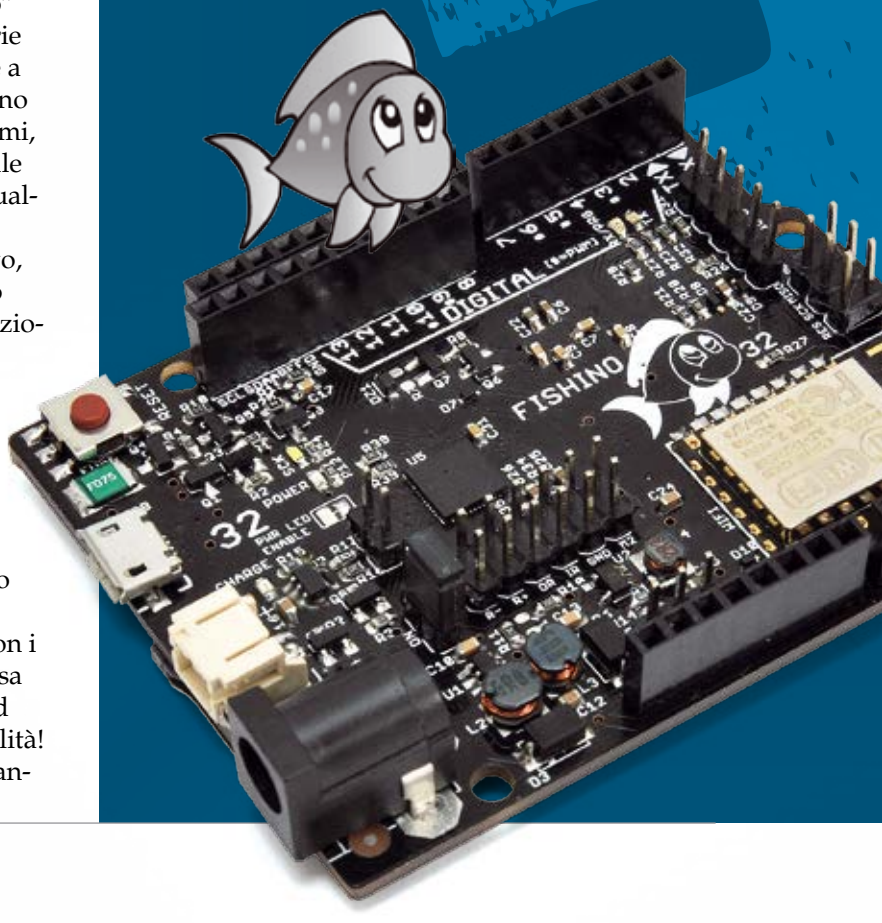
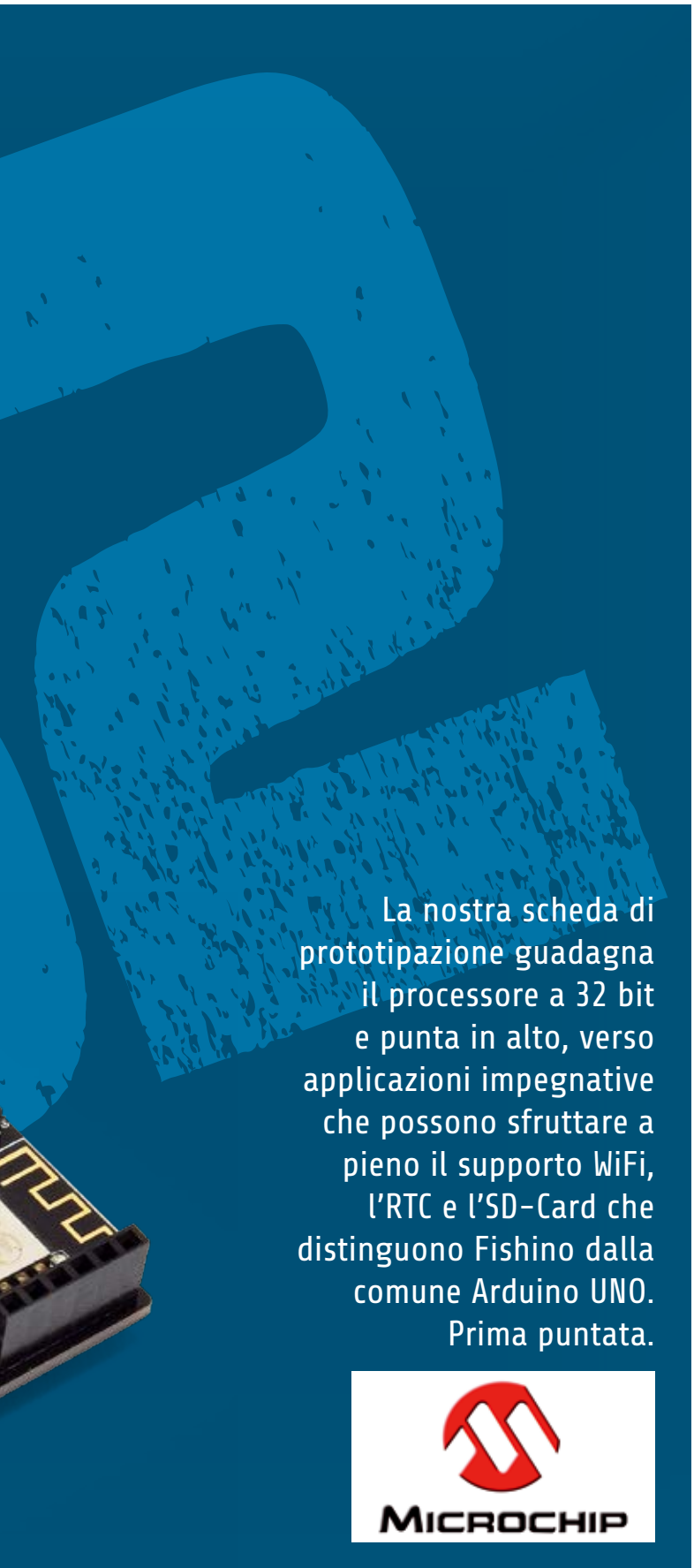
FISHINO A 32 BIT!

di MASSIMO DEL FEDELE

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. Si, 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-

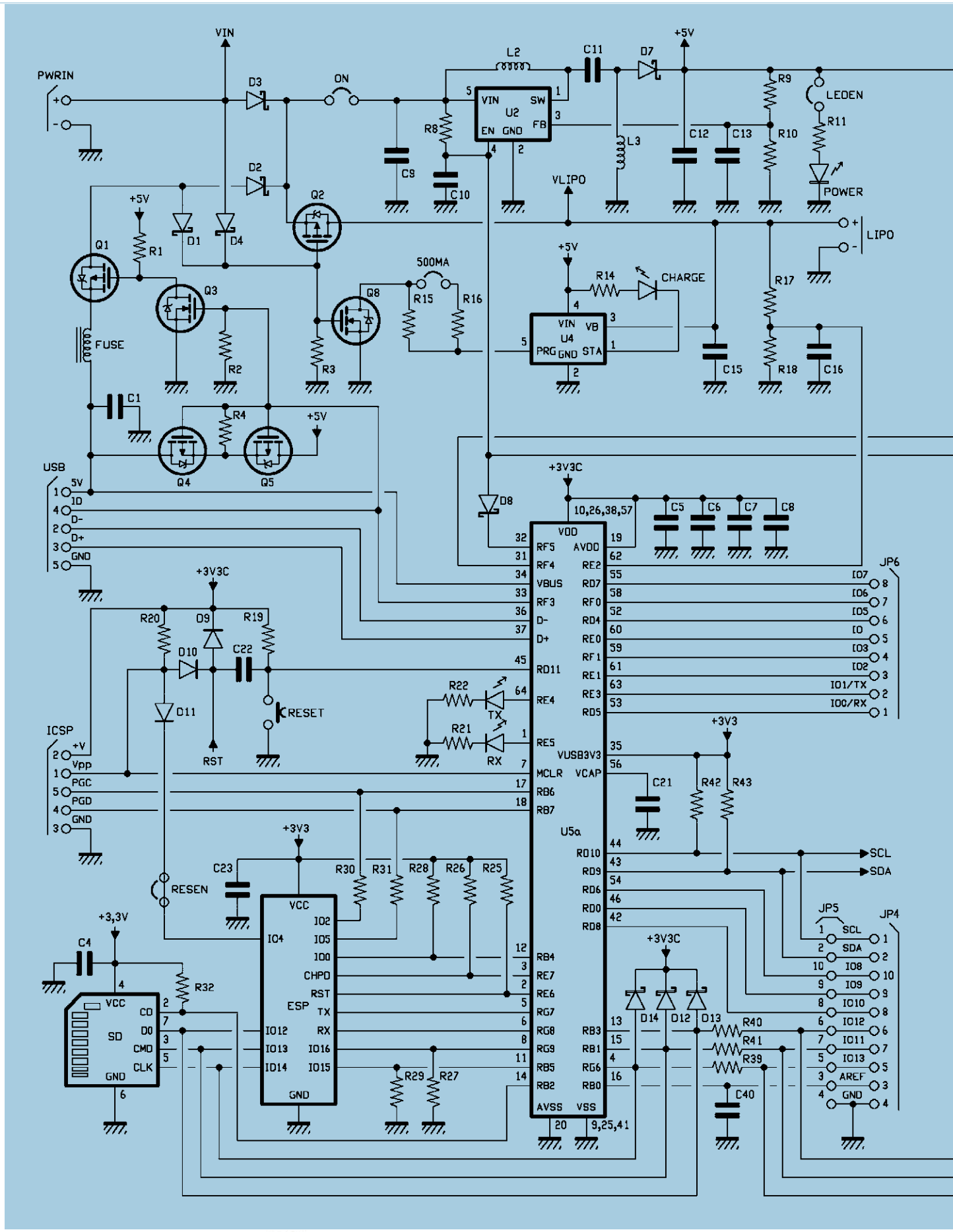


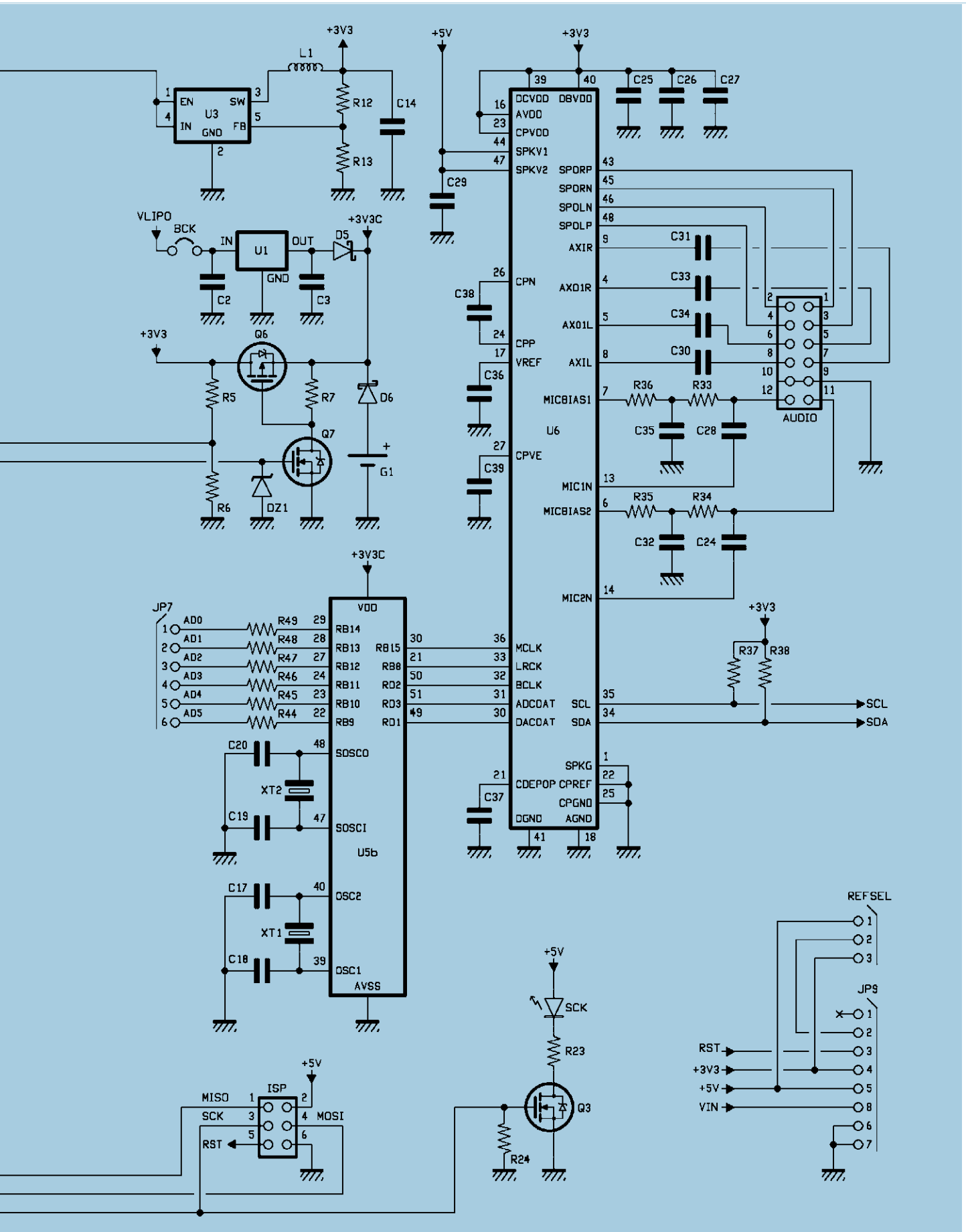


La nostra scheda di prototipazione guadagna il processore a 32 bit e punta in alto, verso applicazioni impegnative che possono sfruttare a pieno il supporto WiFi, l'RTC e l'SD-Card che distinguono Fishino dalla comune Arduino UNO. Prima puntata.

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 microfonici, 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;


VIN
PWRIN
D3
ON
L2
C11
D7
+5V
U2
VIN SW
EN GND FB
R8
C9
C10
L3
C12
C13
R9
R10
LEDEN
R11
POWER
VLIPO
LIPO
D2
Q2
+5V
D1
D4
Q1
R1
Q3
500MA
R15
R16
Q8
R14
CHARGE
R17
U4
VIN VB
PRG GND STA
C15
R18
C16
FUSE
R2
R3
C1
R4
+5V
Q4
Q5
USB
5V
ID
D-
D+
GND
+3V3C
D8
10,26,38,57
C5
C6
C7
C8
VDD
RF5 AVDD
RF4 RE2
VBUS RD7
RF3 RF0
D- RD4
D+ RE0
RD11 RF1
RE4 RE1
RE5 RE3
RD5
VUSB3V3
MCLR VCAP
RB6
RB7
U5a
RD10
RD9
RD6
RD0
RD8
RB4
RE7
RE6
RG7
RG8
RG9
RB5
RB2
RB3
RB1
RG6
RB0
AVSS VSS
20
9,25,41
JP6
IO7
IO6
IO5
IO
IO3
IO2
IO1/TX
IO0/RX
+3V3C
R20
D9
R19
D10
C22
D11
RESET
RST
R22
TX
R21
RX
+3V3
C21
R42
R43
SCL
SDA
ICSP
+V
Vpp
PGC
PGD
GND
RESEN
+3V3
C23
R30
R31
R28
R26
R25
VCC
IO2
IO4
IO5
IO0
CHPD
RST
ESP
TX
IO12
RX
IO13
IO16
IO14
IO15
GND
R29
R27
+3,3V
C4
R32
VCC
CO
SD
DO
CMD
CLK
GND
+3V3C
D14
D12
D13
R40
R41
R39
C40
JP5
JP4
SCL
SDA
IO8
IO9
IO10
IO12
IO11
IO13
AREF
GND


+3V3
L1
U3
EN
SW
IN
GND
FB
R12
R13
C14
VLIPO
BCK
U1
IN
OUT
GND
D5
C2
C3
+3V3C
Q6
R5
R7
D6
Q7
R6
DZ1
G1
+5V
C29
+3V3
C25
C26
C27
DCVDD
DBVDD
AVDD
CPVDD
SPKV1
SPKV2
SPORP
SPORN
SPOLN
SPOLP
AXIR
AXO1R
AXO1L
AXIL
CPN
CPP
VREF
CPVE
MICBIAS1
MICBIAS2
MIC1N
MIC2N
U6
C38
C36
C39
C31
C33
C34
C30
R36
R33
C35
C28
R35
R34
C32
C24
AUDIO
+3V3C
JP7
AD0
AD1
AD2
AD3
AD4
AD5
R49
R48
R47
R46
R45
R44
VDD
RB14
RB13
RB12
RB11
RB10
RB9
RB15
RB8
RD2
RD3
RD1
MCLK
LRCK
BCLK
ADCDAT
DACDAT
SCL
SDA
+3V3
R37
R38
SCL
SDA
C20
XT2
C19
SOSCO
SOSCI
U5b
C17
XT1
C18
OSC2
OSC1
AVSS
SPKG
CDEPOP
CPREF
CPGND
DGND
AGND
C37
REFSEL
JP9
+5V
SCK
R23
Q3
R24
RST
+3V3
+5V
VIN
ISP
MISO
SCK
RST
MOSI

- 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 polarizzato 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 ininfluente, 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

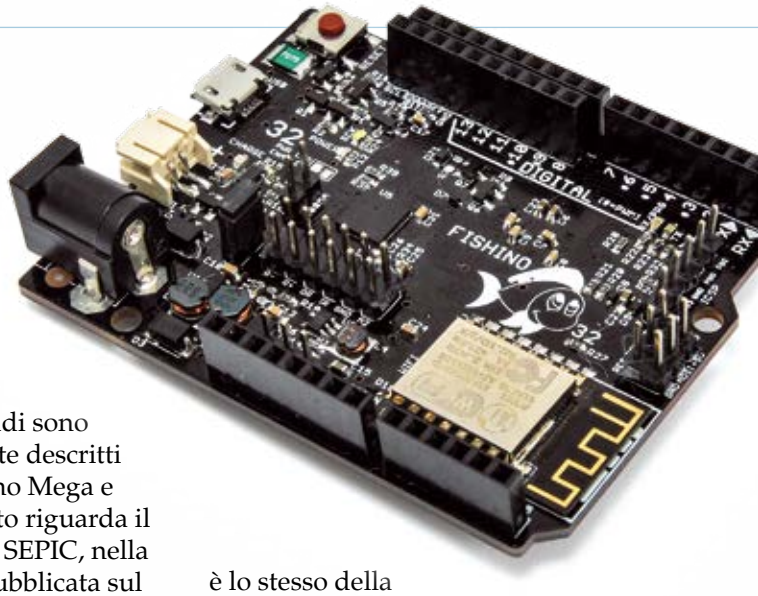
Il microcontrollore PIC utilizzato nella nostra board appartiene alla famiglia PIC32MX, basata sull'architettura MIPS, che si distingue subito dalla classe di microcontrollori che equipaggiano Arduino, i quali sono Atmel AVR, basati su un core ARM.
Lo schema a blocchi che trovate in questo riquadro mostra 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 perifefiche 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 periodimetro; 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.
Qusto 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, 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

La demo **fishino player**

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 irremediabilmente la scheda!

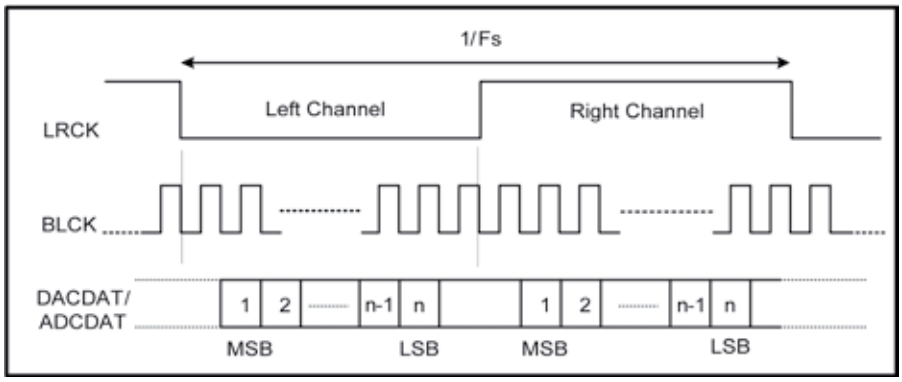


***Fig. 1** - Segnali dell'SPI correlati.*

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 dbA ed una distorsione totale a -90 db;
- un convertitore A/D (ADC) stereo a 24 bit con un rapporto segnale/disturbo di 93 dbA 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 trasmessi; 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.

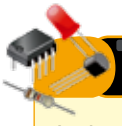
I segnali che transitano sono visibili nella **Fig. 1**, dove manca il clock di riferimento (MCLK) che è comunque un clock fisso di frequenza quadrupla rispetto al BLCK.
Lasciamo per il momento la descrizione del codec, che riprenderemo quando parleremo delle librerie software.

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 programmare 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