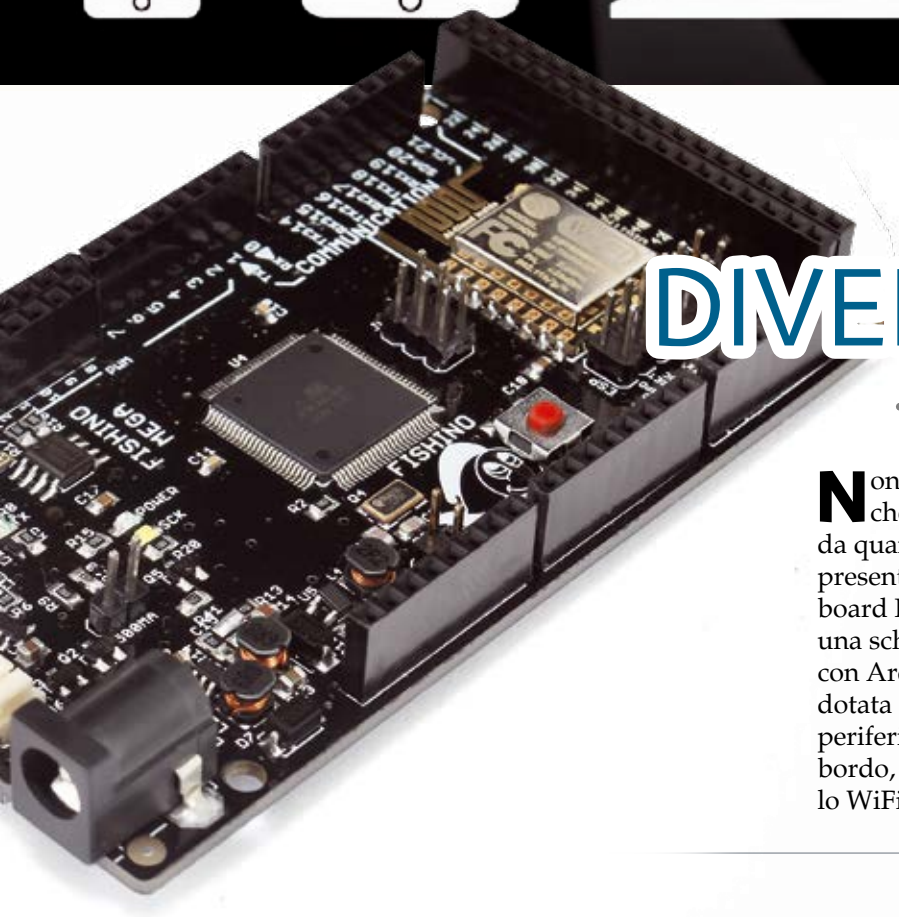
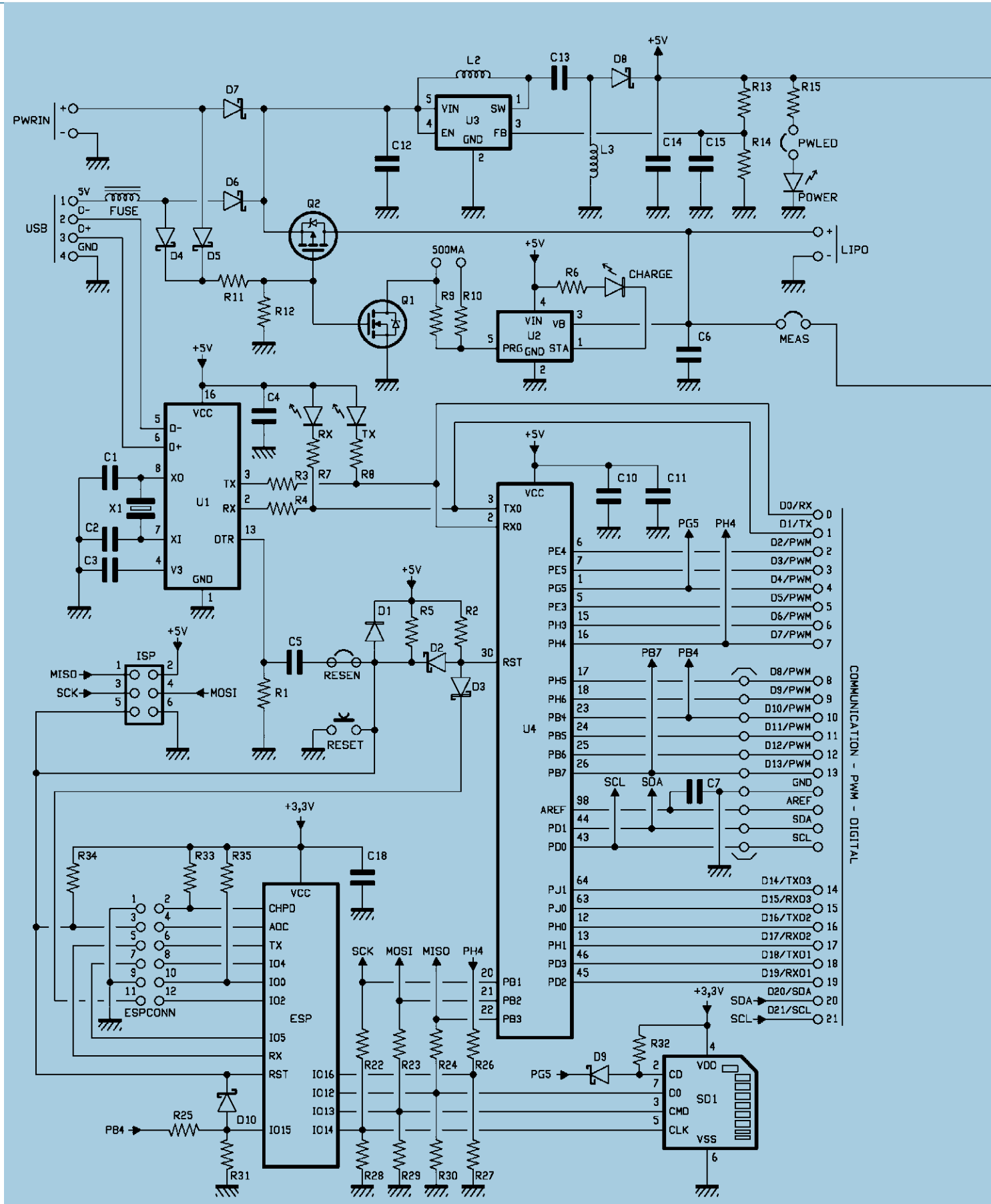
La nostra versione della popolare e potente Arduino MEGA, basata su un hardware rivisitato specialmente nella sezione di alimentazione.

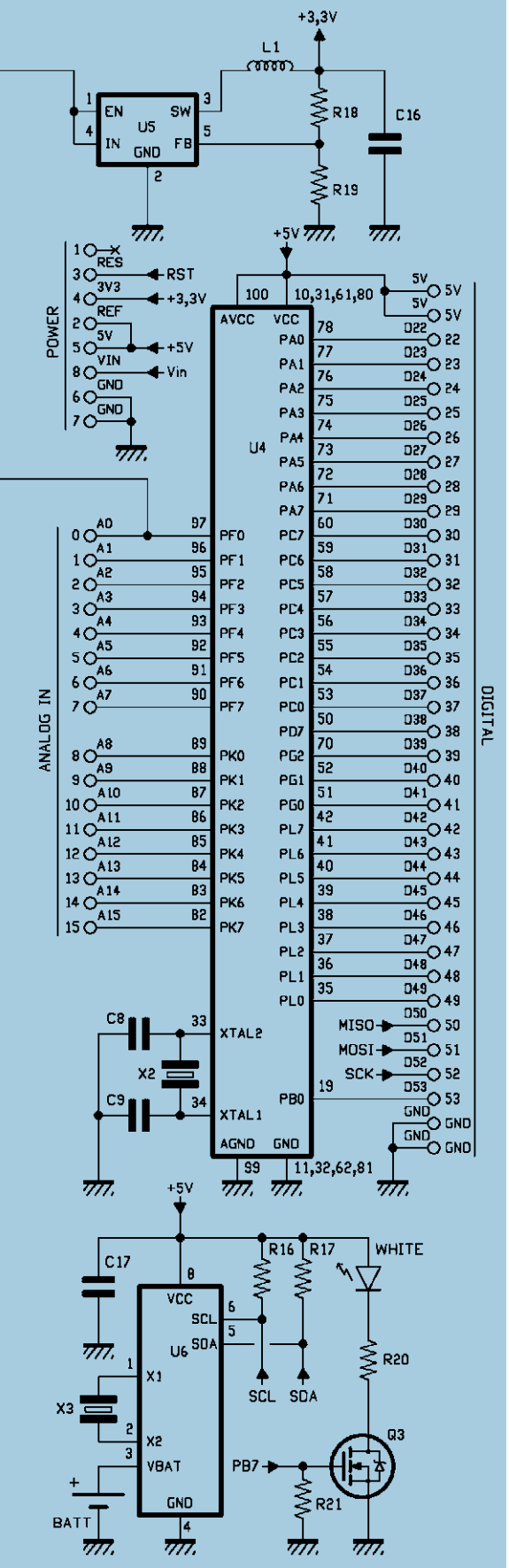


FISHINO DIVENTA MEGA

di MASSIMO DEL FEDELE

Non è passato neanche tanto tempo da quando abbiamo presentato la nostra board Fishino UNO, una scheda compatibile con Arduino UNO ma dotata di interessanti periferiche aggiuntive a bordo, quali un modulo WiFi, un lettore di schede microSD ed un RTC (Real Time Clock, orologio in tempo reale). Come anticipato allora, la board non voleva essere un "pezzo unico" ma la prima di una serie completa di schede Arduino-compatibili con prestazioni e funzionalità estese, di cui si


PWRIN
USB
FUSE
LIPO
PWLED
POWER
CHARGE
MEAS
500MA
U1
U2
U3
U4
U5
ESP
ESPCONN
ISP
RESET
RESEN
MISO
MOSI
SCK
SDA
SCL
+5V
+3,3V
COMMUNICATION - PWM - DIGITAL



sentiva la mancanza. In questo articolo presentiamo quindi la seconda scheda della serie: la Fishino MEGA. Si tratta di una board, compatibile con Arduino Mega, nel cui hardware abbiamo implementato una significativa novità: la possibilità di ottenere l'alimentazione a batteria.
Vediamo innanzitutto le caratteristiche complete della Fishino MEGA:

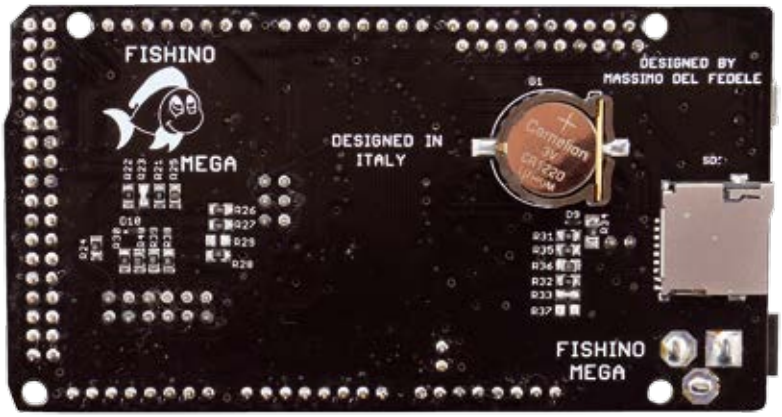
- compatibile al 100% con Arduino MEGA;
- modulo WiFi a bordo, con dotazione completa di librerie da noi sviluppate;
- lettore di microSD a bordo;
- modulo RTC a bordo;
- alimentazione switching integrale, in grado di fornire 800 mA-1A circa (complessivi tra i 3 ed i 5 volt), a partire da una tensione in ingresso che può variare da 3 e 20 volt;
- connettore per batteria LiPo a cella singola e caricabatteria a bordo;
- connettore laterale aggiuntivo per risolvere il noto bug di disallineamento della serie Arduino, che ne rende impossibile l'utilizzo con schede millefori e breadboard.

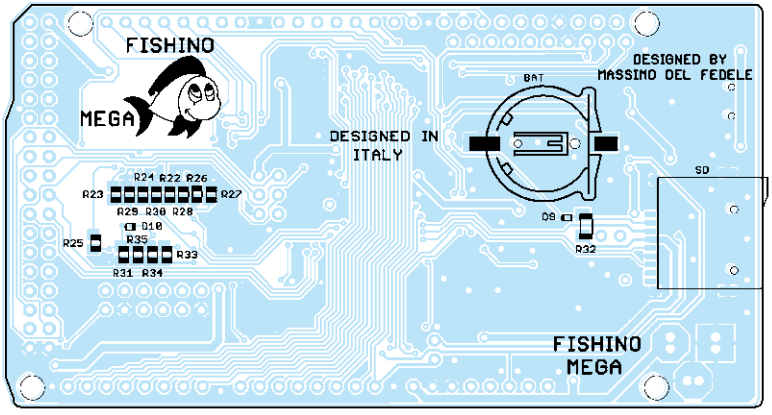
SCHEMA ELETTRICO

Il "pezzo forte" di questa nuova scheda è la sezione di alimentazione, che ha richiesto un lungo tempo di sviluppo ed ha anche portato alla realizzazione di un alimentatore multiuso separato la cui descrizione è iniziata in Marzo e si conclude in questo stesso fascicolo: il Torpedo. Per una descrizione dettagliata del convertitore SEPIC utilizzato rimandiamo quindi al relativo articolo, mentre qui ci limiteremo a farne una descrizione sommaria e ad analizzare la circuiteria di contorno in dettaglio.
Iniziamo dagli ingressi di alimentazione, che portano tre tensioni:

- VLiPo, che fa capo al connettore per la batteria ricaricabile ai polimeri di litio (LiPo);
- VUSB, che fa capo al connettore microUSB;
- VIN, che fa capo al plug di alimentazione PWRIN.

Le due tensioni solitamente più elevate, ossia VUSB e VIN (in realtà al connettore PWRIN possiamo applicare una tensione minore, fino a 3,2 volt, sebbene sia sconsigliabile perché comporta un calo di efficienza), vengono fatte passare attraverso i due diodi Schottky di potenza D6 e D7, mentre la tensione "più bassa", per motivi di efficienza, viene instradata verso il MOSFET a canale P siglato Q2.
Tralasciamo per un istante il funzionamento di Q2, sul quale ritorneremo a breve, e consideriamo solo il diodo interno al medesimo; i tre diodi (D6, D7 ed il diodo interno al MOSFET) realizzano una porta OR di potenza, la quale permette di ottenere in uscita la tensione maggiore tra quelle presenti agli ingressi; supponiamo ad esempio di avere solo la tensione VLiPo, di circa 3,7V: in questo caso il diodo interno al MOSFET risulta polarizzato direttamente portando verso il convertitore la medesima tensione, che non può ritornare verso gli altri due ingressi a causa della presenza dei diodi D6 e D7 polarizzati inversamente.
Ora, se colleghiamo il connettore microUSB, la tensione VUSB di 5 volt polarizza direttamente il diodo D6 e raggiunge il convertitore mentre il diodo interno al MOSFET si trova con una tensione di 4 volt all'anodo e di 5 volt al catodo e risulta quindi polarizzato inversamente, interdicendosi.
Se, infine, alimentiamo anche l'ingresso PWRIN con una





Elenco Componenti:

R1: 1 kohm (0805)
R2: 10 kohm (0805)
R3, R4: 1 kohm (0805)
R5: 10 kohm (0805)
R6: 470 ohm (0805)
R7: 2,4 kohm (0805)
R8: 1 kohm (0805)
R9: 10 kohm (0805)
R10: 2,7 kohm (0805)
R11: 2,2 kohm (0805)
R12: 220 kohm (0805)
R13: 97,6 kohm (0805)
R14: 13,3 kohm (0805)
R15: 470 ohm (0805)
R16, R17: 10 kohm (0805)
R17: 10 kohm (0805)
R18: 475 kohm (0805)
R19: 105 kohm (0805)
R20: 1 kohm (0805)
R21: 220 kohm (0805)
R22÷R25: 1 kohm (0805)
R26: 3,3 kohm (0805)
R27: 10 kohm (0805)
R28÷R31: 3,3 kohm (0805)
R32: 10 kohm (1206)
R33: 10 kohm (0805)
R34: 10 kohm (0805)
R35: 10 kohm (0805)
C1: 22 pF ceramico (0805)
C2: 22 pF ceramico (0805)
C3: 1 µF ceramico (0805)
C4: 100 nF ceramico (0805)
C5: 1 µF ceramico (0805)
C6: 4,7 µF ceramico (0805)
C7: 100 nF ceramico (0805)
C8: 22 pF ceramico (0805)
C9: 22 pF ceramico (0805)
C10: 100 nF ceramico (0805)
C11: 100 nF ceramico (0805)
C12: 22 µF 25 VL tantalio (0805)
C13: 4,7 µF 25 VL tantalio (0805)
C14: 22 µF ceramico

tensione superiore ai 5 volt, ad esempio di 15 volt, il diodo D7 risulterà polarizzato direttamente, portando la tensione VIN al convertitore, mentre i diodi D6 e quello interno al MOSFET verranno polarizzati inversamente interdicendosi.
Abbiamo quindi ottenuto lo scopo prefissato, ovvero una commutazione totalmente automatica della sorgente di alimentazione.
Torniamo ora al MOSFET inserito sulla linea positiva del connettore LiPo; apparentemente si tratta di un componente superfluo, essendo sufficiente il suo diodo interno per realizzare la commutazione; purtroppo i diodi, anche gli Schottky, hanno una caduta di tensione ai loro capi, che, per quanto piccola sia, causa perdite di potenza tali da risultare rilevanti ad alte correnti.
Facciamo un esempio pratico: supponiamo che l'utilizzatore (dopo il convertitore di cui parleremo in seguito) assorba 800 mA a 5 volt di tensione e che il convertitore abbia un'ottima efficienza (il 90% circa) cosa vicina a quella reale del nostro circuito. Ipotizziamo altresì che i diodi Schottky presentino una caduta di tensione di circa 0,5 volt.
Alimentando il tutto attraverso il plug PWRIN con una tensione di 15 volt, otterremo un assorbimento di corrente pari a:

$$I_{IN} = 800\,mA \cdot \frac{5\,V}{90\% \cdot (15\,V - 0.5\,V)} = 306.5\,mA$$

Da subito si nota il grosso vantaggio di utilizzare un alimentatore switching: la corrente assorbita diminuisce con l'aumentare della tensione in ingresso, a differenza dei regolatori lineari utilizzati sulla serie Arduino, nei quali la corrente assorbita resta identica a quella fornita e, moltiplicata per la caduta di tensione entrata-uscita, causa una rilevante dissipazione termica da parte del regolatore.
Nella formula, la caduta di tensione del diodo è stata considerata (0,5 volt), cosa che porta l'efficienza complessiva a:

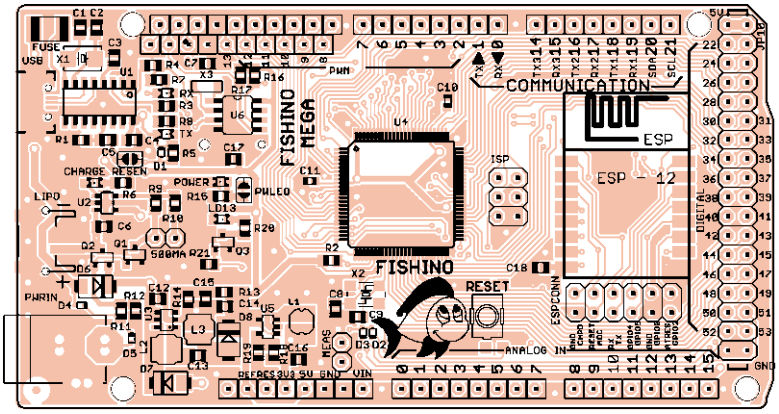
$$E = \frac{V_{OUT} \cdot I_{OUT}}{V_{IN} \cdot I_{IN}} = \frac{5V \cdot 800\,mA}{15V \cdot 306.5\,mA} = 87\%$$

contro il 90% teorico dato dal convertitore. Il diodo causa quindi una perdita di efficienza di un 3%, che possiamo considerare trascurabile. Dalla formula si nota

(0805)
C15: 100 pF ceramico (0805)
C16: 10 µF ceramico (0805)
C17: 100 nF ceramico (0805)
C18: 1 µF ceramico (0805)
CHARGE: LED rosso (0805)
WHITE: LED bianco (0805)
RX: LED giallo (0805)
TX: LED blu (0805)
POWER: LED verde (0805)
D1÷D5: RB521S-30TE61
D6÷D8: SS34
D9: RB521S-30TE61
D10: RB521S-30TE61
Q1: 2N7002
Q2: NTR4171P
Q3: 2N7002
U1: CH340G
U2: MCP73831T-2
U3: SX1308
U4: ATMEGA2560-16AU (MF1257)
U5: LC3406
U6: DS1307Z+

ESP: Modulo ESP8266
X1: Quarzo 12 MHz
X2: Quarzo 16 MHz
X3: Quarzo 32.768 kHz
L1: 2,2 µH
L2: 6,8 µH
L3: 6,8 µH
BAT: Porta Batteria CR1220 da CS
FUSE: 500mA (1210)
RESET: Microswitch
SD: Connettore micro-SD
USB: Connettore micro-USB

Varie:
- Plug alimentazione
- Connettore JST 2 vie passo 2 mm
- Strip maschio 2x3 vie
- Strip maschio 2 vie (2 pz.)
- Strip femmina 8 vie (5 pz.)
- Strip femmina 10 vie (2 pz.)
- Strip femmina 2x18 vie
- Batteria CR1220
- Circuito stampato S1257



che l'incidenza della caduta sul diodo aumenta col diminuire della tensione in ingresso, il che è logico perché lo switching trasferisce la potenza convertendone tensione e corrente: crescendo la tensione cala la corrente e viceversa.
Facciamo ora lo stesso calcolo utilizzando una tensione in ingresso di 3,6 volt (un valore prossimo a quello della batteria LiPo):

$$I_{LiPo} = 800\,mA \cdot \frac{5\,V}{90\% \cdot (3.6\,V - 0.5\,V)} = 1433.7\,mA$$

Vediamo innanzitutto che per poter ottenere 800 mA in uscita dobbiamo prelevare dall'ingresso ben 1.433,7 mA; l'efficienza reale diventa quindi:

$$E = \frac{V_{OUT} \cdot I_{OUT}}{V_{LiPo} \cdot I_{LiPo}} = \frac{5\,V \cdot 800\,mA}{3.6\,V \cdot 1433.7\,mA} = 77.5\%$$

In questo caso la caduta di tensione sul diodo ha portato ad una perdita di efficienza di ben il 12,5%, proprio dove servirebbe invece un'efficienza maggiore, visto che alimentiamo la scheda a batteria.
Oltretutto, una caduta di 0,5 volt sul diodo con una corrente di 1,4 ampere corrisponde ad una potenza dissipata di 0,7 W, che su un componente di piccole dimensioni porta ad un notevole incremento di temperatura.
Il MOSFET, per contro, non ha una caduta fissa ai suoi capi ma presenta, in conduzione, una resistenza che dipende dal componente utilizzato; nel nostro caso vale al massimo 100 milliohm.
Il calcolo si complica un pochino:

$$I_{LiPo} \cdot (V_{LiPo} - I_{LiPo} \cdot R_{DSON}) \cdot 90\% = V_O \cdot I_O$$

che è un'espressione di secondo grado in ILiPo; inserendo i valori e risolvendola, si ottiene:

$$I_{LiPo} = 1280\ mA$$

Una corrente ben inferiore ai 1.433,7 mA del caso precedente. In tali condizioni l'efficienza diventa:

$$E = \frac{V_{OUT} \cdot I_{OUT}}{V_{LiPo} \cdot I_{LiPo}} = \frac{5\,V \cdot 800\,mA}{3.6\,V \cdot 1280\,mA} = 86.8\%$$

Quindi il MOSFET causa una perdita di solo il 3,2% contro il 12% del diodo.
Stabilita l'utilità del MOSFET, resta da trovare il modo per portarlo in conduzione quando serve. Essendo un canale P, per ottenere ciò occorre polarizzarne

il gate con una tensione negativa rispetto al source; trattandosi di un MOSFET a bassa tensione di gate, per ottenere la resistenza di conduzione richiesta (RdsON inferiore ai 100 mΩ) sono sufficienti circa 2,5 volt. A questo provvede la resistenza R12, che polarizza negativamente il gate. Per contro i diodi Shottky per piccoli segnali D4 e D5, uniti alla resistenza R11 (quest'ultima per evitare che sul gate arrivi una tensione superiore a quella ammissibile, fungendo da partitore insieme alla R12) si occupano di portare all'interdizione il MOSFET quando è presente una tensione agli ingressi USB e/o PWRIN.
La tensione prescelta raggiunge i piedini 4 e 5 dell'U3, che è la base del convertitore SEPIC. Il convertitore è stato ampiamente descritto nei relativi articoli, quindi ne daremo solo un breve cenno; il vantaggio di questo schema è principalmente la possibilità di ottenere in uscita una tensione sia inferiore che superiore a quella in ingresso, unendo così i vantaggi di un convertitore Buck e di un Boost, ma utilizzando un solo integrato switching.
La tensione in uscita è regolata dal partitore costituito da R13 ed R14, che forniscono una tensione di riferimento di 0,6 volt all'ingresso FB dell'integrato quando in uscita sono presenti 5 volt.
All'uscita del convertitore SEPIC troviamo quindi una tensione continua di 5 volt. Il LED verde POWER funge da spia di accensione ed è disattivabile tagliando il ponticello SMD PWLED, in modo da ridurre i consumi all'osso se si vuole alimentare il Fishino MEGA a batteria, cosa che ci era stata richiesta dagli utenti di Fishino UNO.
La tensione di 5 volt in uscita dal SEPIC entra in un ulteriore convertitore switching, questa volta un semplice Buck (o step-down, abbassatore) converter che, per motivi di efficienza, è stato realizzato tramite un convertitore sincrono, ovvero dotato all'interno di un secondo MOSFET al posto del diodo Schottky utilizzato normalmente. Da questo convertitore escono i 3,3 volt necessari ad alimentare il modulo WiFi ed il lettore di microSD; come anticipato, la corrente disponibile si aggira sugli 800 mA - 1 ampere complessivi tra i 5 volt ed i 3 volt, cosa che permette di alimentare parecchi moduli esterni senza che l'alimentazione vada in crisi come succede spesso con Arduino ed i suoi regolatori lineari.
Una piccola parentesi: potrete notare la dimensione estremamente ridotta delle tre induttanze e dei condensatori di filtro dell'alimentazione; questo è stato reso possibile da un lato dall'elevata frequenza di lavoro degli integrati switching (sopra il MHz) e dall'altro dalla disponibilità di condensatori ceramici ad alta capacità.

Stadio carica batteria

Il cuore di questo stadio è l'integrato U2, un ben noto MCP73831, che contiene all'interno tutti i circuiti necessari alla carica ed al mantenimento della stessa di una batteria ai polimeri di litio a cella singola. L'integrato viene alimentato tramite una tensione a 5 volt applicata al piedino VIN, mentre la batteria da caricare viene connessa al piedino VBAT disaccoppiandola con un condensatore ceramico da 4,7 µF, che garantisce la stabilità dell'integrato.
Il chip integra un sistema per rilevare la presenza della batteria, sistema a dire il vero piuttosto critico che a volte ne segnala la presenza anche quando non è connessa, in particolar modo a basse tensioni di alimentazione della Fishino MEGA, ma ciò non ne inficia il funzionamento; l'unico inconveniente è l'illuminazione del LED di carica in quei casi.
Il piedino PROG viene utilizzato per impostare la corrente di carica, che nel nostro circuito può essere scelta tra due valori: circa 100 mA con il ponticello siglato 500 MA aperto (viene inserita la sola resistenza R9 da 10 kohm) oppure circa 500 mA con il ponticello chiuso, il che corrisponde a inserire in parallelo alla R9 la R10 da 2,7 kohm. La formula per impostare la corrente di carica, nel caso si ritenesse necessario modificarla, è la seguente:

$$I_{reg} = \frac{1000}{R_{PROG}}$$

dove RPROG è la resistenza applicata al piedino PROG, in KOhm, ed IREG è la corrente di carica, in mA. Lo stesso piedino PROG se connesso al positivo dell'alimentazione o se lasciato fluttuante (disconnesso) serve a disabilitare la carica, disattivando i circuiti interni dell'integrato e riducendone il consumo praticamente a zero; questo viene utilizzato nel nostro schema inserendo il MOSFET Q1, un comune 2N7002 a canale N che, quando il gate risulta polarizzato negativamente, scollega di fatto le 2 resistenze di programmazione rendendo l'ingresso PROG fluttuante.
Il MOSFET è pilotato dallo stesso segnale utilizzato per pilotare l'interruttore della batteria Q2, anche se in modalità inversa : una tensione agli ingressi VUSB o VIN lo porta in conduzione attivando la carica, mentre in assenza (quando il circuito risulta alimentato dalla sola batteria) il MOSFET viene interdetto e la carica disattivata.
Per concludere la descrizione,

si noti il jumper siglato MEAS (measure, misura) che permette di collegare l'ingresso analogico ADC0 del controller alla batteria per eseguire il controllo dello stato della carica.

Interfaccia USB

L'interfaccia USB è la medesima utilizzata per il Fishino UNO, quindi ci limiteremo ad un piccolo accenno, rimandando al succitato articolo per una descrizione approfondita. Lo stadio gravita attorno all'ormai noto CH340G, un'alternativa estremamente valida al più noto FT232 o ad altre soluzioni con microcontrollori; il chip è stato scelto sia per motivi di economicità che di semplicità circuitale, a parità di prestazioni. L'integrato fornisce in uscita tutti i segnali di un'interfaccia RS232 standard, dei quali utilizziamo solo quelli di trasmissione/ricezione dati (Rx e Tx) ed il segnale DTR utilizzato per il reset automatico in fase di programmazione, come nell'Arduino originale, cosa che permette il caricamento degli sketch senza dover premere pulsanti o azionare interruttori. Torneremo in seguito sulla circuiteria di Reset perchè rispetto all'originale è stata modificata in modo da permettere in futuro la riprogrammazione dell'ATmega via WiFi.

ATmega 2560

In questa sezione lo schema di Fishino MEGA non si discosta da quello di Arduino MEGA; il controller è il medesimo, corredato dall'usuale quarzo a 16 MHz, dai 2 condensatori sul circuito oscillante e da un certo numero di condensatori di disaccoppiamento sulle linee di alimentazione, necessari per evitare che disturbi sulle linee di alimentazione possano influenzare il funzionamento del chip. Una nota sui connettori di I/O : come si può notare dalle immagini è stato aggiunto un piccolo connettore a 10 pins affiancato a quello standard ma leggermente sfalsato in modo da poter utilizzare una scheda preforata standard per gli shields, risolvendo quindi l'annoso problema del passo errato dei connettori di Arduino senza peraltro inficiarne la compatibilità con gli shields esistenti.

Interfaccia SPI
Adattatori di livello

Anche questa sezione risulta praticamente identica a quella già vista nel Fishino UNO; serve ad adattare i livelli delle logiche a 5 volt dell'ATMEGA con quelle a 3.3 volt del modulo WiFi e della scheda microSD. L'adattamento viene realizzato semplicemente tramite partitori resistivi (resistenze da R22 a R31) nella direzione 5V -> 3.3V, mentre nella direzione inversa viene sfruttato il fatto che le logiche a 5 volt accettano come segnali alti valori ben inferiori a 3 volt, risultando quindi compatibili con le logiche a 3.3 volt. Il segnale MISO (Master In Slave Out) in teoria non richiederebbe un adattamento, visto che la direzione va dalla logica a 3.3 volt verso quella a 5 volt; l'abbiamo inserito comunque in previsione della futura estensione del firmware del modulo WiFi che permetterà il caricamento degli sketch attraverso il medesimo, cosa che richiede uno scambio di ruoli, diventando in quel caso il modulo WiFi il master e l'ATmega lo slave.

Interfaccia scheda MicroSD

L'interfaccia è la medesima che abbiamo descritto nel Fishino UNO, e rispecchia lo shield SD (o analoghi combinati); funziona attraverso le linee SPI tra cui

Librerie disponibili

Attualmente sono state realizzate due librerie per la gestione del Fishino, di seguito descritte.

- 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), FishinoSecureClient (non presente nelle librerie originali, permette connessioni a server sicuri SSL/HTTPS) 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.
- 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; è già stata recentemente implementata la gestione degli I/O digitali aggiuntivi presenti sul connettore ESPCONN, mentre è prevista a breve la gestione della seriale, dell'input analogico e delle uscite PWM sempre sul connettore ESPCONN.

Nel pacchetto di download sono presenti altre librerie, utilizzate per lo più nelle demo che sono state sviluppate e/o verranno sviluppate in seguito; consigliamo di installare tutto il contenuto nell'apposita cartella di Arduino.

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 (R23) verso il positivo e un diodo (D9), che permette il solo passaggio delle sole correnti negative. Lo schema scelto consente di avere la scheda in standby quando il segnale SDCS (connesso al pin digitale 4 del Fishino MEGA) 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.

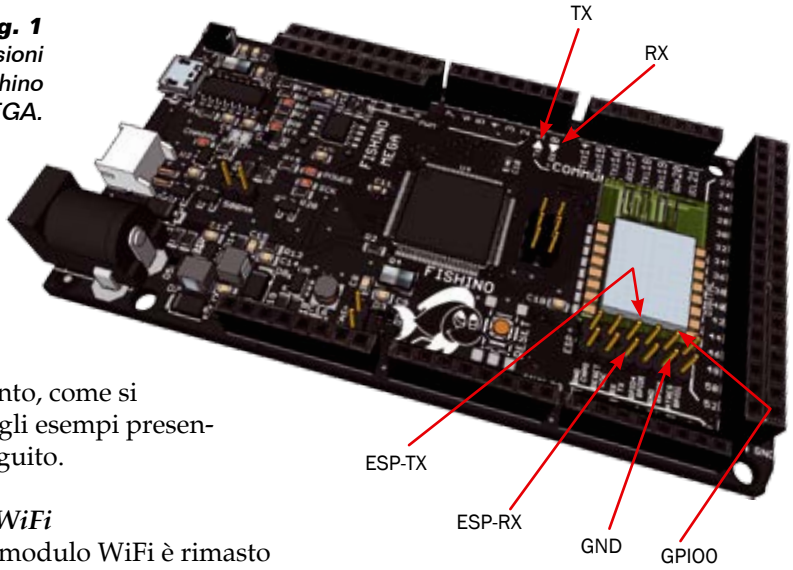


***Fig. 1** Connessioni della Fishino MEGA.*

Sezione WiFi

Anche il modulo WiFi è rimasto invariato rispetto a quello della scheda Fishino UNO; rimandiamo perciò all'articolo, pubblicato nel fascicolo n° 199, dove è stato descritto in maniera esauriente. Riportiamo quindi solo alcune note importanti, ovvero il tipo di comunicazione con l'atmega, realizzata tramite interfaccia SPI grazie ad un firmware da noi appositamente sviluppato, ed alcuni accorgimenti circuitali quali il diodo D10 utilizzato per forzare a livello basso il pin GPIO15 del modulo al reset, senza il quale il modulo stesso si avvierebbe nella modalità "caricamento da SD" che lo renderebbe inutilizzabile. Questo risulta necessario poichè la linea GPIO15 ha anche funzione di Slave Select (SS) del modulo e non può quindi essere collegata direttamente a massa. Tutti i pin utili del modulo sono portati su un connettore (ESPCONN) come riportato qui di seguito.

- GPIO0: oltre ad essere utilizzabile come input/output digitale, serve per selezionare la modalità d'avvio al boot del modulo. Quest'ultimo può infatti essere avviato da Flash interna (funzionamento normale, GPIO0 a 1) o da interfaccia seriale, utilizzato per la riprogrammazione del firmware (GPIO0 a 0).
- GPIO2, GPIO4 e GPIO5 sono disponibili per l'uso come pin digitali, e sono sfruttabili tramite le apposite funzioni di libreria come fossero estensioni dei pin digitali di Arduino. Rx e Tx costituiscono la porta seriale hardware del modulo e sono utilizzati anche in fase di programmazione del firmware. Una prossima estensione del firmware ne permetterà l'uso come porta seriale aggiunti-

Silica apre la strada all'**IoT**

Il nome è già un programma: Visible Things, ossia "cose visibili" (ad evidenziare la possibilità di creare applicazioni concrete) è la piattaforma per lo sviluppo di sistemi e applicazioni IoT di Silica (www.avnetmemec-silica.com), che offre risorse hardware e software integrate per collegare sensori intelligenti e dispositivi embedded direttamente alle applicazioni cloud tramite funzioni di connettività a corto raggio verso unità gateway e connessioni WiFi, 3G e 4G. Supporta anche le reti IoT in tecnologia SIGFOX e LoRaWAN. Il produttore offre già tre starter kit basati su microcontrollori ARM Cortex.

va che consentirà a Fishino MEGA di avere un'ulteriore porta seriale.
- CH_PD è il pin di abilitazione del modulo. Portandolo a livello alto il modulo risulta abilitato (impostazione predefinita), mentre un livello basso mette in standby 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 (1.024 valori possibili.

Circuiteria di RESET

La sezione di reset della Fishino MEGA è uguale a quella della Fishino UNO che già conoscerete; come accennato in precedenza, risulta più complicata di quella dell'Arduino MEGA 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 autoresettarsi.

Iniziamo dal segnale DTR che esce dall'interfaccia USB/Seriale (U1/CH340G); questo, come anticipato, viene posto a livello basso quando la porta seriale viene aperta. Attraverso il condensatore C5 (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 RESEN (tagliando il quale è possibile disattivare l'autoreset), 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, nelle schede Fishino è inserito un diodo (D2) 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.

In sintesi:
- premendo il pulsante RESET, o connettendo la seriale, l'impulso di reset raggiunge sia l'Atmega (attraverso D2) 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 D2, 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 e 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

Fishino MEGA utilizza un convertitore USB/Seriale del tipo

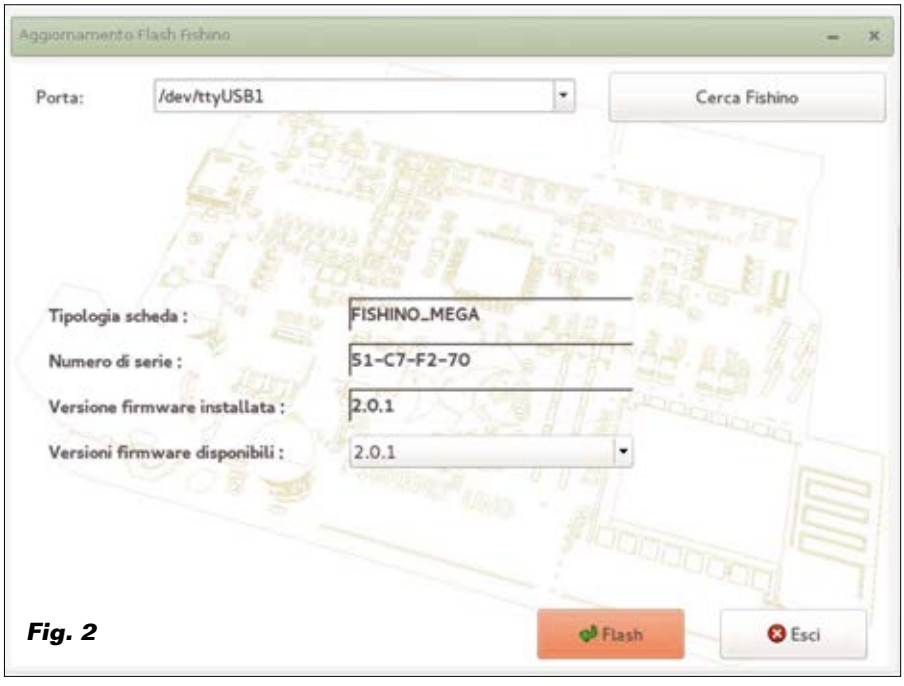
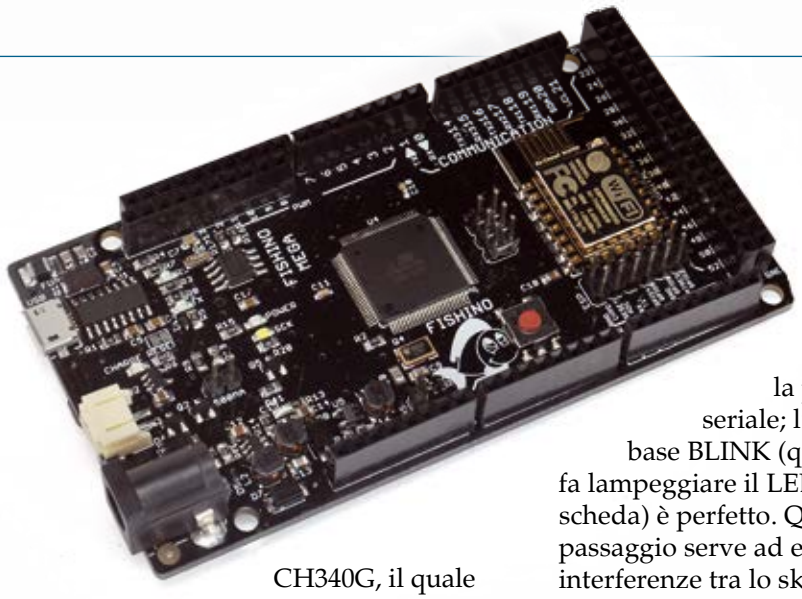


Fig. 2



CH340G, il quale necessita di driver appositi, almeno per Windows (fino alla versione 7 inclusa, pare che dalla 8 in poi i driver siano già presenti) e per Mac. Per l'ambiente Linux, per contro, i driver non sono necessari essendo questi già presenti nel kernel. I driver per Windows e per Mac si possono scaricare dal sito *www.fishino.it* (o *www.fishino.com* per la versione internazionale in inglese), sezione Download.

AGGIORNAMENTO FIRMWARE DEL MODULO WIFI

Fishino MEGA viene fornito con la versione del firmware disponibile al momento dell'assemblaggio. Essendo questo in fase di continuo sviluppo, conviene sicuramente eseguire un aggiornamento immediato ed è consigliato ripeterlo periodicamente. 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 la piattaforma Windows che Linux, che esegue l'operazione in modo completamente automatico ed a prova di errore. È prevista in un prossimo futuro anche una versione del flasher per Mac. I passi 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. Questo passaggio serve ad evitare interferenze tra lo sketch caricato ed il collegamento seriale tramite l'Atmega e l'ESP. Se il programma di flashing non rileva il Fishino, al 99% il problema è uno sketch sbagliato caricato.

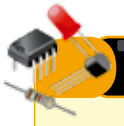
2) Connettere la porta TX di Fishino con la porta ESP-TX sul connettore ESPCONN, e la porta RX di Fishino con la porta ESP-RX sul connettore ESPCONN (**Fig. 1**).

- Connettere la porta GPIO0 a massa tramite un cavetto o un ponticello sempre sul connettore ESPCONN (**Fig. 1**).
- Collegare il Fishino al PC (o premere il pulsante di RESET se già connesso).
- 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 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 (**Fig. 2**). 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 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.

CONCLUSIONI

Terminiamo qui l'esposizione della seconda scheda della serie Fishino, preannunciandovi l'imminente pubblicazione della prossima board, che sarà la Fishino Guppy; questa board, compatibile con la Arduino Nano, si distinguerà da essa perché dotata di WiFi, microSD e l'alimentazione switching con batteria che contraddistinguono il nostro "pesciolino" tecnologico.



per il MATERIALE

La board Fishino MEGA (cod. FISHINOMEGA) viene fornita montata e collaudata. Può essere acquistata presso Futura Elettronica al prezzo di Euro 49,90. 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