore interno a 3,3V (C4).
Il circuito può infatti essere alimentato indifferentemente a 3,3 volt (collegando il pin V3 al Vcc) oppure a 5 volt, inserendo un condensatore di disaccoppiamento tra il pin V3 e la massa.
Il componente fornisce in uscita tutti i segnali di un'interfaccia RS232, ovvero i due segnali di trasmissione/ricezione dati (Rx e Tx) ed i segnali di controllo (CTS, DSR, DXD, DTR e RTS); nel nostro circuito vengono utilizzati solo i segnali dati (RX e TX) e il DTR per generare l'impulso di reset ogniqualvolta viene aperta la porta seriale, in modo da rendere possibile il caricamento degli sketch senza dover resettare manualmente il dispositivo.
Torneremo in seguito sulla circuiteria di reset per spiegare che rispetto all'originale è stata modificata in modo da permettere in futuro la riprogrammazione dell'Atmega via WiFi.
Non disponendo, l'integrato utilizzato, di due uscite separate per i LED TX ed RX (che segnalano le attività di trasmissione e/o ricezione) questi ultimi sono stati inseriti direttamente sulle due rispettive linee dati, con una scelta accurata dei valori di resistenza in modo da non provocare un'attenuazione eccessiva.
La scelta dei valori è stata inoltre calibrata per permettere alla stessa interfaccia USB/Seriale di riprogrammare il firmware del modulo WiFi senza dover ricorrere ad adattatori esterni.
Per quanto riguarda l'ATmega328P, lo schema di Fishino non si discosta da quello di Arduino Uno: il controller è il medesimo, solo in formato SMD per esigenze di spazio sulla board, correda-

T1: FDN340P
L1: MF-MSMF050-2
500mA (1812)
POWER: LED verde (0805)
RX: LED giallo (0805)
TX: LED blu (0805)
SCL: LED bianco (0805)
Q1: Quarzo 12 MHz
Q2: Quarzo 16 MHz
Q3: Quarzo 32.768 KHz
RESET: Microswitch TS42
SD1: Connettore micro-SD

Varie:
- Plug alimentazione
- Connettore micro-USB
- Strip femmina 6 vie (1 pz.)
- Strip femmina 8 vie (2 pz.)
- Strip femmina 10 vie (2 pz.)
- Strip maschio 3 vie (2 pz.)
- Strip maschio 6 vie (2 pz.)
- Circuito stampato S1225

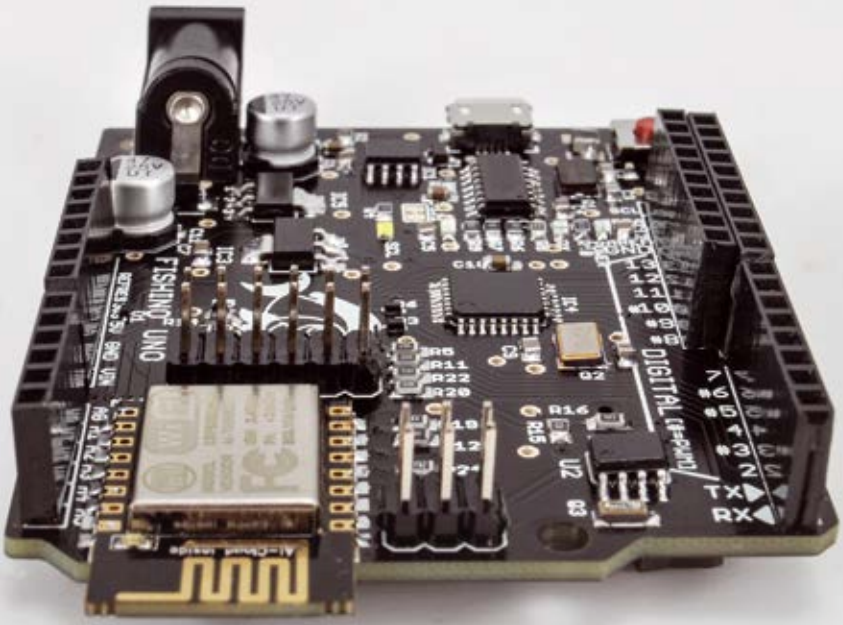




to dall'usuale quarzo a 16 MHz, dai due condensatori sul circuito oscillante e da un certo numero di condensatori di disaccoppiamento sulle linee di alimentazione. Una piccola parentesi su questi ultimi: spesso negli schemi si vedono alcuni (a volte molti) condensatori in parallelo all'alimentazione; qualcuno si chiederà perché al posto dei tanti condensatori non ne abbiamo usato uno solo di capacità equivalente. Ebbene, il motivo è che i moderni circuiti integrati digitali lavorano a frequenze elevate e con segnali impulsivi, che causano grosse variazioni di assorbimento di corrente sui pin di alimentazione dovute alla resistenza delle piste; per questo è necessario applicare in prossimità dei pin di alimentazione dei condensatori che filtrino i disturbi prima che si propaghino al resto del circuito. Un condensatore solo non servirebbe.
Ora una nota sui connettori di I/O: a Fishino è stato aggiunto un piccolo connettore a 10 pin affiancato a quello standard, ma leggermente sfalsato in modo da poter utilizzare una scheda preforata standard per gli shield, risolvendo quindi l'annoso problema del passo incompatibile dei connettori laterali di Arduino.
Tutta la circuiteria di Fishino opera a 5 volt, mentre sia le schede MicroSD che il modulo WiFi funzionano a 3,3 volt e, soprattutto, non sono 5V-tolerant; questo significa che fornendogli segnali TTL a 5 volt si corre il rischio di bruciarli.
Per quanto riguarda il modulo WiFi, da prove effettuate non abbiamo riscontrato alcun problema fornendogli dati TTL a 5 volt, ma, per evitare possibili guasti a medio/lungo termine, abbiamo preferito rispettare le specifiche tecniche ed inserire un circuito di adattamento dei livelli.
Questo è realizzato molto semplicemente tramite partitori resistivi, per quanto riguarda la direzione Atmega --> SD e WiFi, mentre in direzione opposta si sfrutta il fatto che il valore H dei circuiti a 3,3 volt (da 2,95 volt circa in su) è al limite ma dentro il range compatibile con il livello alto a 5 volt.
Si sono quindi risparmiate complesse circuiterie con transistor e/o traslatori integrati.
L'unico "difetto", se così si può dire, dell'approccio utilizzato, è che i partitori resistivi devono sia avere un'impedenza sufficientemente alta per non sovraccaricare le uscite dell'Atmega328P e per non aumentare inutilmente il consumo di corrente, sia avere un'impedenza sufficientemente bassa da non causare perdite di segnali in alta frequenza.
I valori scelti sono un compromesso tra i due requisiti e permettono il buon funzionamento dell'interfaccia anche alla massima velocità. I partitori sono costituiti dalle coppie R13-R21, R12-R19 e R14-R17. Un valore in ingresso a 5 Volt viene convertito a:

Vo = 5 Volt • 3.3K / (1K + 3.3K) = 3,83 Volt

valore leggermente superiore ai 3,3 volt teorici ma accettabile secondo le specifiche, essendo ammessi solitamente valori fino a 0,6 Volt superiori all'alimentazione, quindi 3,3 + 0,6 = 3,9 V.



Interfaccia scheda MicroSD

L'interfaccia è semplicissima e rispecchia lo shield SD (o analoghi combinati); funziona attraverso le linee SPI tra cui MOSI (dati dall'Atmega verso la SD, Master Out Slave In), MISO (dati dalla SD all'Atmega, Master In Slave Out) e SCK (clock). I valori verso la scheda SD sono ovviamente ridotti dagli adattatori di livello di cui al paragrafo precedente.
La selezione della scheda avviene tramite la linea SDCS, attiva a livello basso. In questo caso l'adattamento di livello viene effettuato tramite una resistenza (R18) verso il positivo ed un diodo (D4) che permette il solo passaggio dei valori negativi.
Lo schema scelto consente di avere la scheda in stand-by quando il segnale SDCS (connesso al pin digitale 4 del Fishino) non è utilizzato; col pin in modalità threestate (ovvero ad alta impedenza) il diodo non conduce e sull'ingresso SDCS è presente un valore alto che disattiva la scheda.
L'interfaccia è totalmente compatibile con gli shield di Arduino, quindi utilizza le stesse librerie esistenti per il suo funzionamento, come si vedrà negli esempi presentati in seguito.

Modulo WiFi

Se l'Atmega può essere considerato il cervello del Fishino, il modulo WiFi è sicuramente la sua porta d'accesso al mondo esterno, ed è il principale motivo a monte dello sviluppo della scheda.
L'idea di creare Fishino è nata infatti dall'esigenza di sviluppare un sistema di Home Automation con possibilità di controllo via Internet. In precedenza per poterla realizzare era necessario usare Arduino con annesso modulo WiFi o ethernet, con costi ed ingombri decisamente più elevati e, nel caso dell'ethernet, la necessità di una connessione cablata alla rete.
L'inserimento della connettività WiFi direttamente nel Fishino a costi ragionevoli è stato reso possibile dall'immissione sul mercato dei moduli WiFi ESP8266.
Questi moduli contengono un processore a 32 bit, una Flash di programma molto capiente (da 1 a 4 MBit), circa 90 kByte di RAM di cui 32 disponibili per l'utente, un completo stack WiFi e tutta la componentistica di contorno necessaria al loro funzionamento, fino all'antenna integrata sul PCB.
A prima vista, una potenza di calcolo di questo livello (decisamente superiore a quella dello stesso Atmega utilizzato da Arduino), suggerirebbe l'utilizzo del solo modulo programmandolo direttamente, vuoi con il SDK (Software Development Kit) fornito dal produttore (piuttosto complesso da utilizzare), oppure tramite una serie di strumenti che ne permetta la programmazione con l'IDE di Arduino, quasi fosse un nuovo modello del medesimo. Dove sta il problema, allora? Ebbene, per prima cosa le architetture sono molto diverse, Anche se l'IDE di Arduino è stata adattata per compilare e caricare codice direttamente sull' ESP, la compatibilità è ancora troppo limitata. Inoltre si perde la possibilità di utilizzare l'enorme quantità di librerie e shield di cui Arduino è dotato. Infine l'ESP ha un numero ridottissimo di pin di I/O digitale ed un solo input analogico, cosa che ne limita fortemente le possibilità d'impiego.
Si è scelto quindi di integrare in una board compatibile con Arduino UNO il modulo WiFi, cercando di renderlo il più simile possibile nell'uso agli analoghi shield WiFi ed Ethernet, grazie ad un'apposita libreria di cui parleremo in seguito.
L'utilizzo del modulo ESP è stato inizialmente difficoltoso e fonte di vari tentativi più o meno fallimentari, a causa principalmente del firmware fornito dal costruttore.
Questo infatti è realizzato per comunicare attraverso la porta seriale e non, come negli shield originali, attraverso la porta SPI.
I vantaggi della porta seriale sono ovviamente la semplicità d'uso (basta collegare un terminale seriale al modulino per

dialogare con esso) e la necessità di due sole linee di dati per la comunicazione.
A fronte di suddetti vantaggi, però, si hanno i seguenti problemi:

- velocità limitata; la porta seriale può viaggiare ad un massimo di 2-300 kbit/secondo, nei casi migliori (velocità più elevate sono possibili ma richiedono connessioni molto corte ed una velocità notevole del controller connesso);
- mancanza di handshake o controllo di flusso dei dati; pur prevedendo alcuni moduli ESP 2 linee di controllo di flusso hardware, il firmware precaricato non le utilizza, ed è quindi impossibile bloccare un trasferimento dati dal modulo verso l'Atmega.

Riguardo a questo secondo problema, immaginiamo di aprire una pagina web contenente da qualche decina a qualche centinaio di kilobyte di dati (cosa usuale per una pagina web), che vengono inviati dal modulo al Fishino, il quale dispone di soli 2 kB di RAM, senza possibilità di comunicare al modulo stesso di attendere l'elaborazione dei dati; ecco, appare evidente cosa accade se non si può controllare il trasferimento dei dati.
La porta seriale hardware dal lato Fishino è necessaria per ottenere velocità decenti. Con la seriale software non siamo riusciti ad ottenere velocità affidabili sopra i 57 kbaud.
Dopo svariati tentativi infruttuosi di realizzare una libreria Arduino che permettesse l'uso del modulo ESP con il firmware standard, si è deciso di risolvere il problema a monte riscrivendo un firmware ad hoc, che permettesse l'utilizzo della ben più veloce interfaccia SPI.
A fronte del lungo tempo di sviluppo software del medesimo (la documentazione reperibile è decisamente scarsa) vantaggi sono stati subito evidenti:

- velocità; il bus SPI può viaggiare a 8 MHz, con un clock dell'Atmega di 16 MHz. Anche considerando che il protocollo richiede dei dati di

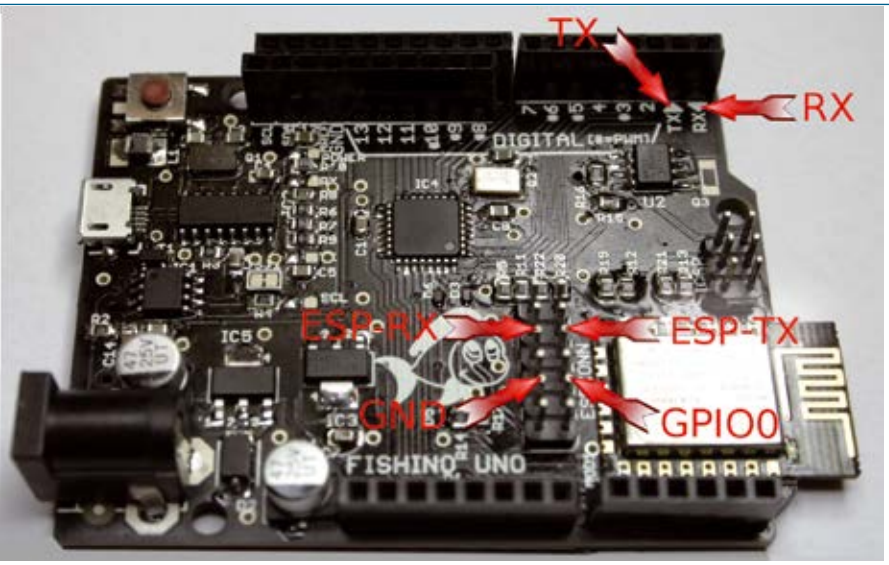


Fig. 1

controllo (cosa peraltro necessaria anche con l'interfaccia seriale) la velocità ottenibile è di almeno un ordine di grandezza superiore a quella della porta seriale, ovvero circa 10 volte tanto;

- controllo di flusso; il protocollo SPI è totalmente gestito dal master (in questo caso l'Atmega). Il controller richiede i dati quando è in grado di elaborarli e non è costretto a "rincorrere" il modulo WiFi;
- possibilità di spostare buona parte dell'impegno sia in termini di elaborazione che di memoria necessari sull'ESP (basti pensare ai 32 K di RAM disponibili contro i 2 dell'Atmega e gli 80/160 MHz di clock contro i 16);
- possibilità di incapsulare tutte le funzionalità del modulo in una libreria dall'uso praticamente identico alle varie WiFi/ethernet standard.

Il firmware realizzato e l'annessa libreria Arduino permettono di ottenere velocità di trasferimento dati (misurata da browser a SD card) di 60÷70 kB/s, quindi leggermente superiore alla velocità ottenibile con la scheda ethernet originale via cavo e decisamente superiori (2÷4 volte tanto) rispetto allo shield WiFi originale; il software ha comunque ulteriori margini di ottimizzazione che verranno sfruttati in seguito. Futuri aggiornamenti del firmware aggiungeranno ulteriori funzionalità, tra cui l'utilizzo della porta seriale del modulino come seriale hardware aggiuntiva, la possibilità di programmare l'Atmega direttamente via WiFi ed altri.

Passiamo ora alla descrizione dello schema elettrico della sezione WiFi, che contiene alcune particolarità degne di nota dovute principalmente alle problematiche hardware del modulo ESP. Come si nota dal simbolo sullo schema, il modulo contiene varie connessioni denominate GPIO (General Purpose Input Output) utilizzabili per vari compiti a seconda della programmazione e/o dello stato del modulo stesso. I pin sono infatti utilizzabili come gli I/O digitali (e un input analogico) di Arduino, salvo che quasi tutti hanno funzionalità aggiuntive, alcune delle quali utilizzate all'avvio dell' ESP che ne rendono difficoltoso l'utilizzo. Di seguito una breve descrizione di tali pin.

- GPIO0 e GPIO15 : oltre ad essere utilizzabili come input/output digitale, servono per selezionare la modalità d'avvio al boot del modulo. Quest'ultimo può infatti essere avviato da flash interna (funzionamento normale, GPIO15 a 0 e GPIO0 a 1), da interfaccia seriale, utilizzato per la riprogrammazione del firmware (GPIO15 a 0, GPIO0 a 0) e boot da scheda SD esterna (non l'abbiamo mai utilizzato, GPIO15 a 1, GPIO0 ininfluente). A complicare le cose, il pin GPIO15 è anche utilizzato per selezionare il modulo nella modalità SPI slave, con valore attivo basso. Il modulo deve quindi partire con GPIO15 a 0 e, brevemente dopo l'avvio, questo dev'essere portato a 1 per liberare la porta SPI necessaria anche, ad esempio, per la scheda SD.
- GPIO12 (MISO), GPIO13(MOSI) e GPIO14(SCK), oltre ad essere degli I/O generici, insieme al suddetto GPIO15 sono utilizzati dall'interfaccia SPI.
- GPIO16, I/O generico ma utilizzato per il "risveglio" del modulo dalla funzione di deep sleep. Non lo utilizziamo come tale ma solo come pin di handshake verso l'Atmel. Su questo pin all'avvio del modulo sono presenti degli impulsi di "risveglio" da scartare prima che acquisti la sua funzionalità definitiva.
- GPIO2, GPIO4 e GPIO5 sono disponibili per l'uso come pins digitali, e saranno sfruttabili in una prossima versione del firmware come fossero estensioni dei pins digitali di Arduino.
- Rx e Tx costituiscono la porta seriale hardware del modulo e sono utilizzati anche in fase di programmazione del firmware. Anche qui, una prossima estensione del firmware ne permetterà l'uso come porta

seriale aggiuntiva che consentirà a Fishino di raddoppiare la sua connettività seriale.
- CH_PD è il pin di abilitazione del modulo. Portandolo a livello alto il modulo risulta abilitato (default), mentre un livello basso mette in stand-by l'ESP riducendone i consumi praticamente a zero.
- RESET è il reset hardware dell'ESP, attivo a livello basso
- ADC è l'ingresso analogico dell'ESP, diretto verso un convertitore A/D da 10 bit (1024 valori possibili).

Volendo utilizzare questi pin bisogna ricordare che non gli vanno applicati più di 3,3 V perché non sono 5V-tolerant.
Dello schema elettrico notiamo alcuni particolari, come il diodo D5, che serve quando Fishino (e quindi anche il modulo ESP, vedere di seguito la circuiteria di RESET) viene resettato, a portare a livello 0 il pin GPIO15 forzando quindi l'avvio dalla Flash interna; senza questo accorgimento un valore casuale alto sul GPIO15 impedirebbe l'avvio del modulo. La resistenza R23 a massa sul GPIO16 serve ad indicare che il modulo è impegnato nell'avvio (e quindi il GPIO16 non è ancora stato impostato come handshake); la libreria sfrutta questo segnale per rilevare se il modulo è pronto ad operare dopo il boot. Notate che sono state utilizzate le stesse linee digitali dell'Atmega sfruttate dagli shield WiFi ed ethernet, garantendo quindi la totale compatibilità con eventuali shield aggiuntivi.

Connettore ESP

Su questo connettore vengono riportati alcuni GPIO liberi del modulo ESP (per l'uso come porte) ed è presente lo spazio per alcuni ponticelli di configurazione.

- PIN 1-2: chiudendoli, il modulo ESP viene disattivato completamente (utile nel caso non sia necessario oppure si voglia montare uno shield che va in conflitto con gli I/O utilizzati). È anche possibile connettere il pin 2 ad un'uscita digitale di Fishino e controllare così via software se e quando accendere il modulo WiFi.
- PIN 3: RESET. Un livello zero su questo pin provoca il reset sia del modulo WiFi che dell'Atmega 328.
- PIN 4: ADC. Input del convertitore A/D a 10 bit dell'ESP.
- PIN 5 e 6: ESP-RX ed ESP-TX, rispettivamente (interfaccia seriale hardware del modulo WiFi). Utilizzati anche per riprogrammare il firmware.
- PIN 7-8: GPIO5 e GPIO4. Utilizzabili come pin di I/O digitale.
- PIN 9-10: chiudendoli con un jumper e premendo RESET, il modulo entra in modalità flash, per l'aggiornamento del firmware (vedremo i dettagli più avanti). Il PIN 10 corrisponde anche al GPIO0, utilizzabile come linea di I/O digitale.
- PIN 11-12: chiudendoli con un jumper si abilita la riprogrammazione dell'Atmega tramite WiFi. Al momento non è stato implementato il firmware necessario, ma sarà fatto nelle future versioni scaricabili dal nostro sito web. Il PIN12 corrisponde anche al GPIO2, utilizzabile anche come linea di I/O digitale.

Circuiteria di RESET

La sezione del RESET risulta più complicata rispetto a quella di Arduino originale per i seguenti motivi:
- occorre resettare sia l'Atmega che l'ESP alla pressione del tasto di reset, all'avvio e alla richiesta di programmazione da parte dell' IDE;
- per poter eseguire la programmazione dell'Atmega tramite WiFi, il modulo ESP dev'essere in grado di resettare l'Atmega stesso senza a sua volta auto-resettarsi.

Fig. 2

Listato 1

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

// here pur SSID of your network
// inserire qui lo SSID della rete WiFi
#define SSID       ""

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

// 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

// NOTE : for prototype green version owners, set SD_CS to 3 !!!
// NOTA : per i possessori del prototipo verde di Fishino, impostare SD_CS a 3 !!!
const int SD_CS = 4;

// END OF CONFIGURATION DATA
// FINE DATI DI CONFIGURAZIONE
////////////////////////////////////////////////////////////////////////
```

Iniziamo dal segnale DTR che esce dall'interfaccia USB/Seriale (U2, CH340G): esso, come anticipato, viene posto a livello basso quando la porta seriale viene aperta. Attraverso il condensatore C6 (scelto da 1 µF ceramico, contro i 100 nF dell'originale, per allungare l'impulso di reset) viene generato un breve impulso che, passato attraverso il jumper SMD SJ1 (tagliando il quale è possibile disattivare l'auto-reset), raggiunge la linea di "reset esterno", alla quale sono connessi anche il pulsante di reset ed il pin 5 sul connettore di programmazione (ICSP). A differenza del circuito originale, nel Fishino è presente un diodo (D6) tra la linea di RESET ed il pin dell'Atmega. Lo scopo di questo diodo (e del diodo D3 che vedremo in seguito) è di poter resettare solo l'Atmega senza peraltro veicolare il segnale anche all'ESP.
Premendo il pulsante RESET, o connettendo la seriale, l'impulso di reset raggiunge sia l'Atmega (attraverso D6) che l'ESP (direttamente), resettandoli entrambi.
Un segnale sulla linea ATRES-ESP, generato dall' ESP (nel caso si sia abilitata la riprogrammazione attraverso il WiFi) raggiunge attraverso D3 la linea di reset dell'Atmega ma, a causa di D6, non può propagarsi all'ESP stesso. Tramite questo sistema abbiamo quindi dato la possibilità al modulo WiFi di controllare la linea di reset dell'Atmega che, unitamente all'interfaccia SPI, ne permette la riprogrammazione senza nemmeno la necessità di un bootloader precaricato.
In pratica, una volta completato lo sviluppo nel firmware, sarà possibile non solo riprogrammare via WiFi l'Atmega ma farlo utilizzando anche lo spazio normalmente riservato al bootloader.

Modulo RTC

Concludiamo lo schema elettrico con il modulo RTC (Real Time Clock), costituito da un classico DS1307 della Maxim, un quarzo a 32 kHz, una batteria di backup ed un paio di resistenze sulla linea I²C. Lo schema è quanto di più classico esista ed è completamente compatibile con le librerie di Arduino esistenti; tutte le funzioni sono gestite tramite linea I²C (SDA/SCL).

Driver USB

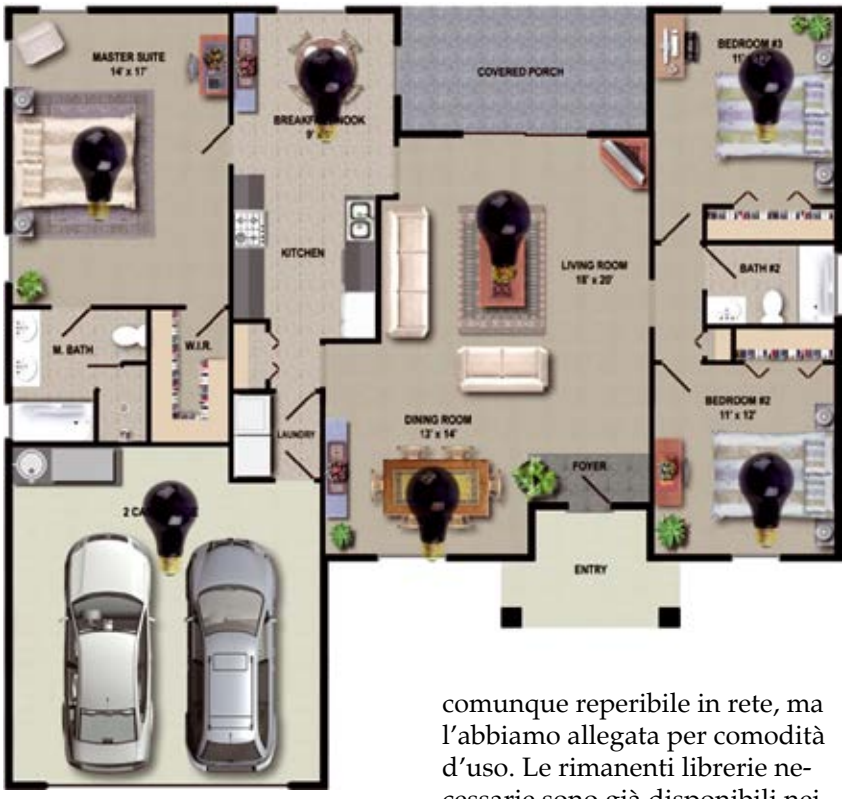
Per il converter USB/seriale Fishino richiede driver specifici, almeno per Windows fino alla versione 7 (si scaricano dal sito *www.fishino.it* o *www.fishino.com*, nella sezione Download); pare che dalla 8 in poi i driver siano inclusi. Per l'ambiente Linux, per contro, i driver non sono necessari, essendo già presenti nel kernel.

AGGIORNAMENTO FIRMWARE DEL MODULO WIFI

Fishino viene fornito con la versione del firmware disponibile al momento dell'assemblaggio. Essendo in fase di continuo sviluppo, conviene aggiornare periodicamente il firmware. Le librerie di Arduino disponibili sono infatti aggiornate continuamente in base alle nuove possibilità offerte dal firmware.
La procedura di aggiornamento è semplificata da un programma apposito, disponibile sia per Windows che per Linux, che esegue l'operazione in modo completamente automatico e a prova di errore. I passi per l'aggiornamento sono i seguenti.

1. Caricare uno sketch che non utilizzi la porta seriale, come ad esempio BLINK (quello che fa lampeggiare il LED sulla scheda); ciò 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 il Fishino, al 99% il problema è che avete caricato uno sketch sbagliato.
2. Connettere il TX di Fishino con la porta ESP-TX sul connettore ESP, e l'RX di Fishino con la porta ESP-RX sul connettore ESP (vedere **Fig. 1**).
3. Connettere GPIO0 a massa tramite un cavetto o un ponticello sempre sul connettore ESP (**Fig. 1**). Collegare Fishino al PC (o premere il pulsante di RESET se già connesso);
4. Lanciare il programma Fishi-

***Fig. 3** - Schermata della demo.*



noFlasher, assicurandosi che il PC sia connesso ad Internet.

Se i collegamenti sono stati eseguiti correttamente, il programma rileverà la porta a cui è connesso Fishino, 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 e permettendo comunque la selezione delle versioni precedenti nel caso si voglia fare un downgrade, come in **Fig. 2**. Premendo il pulsante "Flash" verrà avviata la procedura di aggiornamento alla fine della quale apparirà un messaggio di conferma. Per terminare il programma occorre premere il pulsante "Esci".
Nel caso Fishino non venga rilevato 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 un Fishino sia connesso 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.

LIBRERIE DISPONIBILI

Attualmente sono disponibili due librerie per la gestione del Fishino: la *Fishino* e la *Fishino WebServer*.
La libreria Fishino è l'esatto equivalente della libreria Ethernet o WiFi di Arduino.
In questa libreria vengono definite le classi FishinoClass (gestione a basso livello, analoga alla EthernetClass o WiFiClass di Arduino), e le rispettive classi FishinoClient (l'analogo della EthernetClient e WiFiClient) e FishinoServer (EthernetServer e WiFiServer).
Queste classi sono utilizzate in modo pressochè identico alle originali, quindi negli sketch esistenti che utilizzano l'Ethernet o il WiFi basta cambiare il tipo delle varie variabili per ottenerne il funzionamento con il WiFi di Fishino. Le uniche (leggere) differenze sono sull'inizializzazione, essendo il modulo WiFi di Fishino dotato di caratteristiche aggiuntive rispetto al WiFi originale. La libreria FishinoWebServer è il porting su Fishino della nota TinyWebServer; consente la creazione di un completo server web sulla scheda.
È stata inoltre inserita la libreria Flash nel pacchetto di download visto che è utilizzata dalla FishinoWebServer per spostare le costanti nella memoria Flash liberando così la poca RAM a disposizione. Questa libreria è comunque reperibile in rete, ma l'abbiamo allegata per comodità d'uso. Le rimanenti librerie necessarie sono già disponibili nei vari download dell'IDE di Arduino; sono in particolare necessarie la SD (per la gestione delle schede MicroSD) e la RTClib per la gestione del modulo RTC con il DS1307, ed altre librerie di sistema. Le librerie sono in continuo sviluppo e presto verranno corredate di funzionalità aggiuntive, in particolar modo per la gestione dei PIN di I/O e della porta seriale aggiuntiva disponibili sul modulo WiFi ed altro.

DEMO HOME AUTO

Per concludere questo articolo presentiamo la demo FishinoHomeAuto che mostra alcune delle caratteristiche più interessanti di Fishino all'opera. Si tratta di un piccolo server web che consente tramite browser la gestione dei pin di I/O digitali sulla board.
Ci limiteremo a descriverne brevemente il funzionamento e l'utilizzo affinché possiate provarne la funzionalità immediatamente.
Premettiamo che la demo non vuol essere un applicativo completo di home automation ma la

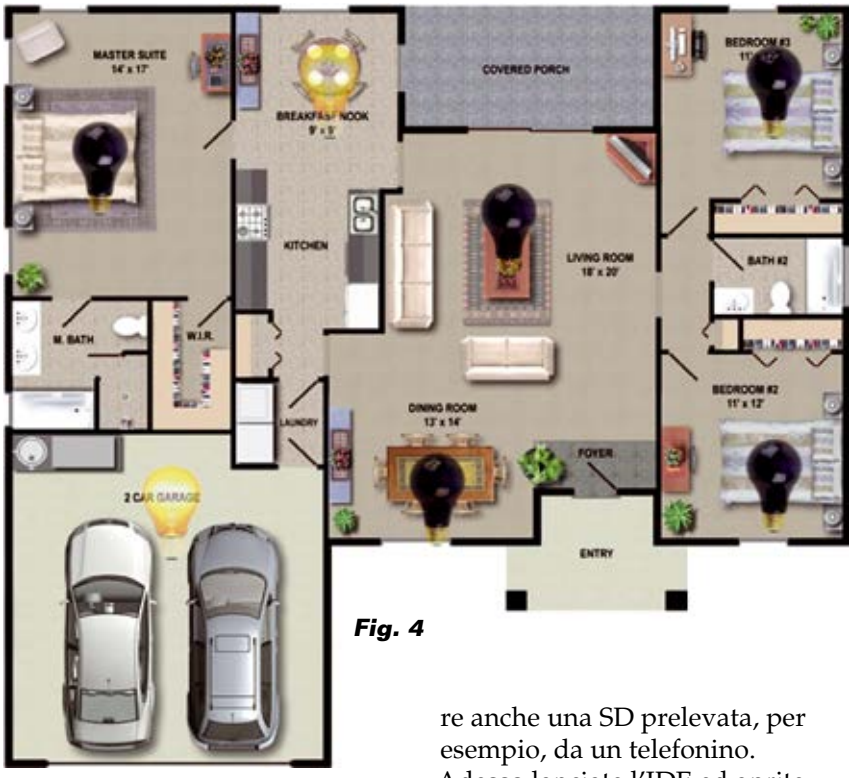


Fig. 4

base per scriverne uno; in particolare, sono gestiti solo output digitali (rappresentati dalle lampadine alle figure seguenti). Il software è stato comunque pensato con l'estensibilità in mente, quindi successivamente verranno implementate le funzioni di input e quelle analogiche.
Per prima cosa dovete decomprimere il file *FishinoLibs.zip* nella cartella "libraries". Una volta eseguito avrete tre nuove librerie: Fishino, FishinoWebServer e Flash. Ora decomprimete il file FishinoHomeAuto nella cartella "sketches": apparirà una cartella chiamata FishinoHomeAuto e, dentro questa, alcuni file e la sottocartella **STATIC**. Copiate tutto il contenuto della cartella **STATIC** (non la cartella, ma solo i files in essa contenuti) nella directory radice di una scheda MicroSD. Non occorre cancellare quel che c'è dentro; basta copiare i file nella cartella principale. Lo sketch non altera i dati nella scheda, quindi potrete utilizzare anche una SD prelevata, per esempio, da un telefonino.
Adesso lanciate l'IDE ed aprite lo sketch FishinoHomeAuto. All'inizio di questo è presente una parte di configurazione, da modificare per adattarla alla propria rete WiFi (**Listato 1**). Leggere i commenti e modificare le impostazioni come indicato. Salvare lo sketch e caricarlo sul Fishino.
A questo punto:

- inserire la MicroSD nel fishino;
- se si desidera vedere cosa succede, aprire il monitor seriale sull' IDE;
- volendo vedere i LED accendersi e spegnersi seguendo i comandi da web, collegare uno o più LED (con le relative resistenze in serie) ai pin di Fishino 2, 5, 6, 8, 9, 14 e 15 (queste sono le uscite digitali previste dalla demo, ognuna delle quali è associata ad una "stanza" nell'immagine che apparirà sul browser);
- premere il pulsante RESET;
- dal PC lanciare il browser e inserire nella barra degli indirizzi l'IP impostato nello sketch. Se avrete optato per l'IP dinamico le cose si complicano, visto che occorre determinare quale IP è stato assegnato a Fishino. Suggeriamo l'impostazione di un IP statico per le prime prove.

Se tutto è andato a buon fine, sul browser verrà visualizzata la pagina con la **Fig. 3**. La pagina è totalmente configurabile tramite i file presenti nella cartella STATIC e poi copiati sulla SD card. Cliccando su una delle lampadine spente (nere) l'immagine cambierà in una lampadina accesa (gialla) e contemporaneamente l'eventuale led connesso al Fishino si illuminerà. Cliccando nuovamente sulla lampadina accesa, questa tornerà nera ed il LED si spegnerà (**Fig. 4**).
Pur essendo una demo, il programma è sufficientemente configurabile: è possibile per esempio cambiare l'immagine della "casa", la posizione e le immagini delle lampadine, eccetera. Stiamo preparando una versione che preveda la lettura/scrittura di valori analogici, per esempio di temperatura o per regolare un termostato. ■

per il MATERIALE

La board Fishino (cod. FISHINOUNO) viene fornita montata e collaudata. Può essere acquistata presso Futura Elettronica al prezzo di Euro 36,00. 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