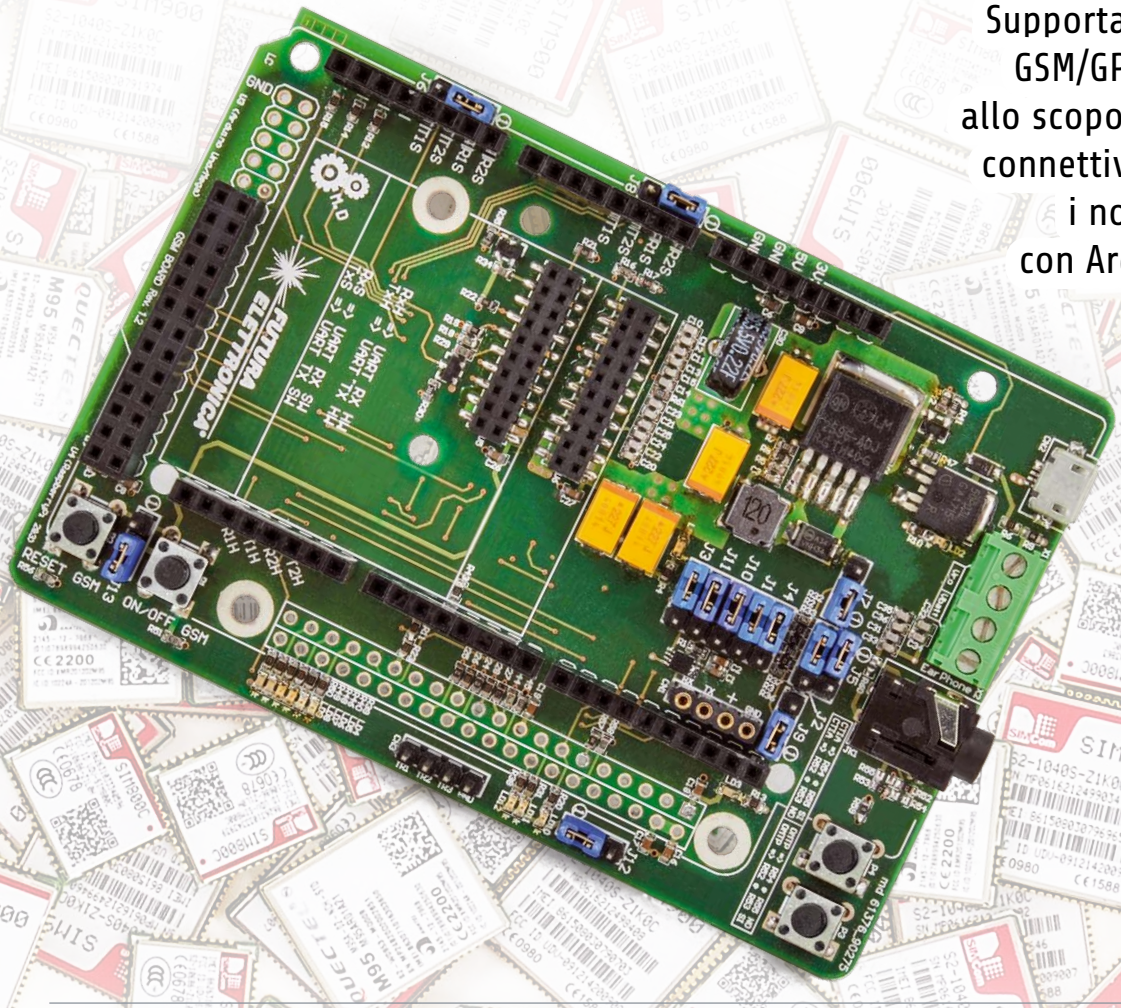
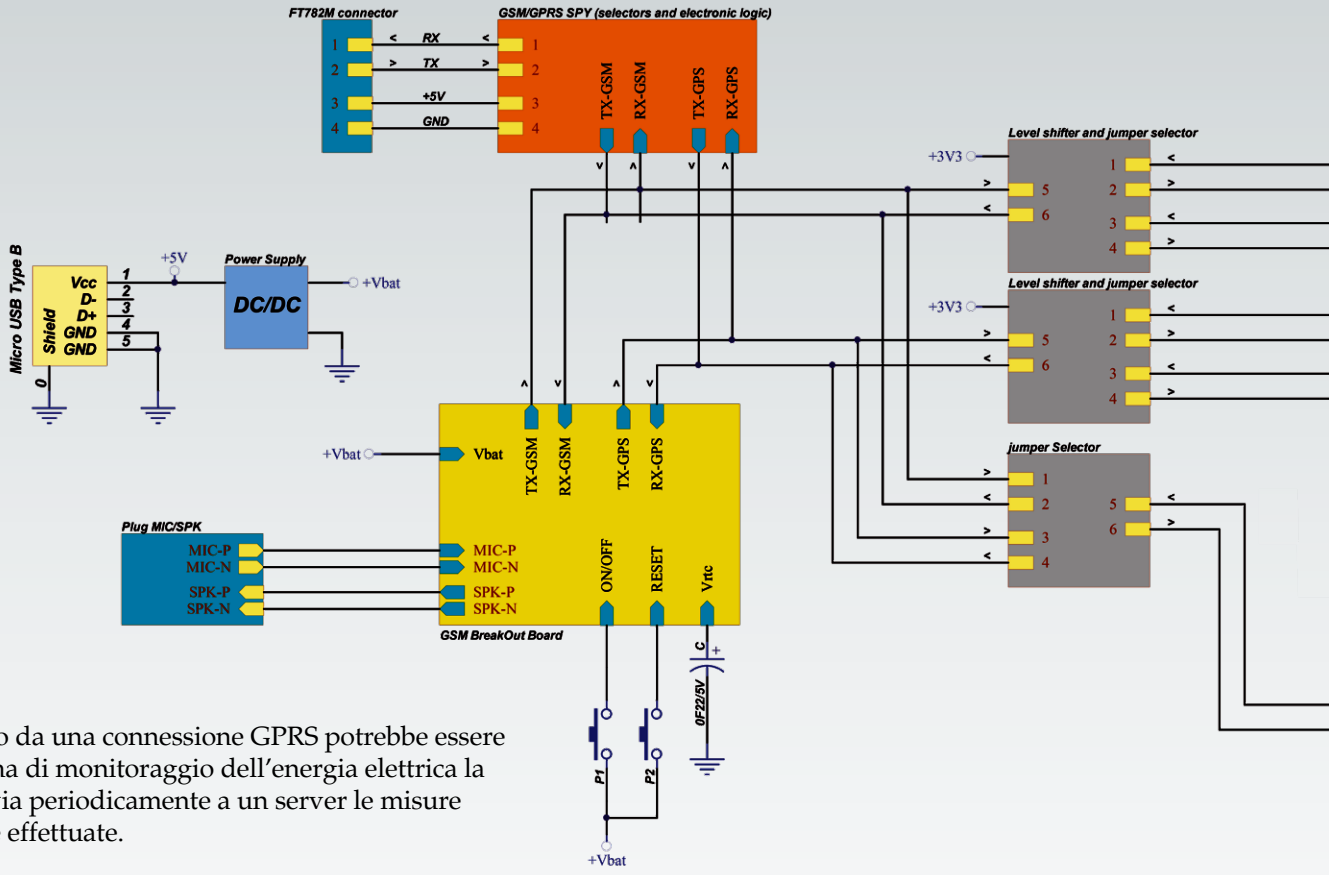
GSM SHIELD UNIVERSALE

di MATTEO DESTRO

Data la richiesta crescente di apparecchiature elettroniche con connessione alla rete GSM/GPRS, abbiamo deciso di sviluppare un nuovo shield GSM per piattaforma Arduino e Raspberry Pi 3, capace di supportare svariati moduli su breakout-board GSM sviluppate da Futura Elettronica, le quali utilizzano i motori GSM di SimCom, Quectel e Fibocom sulla medesima pin out. Quindi sarà possibile provare e trovare il modulo GSM più adatto alla propria applicazione, sviluppare il firmware sullo shield e poi integrare nel circuito definitivo il modulo scelto. Lo shield consente di sfruttare sia le funzioni di chiamata vocale e scambio di SMS, sia per la gestione delle connessioni dati GPRS. In quest'ultimo caso la SIM dovrà essere abilitata al traffico dati.
Le applicazioni che fanno uso dei semplici SMS possono essere sistemi che dato il verificarsi di un evento devono inviare un testo, con semplici dati correlati, verso un telefono cellulare.
La connessione dati è utile nelle applicazioni in cui bisogna inviare o ricevere dati da o verso un server. Mentre una tipica applicazione che potrebbe trarre

Supporta vari moduli GSM/GPRS low-cost allo scopo di dotare di connettività cellulare i nostri progetti con Arduino. Prima puntata.





vantaggio da una connessione GPRS potrebbe essere un sistema di monitoraggio dell'energia elettrica la quale invia periodicamente a un server le misure elettriche effettuate.

SCHEMA A BLOCCHI

Prima di analizzare lo schema elettrico soffermiamoci sulla sua composizione, descritta dallo schema a blocchi proposto in questa pagina. Iniziamo con l'alimentazione principale, che viene prelevata tramite un connettore **µUSB** al quale è possibile collegare un alimentatore come quello dei cellulari, oppure è possibile collegare il cavo USB direttamente a una presa USB tipo A del PC, prendendo l'alimentazione dallo stesso. I 5 volt presenti sul connettore µUSB vengono portati sia al DC/DC converter, per generare la tensione di alimentazione dei moduli GSM (+VBAT) sia agli header per le schede Arduino e Raspberry Pi.

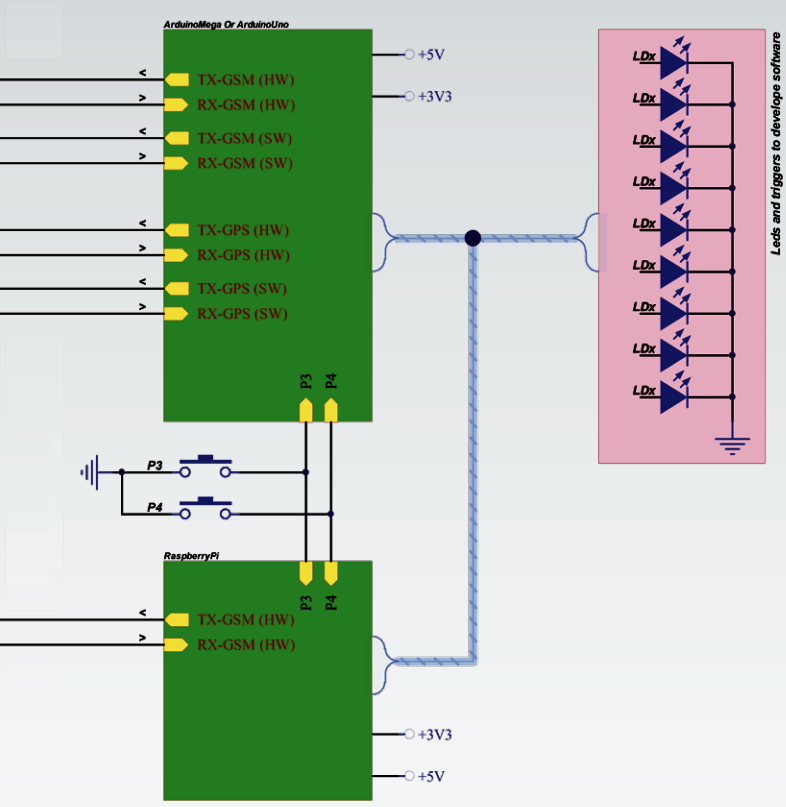
Nello schema a blocchi le schede Arduino e Raspberry Pi sono rappresentate da due blocchi funzionali di colore verde, dai quali partono alcune linee di comunicazione, di diagnostica e debug.

Ma procediamo con ordine: per prima cosa osserviamo che dal blocco funzionale corrispondente ad Arduino partono diverse linee di comunicazione UART, sia hardware sia assegnate da software; queste ultime sono I/O che tramite librerie possono fungere da seriali e sono utili, ad esempio su Arduino Uno R3, dove c'è un solo UART hardware. Se questo dev'essere usato per debuggare il firmware attraverso il monitor seriale, volendo comunicare con il modulo GSM occorre inizializzare una seriale virtuale. Qualcuno si starà chiedendo perché abbiamo predisposto due UART? La scelta nasce dal fatto che ci possono essere moduli GSM dotati di due UART: ad esempio il SimCom SIM928A, che integra un ricevitore GPS e dispone di una seriale per ricevere i comandi AT per la gestione della sezione GSM/GPRS e l'altra per la lettura dei dati GPS NMEA sul posizionamento.

Per quanto riguarda Raspberry Pi, questa mette a disposizione un solo UART hardware; in questo caso non è possibile avere due comunicazioni simultanee.

Le linee UART, nel caso delle schede Arduino, prima di giungere ai moduli GSM passano attraverso dei traslatori di livello che permettono di accoppiare linee dati con tensioni di lavoro a +5V con linee dati con tensioni di lavoro a +3,3V. Questo non è necessario per quanto riguarda le schede Raspberry Pi, in quanto queste lavorano già a tensione +3,3V.

Oltre ai traslatori di livello, le linee UART prima di giungere ai moduli GSM incontrano una serie di jumper a 3 posizioni utili per selezionare se utilizzare una comunicazioni di tipo hardware piuttosto che software e se si vuole attivare/disattivare la spia di comunicazione per vedere quali comandi si stanno scambiando le schede Arduino/Raspberry



Pi con i motori GSM. È anche possibile instradare i segnali in modo da inviare i comandi AT direttamente da PC, questo è molto utile durante la fase di studio dei comandi per verificarne il corretto funzionamento prima di implementarli in codice nelle schede Arduino e Raspberry Pi. Queste sezioni di level shifting e instradamento dei segnali sono rappresentate da dei blocchi di colore grigio (sono in tutto tre blocchi distinti, due nominati "***Level shifter and jumper selector***" mentre il terzo è solo un "***Jumper selector***").

La sezione dei motori GSM è rappresentata dal blocco di colore giallo "***GSM BreakOut Board***" al quale giungono le linee di comunicazione discusse in precedenza. A questo blocco, oltre alle linee dati, giunge anche l'alimentazione +VBAT generata dal DC/DC converter e l'alimentazione ***Vrtc*** resa disponibile dal condensatore a foglie d'oro da 0,22F usato per tenere alimentato il Real Time Clock Calendar presente sui moduli GSM in caso di mancata alimentazione primaria. La carica del condensatore viene garantita dalla stessa elettronica del moduli GSM, nel momento in cui questa viene a mancare il condensatore provvede a tenere attivo il RTCC fino a sua completa scarica.

Ai moduli GSM vengono collegati anche due pulsanti i quali servono per accendere/spegnere il motore GSM, pulsante P1, oppure per resettare il motore GSM, pulsante P2. Questi pulsanti vengono utili quando si deve agire manualmente sui motori, ad esempio accenderli per poi inviare dei comandi AT da PC. La durata dell'impulso di accensione/spegnimento, piuttosto che la durata dell'impulso di reset dipendono dal motore stesso. Quindi è necessario consultare i datasheet per i dettagli, ovviamente si parla sempre di impulsi di durata minima.

Concludiamo la carrellata con le linee MIC e SPK, linee differenziali, le quali servono per collegare rispettivamente un microfono e uno speaker utili durante le chiamate foniche.

Esaminiamo ora il blocco arancione "***GSM/GPRS SPY***" il quale come accennato prima serve per spiare la comunicazione seriale tra i motori GSM e le schede Arduino/Raspberry Pi, piuttosto che inviare direttamente i comandi AT da PC. Il blocco funzionale è molto semplice e comprende delle porte logiche e qualche jumper per l'instradamento dei segnali nonché un connettore 4 poli, blocco azzurro adiacente, al quale collegare il convertitore USB/UART FT782M della Futura Elettronica.

Concludiamo l'analisi dello schema a blocchi dicendo che alle schede Arduino/Raspberry Pi sono collegati due pulsanti P3 e P4 per sviluppi futuri e una serie di linee digitali per il debug del firmware che si andrà a scrivere. In particolare, abbiamo tre linee di trigger, con relativi LED, più sei LED di diagnostica: due rossi, due verdi e altrettanti gialli; sono utili durante le fasi di inizializzazione dei motori GSM, invio e ricezione SMS, chiamate vocali ecc.

Con questo abbiamo concluso l'analisi dello schema a blocchi. Possiamo quindi passare alla descrizione dettagliata dello schema elettrico.

SCHEMA ELETTRICO

Cominciamo dalla sezione di alimentazione, la quale attinge tensione tramite il connettore microUSB, dal quale i 5V raggiungono l'integrato U2 che è un regolatore di tensione step-down. In dettaglio abbiamo usato un LM2596, package D²PACK, con tensione di uscita regolabile da un minimo di +1,23V a un massimo di +37V.

Per la nostra applicazione abbiamo impostato la tensione di uscita a +4,1V in quanto i moduli GSM si aspettano un'alimentazione stabilizzata tra +3,2V a +4,8V. L'integrato LM2596 garantisce una corrente di uscita fino a 3A con una corrente di standby di 80µA, accetta tensioni di ingresso fino a +40V e la frequenza di lavoro della sezione switching è di 150kHz. La **Fig. 1** mostra lo schema a blocchi del

SCOPRI LA NUOVA RASPBERRY PI 3 A+

Utilizzala subito con lo starter Kit!

€ 59,90

Cod. RASPKITV8

Il kit comprende:

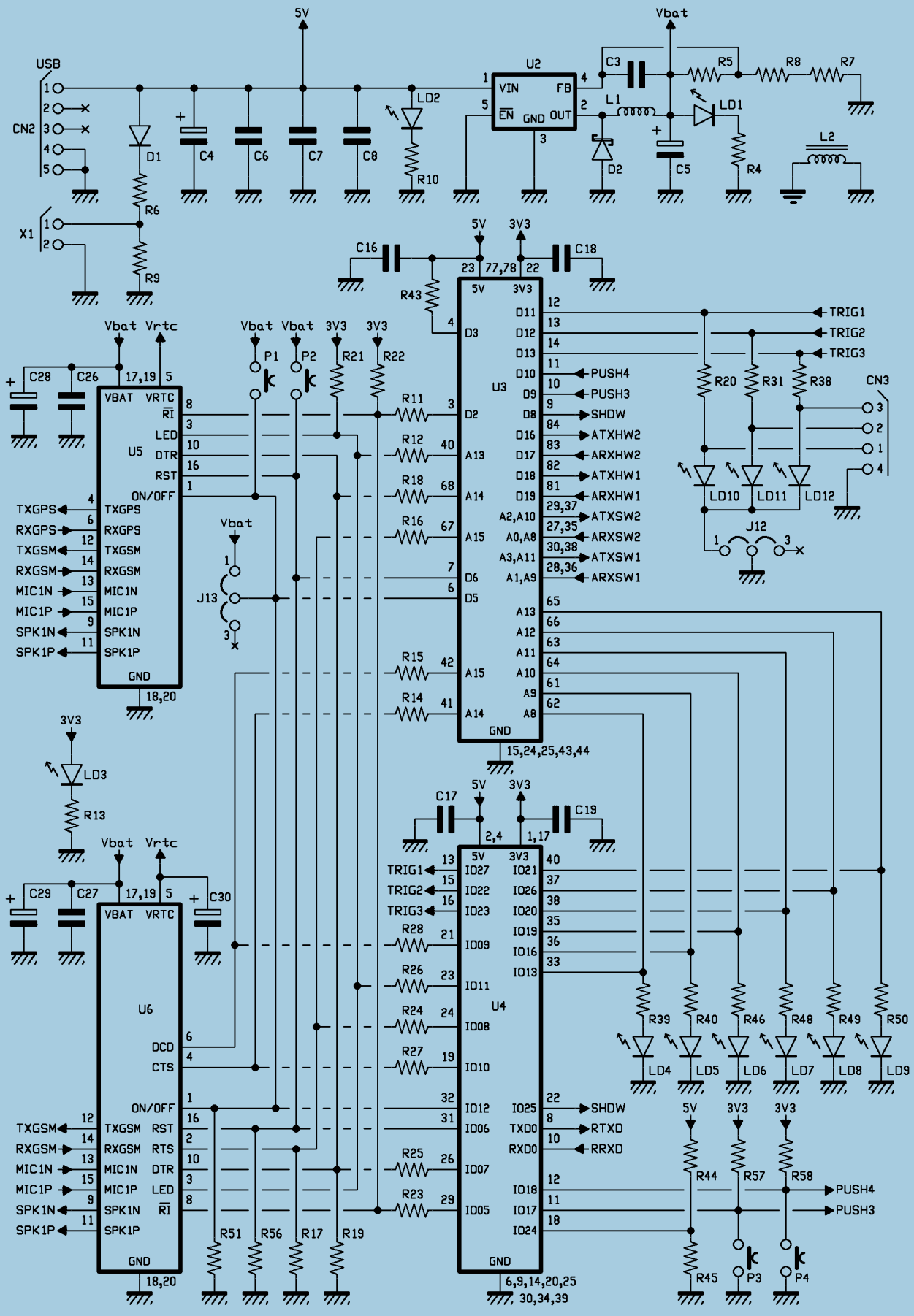
- Scheda Raspberry Pi 3 modello A+.
- Box protettivo in plexiglass trasparente per Raspberry Pi 3 modello A+.
- NOOBS su microSD da 16GB per Raspberry Pi 3 modello A+.
- Cavo HDMI 2.0 high speed
- Alimentatore switching 5 VDC / 2,5 A con uscita micro USB

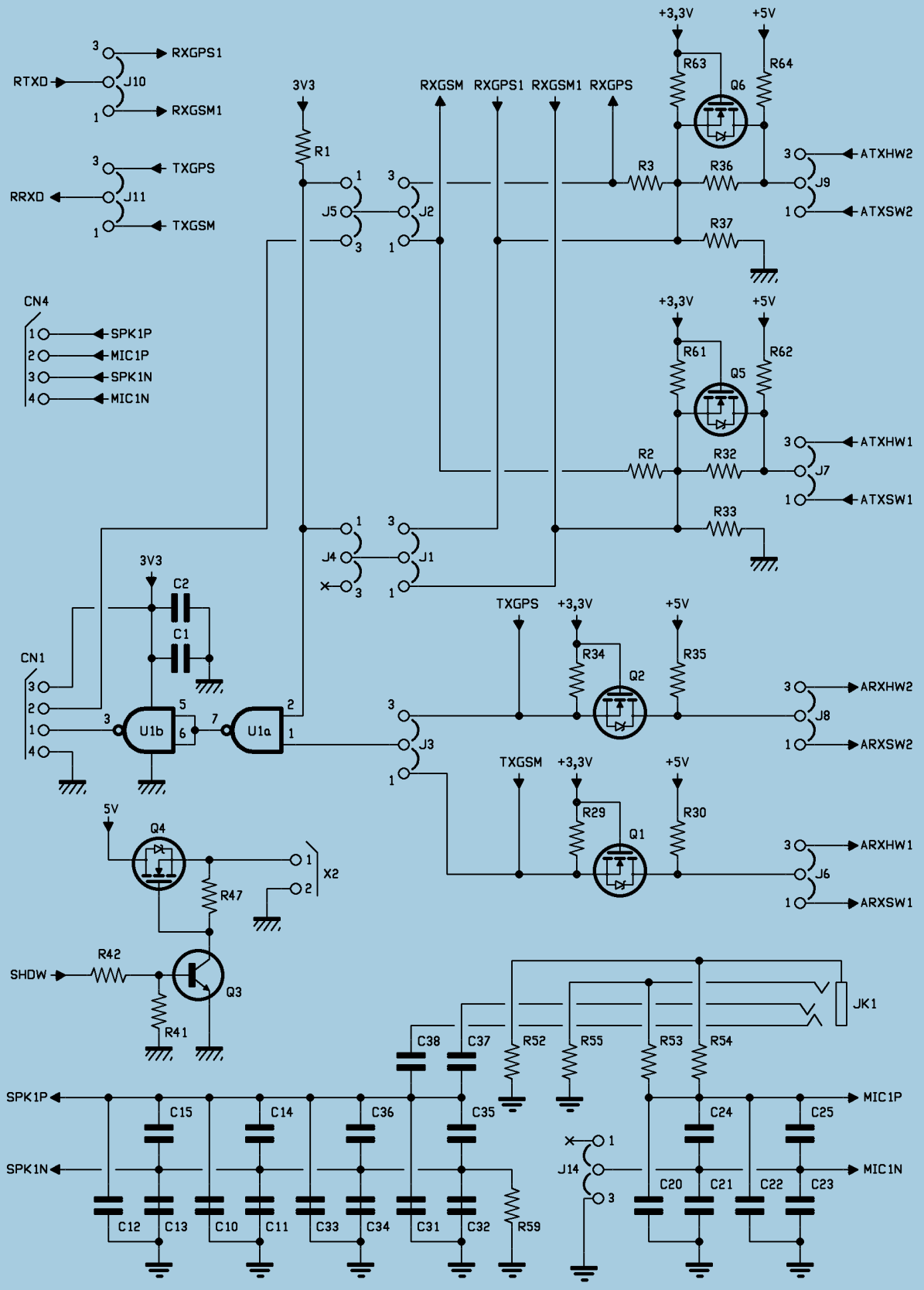
FUTURA ELETTRONICA®

www.futurashop.it

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

Caratteristiche tecniche di questi prodotti e acquisti on-line su www.futurashop.it


5V
Vbat
USB
CN2
X1
D1
D2
R6
R9
C4
C6
C7
C8
C3
C5
L1
L2
R4
R5
R7
R8
R10
LD1
LD2
U2
VIN
FB
EN
GND
OUT
C16
C18
R43
3V3
77,78
23
22
U3
D3
D2
D5
D6
A13
A14
A15
A12
A11
A10
A9
A8
A2,A10
A0,A8
A3,A11
A1,A9
D11
D12
D13
D10
D9
D8
D16
D17
D18
D19
PUSH4
PUSH3
SHDW
ATXHW2
ARXHW2
ATXHW1
ARXHW1
ATXSW2
ARXSW2
ATXSW1
ARXSW1
TRIG1
TRIG2
TRIG3
R20
R31
R38
CN3
LD10
LD11
LD12
J12
C28
C26
C29
C27
C30
U5
U6
VBAT
VRTC
RI
LED
DTR
RST
ON/OFF
TXGPS
RXGPS
TXGSM
RXGSM
MIC1N
MIC1P
SPK1N
SPK1P
GND
17,19
18,20
P1
P2
R21
R22
R11
R12
R18
R16
R15
R14
J13
LD3
R13
C17
C19
U4
IO27
IO22
IO23
IO09
IO11
IO08
IO10
IO12
IO06
IO07
IO05
IO21
IO26
IO20
IO19
IO16
IO13
IO25
TXD0
RXD0
IO18
IO17
IO24
R28
R26
R24
R27
R25
R23
R39
R40
R46
R48
R49
R50
LD4
LD5
LD6
LD7
LD8
LD9
R44
R57
R58
R45
P3
P4
R51
R56
R17
R19
6,9,14,20,25
30,34,39
15,24,25,43,44


RXGPS1
RTXD
J10
RXGSM1
TXGPS
RRXD
J11
TXGSM
3V3
R1
J5
J2
RXGSM
RXGPS1
RXGSM1
RXGPS
+3,3V
+5V
R63
R64
Q6
R3
R36
R37
ATXHW2
J9
ATXSW2
CN4
SPK1P
MIC1P
SPK1N
MIC1N
R61
R62
Q5
R2
R32
R33
ATXHW1
J7
ATXSW1
J4
J1
C2
C1
CN1
U1b
U1a
J3
TXGPS
R34
R35
Q2
ARXHW2
J8
ARXSW2
TXGSM
R29
R30
Q1
ARXHW1
J6
ARXSW1
5V
Q4
X2
R47
R42
SHDW
Q3
R41
JK1
C38
C37
R52
R55
R53
R54
SPK1P
C15
C14
C36
C35
C24
C25
MIC1P
SPK1N
J14
C20
C21
C22
C23
MIC1N
C12
C13
C10
C11
C33
C34
C31
C32
R59

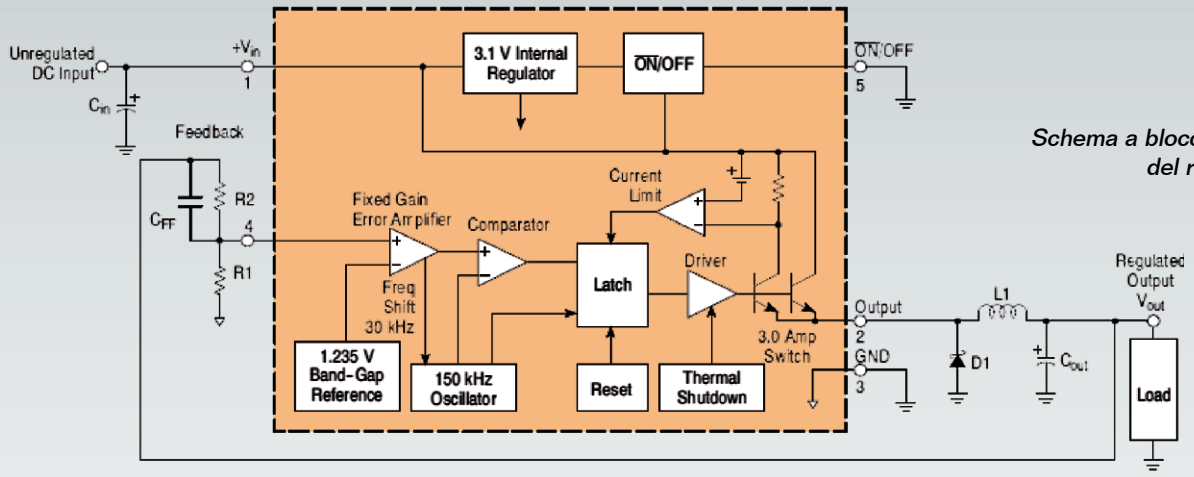


Fig. 1
Schema a blocchi interno del regolatore.

regolatore di tensione mettendo in evidenza la logica di funzionamento e i vari pin di ingresso/uscita e controllo. Come si può osservare, al pin 1 viene portata la tensione di ingresso mentre dal pin 2 si preleva la tensione switching di uscita che tramite la rete formata dal diodo Schottky, l'induttanza e il condensatore a bassa ESR restituisce la tensione di uscita regolata al valore desiderato. Il valore di tensione si ottiene tramite la retroazione con partitore resistivo da collegare al pin 4 di feedback.
È disponibile un ingresso digitale, pin 5, per accendere/spegnere il regolatore di tensione. La logica è attiva bassa e nella nostra applicazione lo forziamo a massa ovvero sempre acceso. La massa dell'integrato è il pin 3. Il principio di funzionamento del regolatore di tensione può essere riassunto con lo schema di **Fig. 2**.
L'integrato opera su due distinti periodi di tempo, il primo periodo è dato dalla condizione di switch ON mentre il secondo dalla condizione di switch OFF. Nella condizione di switch ON la tensione di ingresso è connessa direttamente all'ingresso dell'induttanza e il diodo di ricircolo è interdetto. Durante il periodo di OFF l'interruttore è aperto e la tensione sull'induttanza inverte la sua polarità e fa intervenire il diodo di ricircolo. La corrente fluisce attraverso il diodo mantenendo il loop di corrente con il carico. La frequenza con cui avviene l'accensione/spegnimento dell'interruttore elettronico è 150kHz come indicato in precedenza.
La regolazione del convertitore si ottiene variando il duty-cycle ovvero agendo sui tempi di ON e OFF dello switch come indicato dalla seguente:

$$d=\frac{t_{on}}{T}$$

Fatta questa premessa, calcoliamo i valori che devono assumere le resistenze di feedback per impostare la tensione di uscita al regolatore. La formula da seguire è la seguente (con riferimento alla **Fig. 1**):

$$V_{out}=1{,}23*\left(1+\frac{R2}{R1}\right)$$

che applicato al nostro circuito diventa:

$$V_{out}=1{,}23*\left(1+\frac{R5}{R7+R8}\right)=1{,}23*\left(1+\frac{5600}{200+2200}\right)=4{,}1\ V$$

L'alimentazione viene quindi portata ai moduli GSM, ovvero i connettori siglati U5 e U6, e ai pulsanti P1 e P2 rispettivamente usati per accendere/spegnere manualmente i moduli e resettare gli stessi (manualmente). Ad indicare la presenza dell'alimentazione +VBAT ci pensa il LED siglato LD1. Invece LD2 serve a indicare la presenza dell'alimentazione +5V e infine LD3 indica la presenza dell'alimentazione +3,3V ritornata dalle schede Arduino, piuttosto che Raspberry Pi.
La tensione di alimentazione +5V viene anche riportata, tramite partitore di tensione, sulla morsettiera X1, così da avere un riferimento di tensione da portare a un'elettronica esterna per sviluppi futuri. Completano la predisposizione la morsettiera X2 e il MOSFET a canale P Q4, utilizzato come interruttore elettronico.
Se il MOSFET Q4 è spento, ovvero la tensione VGS non supera la soglia di attivazione del canale, si ha che la tensione di alimentazione +5V, attraverso

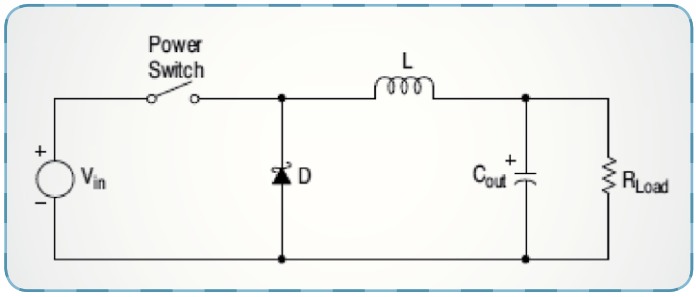


Fig. 2 *- Schema di principio del regolatore.*

il diodo interno del MOSFET, raggiunge il pin 1 della morsettiera X2. Ora se si porta alto il segnale "***shutdw***" succede che Q3 si accende portando a massa il gate di Q4 e, grazie alla tensione presente sul pin 1 di X2, la VGS supera la soglia attivando il canale di Q4. In questa condizione nel momento in cui viene a mancare l'alimentazione principale (+5V) e se sulla morsettiera è presente una tensione +5V generata da una scheda ausiliaria si avrà che la nostra elettronica continuerà a funzionare fintantoché non verrà portata bassa la linea "***shutdw***".
Passiamo ora a descrivere le interconnessioni presenti sia sulle schede Arduino, sia sulle schede Raspberry Pi. Le schede Arduino supportate sono Arduino Uno R3, Arduino Mega 2560 R3 e le schede Fishino (Uno e Mega). Nel caso in cui si usino le schede Fishino (usare schede con implementato ultimo firmware che libera I/O 7) è necessario ricordare che gli I/O 10, 11, 12 e 13 sono usati per il WiFi e nel caso si usi la scheda microSD anche I/O 4 deve essere libero. Dato che agli I/O 10, 11, 12 e 13 avevamo previsto di collegare il pulsante P4 e tre trigger per il debug si dovrà tenere conto di ciò quando si andrà ad usare la scheda Fishino Uno. Se invece si usa la scheda Fishino Mega gli I/O da lasciare liberi sono il 4, 10, 50, 51 e 52 (anche in questo caso usare schede con implementato ultimo firmware che libera I/O 7). Se si devono liberare gli I/O 11, 12 e 13 spostare il jumper J12 sulla posizione 2-3.
Per quanto riguarda le schede Raspberry Pi sono supportate: Raspberry Pi 2 model B+, Raspberry Pi 3 model B e Raspberry Pi 3 model B+.
Cominciamo dalle schede Arduino, per comodità abbiamo usato come riferimento di partenza la pinout della scheda Arduino Mega con evidenziati in azzurro i pin corrispondenti della scheda Arduino Uno. Se si usa la scheda Arduino Mega è possibile scegliere se sfruttare le due UART hardware, UART1 e UART2, mappate sui pin di I/O 16, 17, 18 e 19 oppure le due UART software che sono mappate sui pin di I/O A8 (RX UART2), A9 (RX UART1), A10 (TX UART2) e A11 (TX UART1). I corrispondenti I/O per la scheda ArduinoUno sono A0, A1 A2 e A3. Le UART di tipo software, per quanto riguarda il segnale di ricezione, devono essere mappate obbligatoriamente sui pin di I/O che supportano il port change interrupt. Quindi prestate attenzione nello sviluppare eventuale hardware per le vostre applicazioni.
I segnali TX e RX delle due UART, sia hardware che software, devono subire un adattamento di livello in quanto le schede Arduino lavorano con livelli logici a +5V mentre i moduli GSM lavorano con livello a +3,3V. Quindi il segnale TX, prima di giungere al motore GSM, viene adattato tramite l'utilizzo di un mosfet a canale N opportunamente collegato. Stesso discorso per il segnale RX che dal modulo GSM giunge alla scheda Arduino. Per capire meglio quanto detto esaminiamo la connessione delle linee TX e RX, sia hardware che software, usate per inviare i comandi AT per la gestione della sezione GSM. Come si può osservare dallo schema elettrico, lato Arduino, le linee interessate sono identificate come segue "ARD_RXD1_HW", "ARD_TXD1_HW", "ARD_RXD1_SW" e "ARD_TXD1_SW".
Queste giungono a dei jumper di selezione, J6 e J7, necessari per decidere se sfruttare una connessione seriale di tipo hardware oppure software. Quindi se i jumper sono in posizione 1-2 viene identificata una connessione UART software mentre se i jumper sono in posizione 2-3 viene identificata una connessione UART hardware.
Per quanto riguarda il segnale TX Arduino, che sia hardware o software, giunge al MOSFET Q5, il quale adatta il ivello di tensione a +3,3V compartibile con l'elettronica dei moduli GSM. Oltre al MOSFET Q5 sono state predisposte anche due resistenze, non montate, in configurazione partitore di tensione per adattare il ivello di tensione da +5V a +3,3V (Solo per la linea TX). Per quanto riguarda il segnale RX, sempre lato Arduino, abbiamo che questo viene portato al drain del MOSFET Q1 che così collegato permette di adattare il segnale proveniente dal motore GSM a +3,3V al livello di tensione +5V di Arduino. Questo tipo di adattamento lo abbiamo già usato in tante altre applicazioni.
Lo stesso ragionamento è stato applicato alle linee TX e RX per la gestione dei comandi AT del GPS e quindi facenti capo ai jumper J8 e J9.
Come accennato, è possibile spiare i comandi AT che vengono inviati al modulo GSM/GPS e le relative risposte sfruttando l'elettronica formata dall'integrato U1 e dai jumper J1, J2, J3, J4 e J5.
A completare la spia il solito convertitore USB/Seriale FT782M della Futura Elettronica configurato per lavorare con I/O a livello logico +3,3V e inserito nell'apposito connettore CN1. Grazie a questa elettronica e a una opportuna configurazione dei jumper è possibile fare le seguenti:

1) spiare i comandi AT inviati al modulo GSM/GPS da scheda Arduino/Raspberry Pi;
2) spiare le risposte dei comandi AT inviati alla scheda Arduino/Raspberry Pi;
3) inviare i comandi AT direttamente da PC escludendo le linee TX e RX di Arduino/Raspberry Pi.

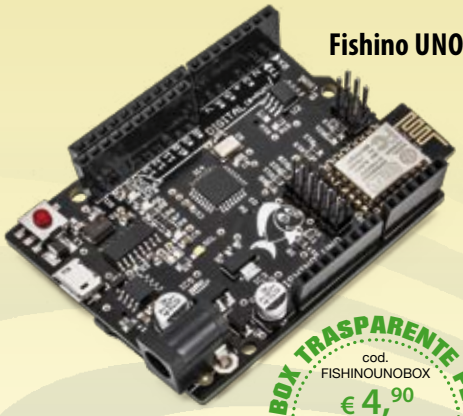
Quindi se si vogliono spiare i comandi AT, e le rela-

Fishino
Design With Simplicity

I TUOI PROGETTI DIVENTANO WIRELESS

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

Fishino UNO



- Controller: ATmega328
- Alimentazione:
  - da 7 a 12 Vdc (tramite plug)
  - 5 Vdc (tramite porta USB)
- Compatibile al 100% con Arduino UNO
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Modulo RTC con batteria di mantenimento
- Sezione di alimentazione a 3,3 V potenziata
- Compatibile con shield e schede millefori
- Peso 25 g
- Dimensioni 76,5 x 53,5 x 14 mm

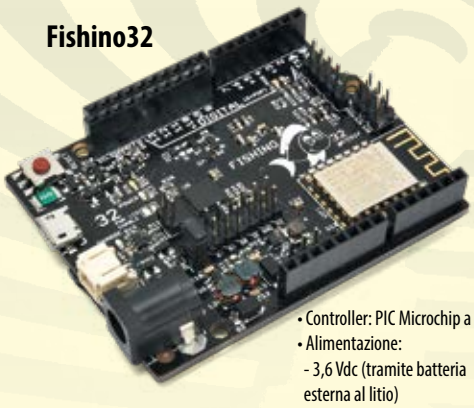
cod. FISHINOUNO
€ 36,00

Fishino MEGA

- Controller: ATmega2560
- Alimentazione:
  - 3,6 Vdc (tramite batteria esterna al litio)
  - da 3,5 a 20 Vdc (tramite plug)
  - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile al 100% con Arduino MEGA
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Modulo RTC con batteria di mantenimento
- Stadio di alimentazione switching (5 Vdc e 3,3 Vdc)
- Compatibile con shield e schede millefori
- Peso 37 g
- Dimensioni 101,5 x 53,5 x 15 mm

cod. FISHINOMEGA
€ 49,90

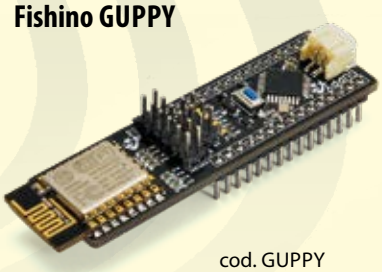
Fishino32



- Controller: PIC Microchip a 32 bit
- Alimentazione:
  - 3,6 Vdc (tramite batteria esterna al litio)
  - da 3,5 a 20 Vdc (tramite plug o sull'ingresso Vin)
  - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile con Arduino UNO
- Modulo WiFi
- Modulo RTC con batteria di mantenimento
- Clock 120 MHz, riducibile via software
- Interfaccia per scheda microSD
- Codec audio stereo
- Peso 29 g
- Dimensioni 76,5 x 53,5 x 14 mm

cod. FISHINO32
€ 59,00

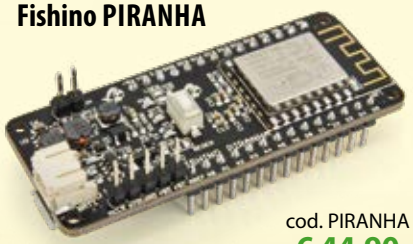
Fishino GUPPY



- Controller: ATmega328
- Alimentazione:
  - 3,6 Vdc (tramite batteria esterna al litio)
  - da 6,5 a 20 Vdc (tramite PIN VIN)
  - 5 Vdc (tramite porta USB)
- Circuito di ricarica per batteria al litio
- Compatibile al 100% con Arduino NANO
- Modulo WiFi
- Interfaccia per scheda MicroSD
- Peso 10 g
- Dimensioni 75 x 20 x 18,5 mm

cod. GUPPY
€ 33,90

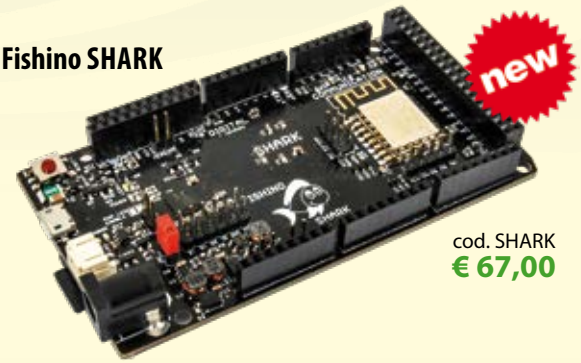
Fishino PIRANHA



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

cod. PIRANHA
€ 44,90

Fishino SHARK



cod. SHARK
€ 67,00

- Formato standard Arduino MEGA
- Processore PIC32MX470F512
- 512 kB di ROM
- 128 kB di RAM
- Frequenza di clock di 105 MHz
- Interfaccia USB nativa, sia device che host
- RTC incorporato
- Lettore per schede microSD
- Modulo WiFi integrato
- Codec audio
- Tensione di alimentazione: 3÷20Vdc
- Alimentazione a batteria, plug e/o via connettore USB
- Funzionamento interno a 3,3 V
- Pin DIGITALI 5V-tolerant
- Ricarica automatica di una batteria LiPo
- Spegnimento da software con processore in standby
- Wake-up tramite pin esterno o in tempi prefissati

Utilizza subito Fishino UNO!

Set contenente il necessario per realizzare applicazioni con Fishino UNO.

cod. FISHINOSTARTER
€ 65,00

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

Caratteristiche tecniche e vendita on-line su: **www.futurashop.it**

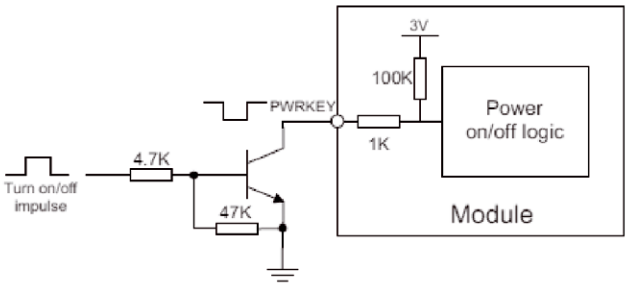


***Fig. 3** - Comando della linea di accensione del modulo GSM.*

tive risposte, inviati al modulo GSM i jumper J1, J2, J3, J4 e J5 devono essere in posizione 1-2. Se invece si vogliono spiare i dati ricevuti dal moduli GPS i jumper J1, J2 e J3 devono essere in posizione 2-3 mentre i jumper J4 e J5 devono essere in posizione 1-2. Se invece si volesse inviare i comandi AT direttamente da PC, ad esempio per studiare nuovi comandi da implementare negli sketch Arduino o Raspberry Pi, si devono configurare i jumper nel seguente modo: jumper J2 e J3 in posizione 1-2, jumper J4 e J5 in posizione 2-3 e jumper J1 indifferente (*configurazione per invio dei comandi AT a sezione GSM*). Oppure jumper J2, J3, J4 e J5 in posizione 2-3 e J1 indifferente (*configurazione per lettura dati GPS*).
Per escludere la possibilità di inviare comandi AT da PC in modo permanente è sufficiente non montare le due resistenze da 0 ohm R2 e R3. In questo modo è impossibile inviare comandi AT da PC verso il motore GSM/GPS in quanto la linea risulta interrotta. Consigliamo comunque di montare sempre la sezione spia, dato che risulta utile durante le fasi di studio e apprendimento dei comandi AT.
Inoltre data la possibilità di sfruttare un terminale PC per l'invio/ricezione dei comandi AT si ha un metodo di studio veloce e efficace che vi svincola dagli eventuali bug introdotti durante la scrittura di codice C per lo sviluppo o integrazione di una libreria Arduino/Raspberry Pi. In altre parole, prima si studia il comando AT che si vuole implementare sfruttando un terminale da PC e poi, solo dopo averne capito il funzionamento, si implementa nella propria libreria.
Tornando alle linee di comunicazione TX e RX, quelle che giungono dalla Raspberry Pi hanno già livelli compatibili con i moduli GSM e quindi non necessitano di adattamento di linea. Tuttavia si deve decidere tramite i jumper J10 e J11 se pilotare la sezione GSM piuttosto che quella GPS. Se J10 e J11 in posizione 1-2 si comunicherà con la sezione GSM, viceversa se in posizione 2-3 si comunicherà con sezione GPS.
Oltre alle linee TX e RX abbiamo predisposto anche le linee di controllo le quali però, attualmente, non sono utilizzate nella nostra libreria, ovvero le linee "*CTS*", "*DCD*", "*RTS*" e "*DTR*".
Molto più importanti le linee "*ON/OFF*" e "*RST*" che rispettivamente servono per accendere/spegnere il modulo GSM/GPS oppure resettare gli stessi.
La **Fig. 3** evidenzia come deve essere predisposta l'elettronica necessaria per accendere/spegnere il modulo GSM, come si può vedere l'impulso di accensione dovrà essere a logica positiva e la sua durata dovrà essere superiore a un secondo.
La **Fig. 4** mostra il diagramma temporale di accensione del modulo GSM e più precisamente mostra l'andamento che deve avere il segnale *PWRKEY* che si trova sul collettore del transistor. Quindi affinché l'operazione di accensione del GSM vada a buon fine, è necessario che, dopo avere dato alimentazione, si attenda almeno 500ms prima di inviare l'impulso di accensione (che deve durare più di 1 secondo).
Oltre all'accensione del GSM sfruttando l'apposita linea, abbiamo predisposto il pulsante P1 che assolve al medesimo compito; fornendo il livello logico 1 perché deve attivare il transistor che intrnamente al modulo pone a zero il *PWRKEY*.
La **Fig. 5** mostra l'andamento che deve avere il segnale *PWRKEY* allo spegnimento del modulo GSM: l'impulso a zero logico deve avere una durata superiore al secondo ma inferiore a 33 secondi. I diagrammi temporali appena mostrati facevano riferimento a un motore SIM800C. Se si usano altri motori , ad esempio un M95 della Quectel, la durata degli impulsi di accensione e spegnimento potrebbe essere diversa. In questo caso l'impulso di spegnimento deve avere una durata compresa tra 700ms e al massimo un secondo. Quindi è sempre buona norma consultare il datasheet del modulo.
Oltre alla linea di accensione e spegnimento, abbiamo predisposto una linea di reset ("*RST*") la quale serve appunto a resettare il motore GSM. Il pulsante

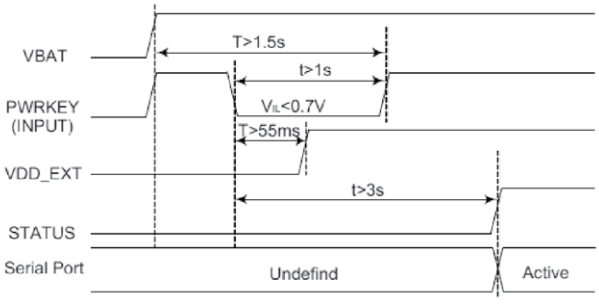


***Fig. 4** - Temporizzazioni dei segnali riguardanti il GSM.*

P2 assolve alla funzione di reset manuale.
Concludiamo la rassegna con la linea "***RI***", detta "***Ring Indicator***", la quale viene portata all'ingresso INT0 (ArduinoUno) o INT4 se fosse ArduinoMega. Questa linea, attiva a livello logico basso, indica la ricezione di un SMS oppure di una chiamata fonica. Se prendiamo come riferimento un motore SIM800C abbiamo che durante la ricezione di un SMS la linea va bassa per 120ms per poi ritornare alta. Invece se si riceve una chiamata fonica la linea va bassa e ritorna alta soltanto se si risponde alla chiamata fonica con apposito comando AT oppure se si rifiuta la chiamata sempre con apposito comando AT.
La **Fig 6** descrive l'andamento del segnale sulla linea RI (*Ring Indicator*) per quanto riguarda la ricezione di un SMS mentre la **Fig. 7** mostra l'andamento durante la ricezione di una chiamata vocale.
Notate il jumper J13, che può essere usato per forzare in boot i moduli GSM nel caso in cui sia necessario aggiornargli il firmware.
I moduli GSM solitamente presentano una sezione audio cui è possibile collegare un microfono e degli altoparlanti per poter effettuare delle chiamate vocali; solitamente si usano delle cuffie con microfono dotate di jack audio del microfono separato dal jack degli altoparlanti, oppure (sono le più comuni) con un unico jack, che però non è standardizzato ma esiste in due versioni: ***CTIA*** e ***OMTP*** (**Fig. 8**).
La differenza è nel contatto del microfono, che si trova in posizioni differenti. La nostra elettronica è predisposta per funzionare sia con il formato OMTP che con il formato CTIA: montando le resistenze R54 e R55 viene selezionato lo standard CTIA, mentre se si montano le resistenze R52 e R53 viene selezionato lo standard OMTP. Il connettore utilizzato SJ-43514, JK1, è un jack da 3,5mm della CUI-INC il quale ha la pin-out evidenziata dalla **Fig. 8**, dove notate la corrispondenza esistente tra il jack in formato OMTP/CTIA e i contatti del connettore, come già ampiamente detto l'unica differenza tra i due formati è il fatto che la linea microfonica e la GND sono invertite l'una rispetto all'altra soluzione.
Completano la sezione audio una serie di condensatori da collegare sulle linee differenziali degli altoparlanti e sulle linee differenziali del microfono. Per quanto riguarda i condensatori degli altoparlanti, ci sono due blocchi; uno da collegare vicino al GSM e l'altro vicino al connettore. Il segnale SPK_P viene portato tramite i condesantori C37 e C38 ai due altoparlanti, destro e sinistro, del connettore audio.
Non ci rimane che descrivere le linee di trigger e debug, utili per lo sviluppo del codice sia a livello di libreria sia a livello di sketch. Sulla scheda Arduino

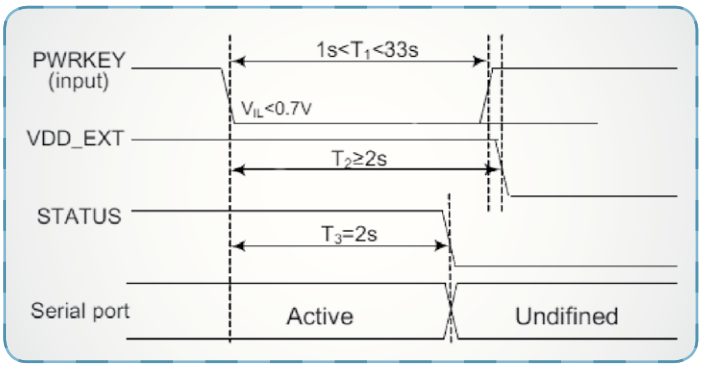


Fig. 5 *- Segnale PWRKEY allo spegnimento del GSM.*

sono state predisposte le linee di trigger *"Trigger1, 2 e 3"* e portate tramite resistenza serie al connettore CN3 al quale ci si può connettere con una sonda oscilloscopio a seconda dei casi. Le linee di trigger fanno capo ai pin di I/O 11, 12 e 13. Ricordiamo che nel caso delle schede Fishino tali pin sono condivisi con la sezione WiFi. Completano una serie di LED collegati agli I/O 32, 33, 34, 35, 36 e 37 e disponibili solo sulle schede Arduino Mega e Fishino Mega.
Sia le linee di trigger che le linee dei LED sono state portate alla scheda Raspberry Pi.

LAYOUT DEL PCB

Diamo un breve sguardo al layout della scheda, proposto nella **Fig. 9** che mostra le sezioni ognuna distinta d aun colore:

- ***sezione arancione***, dove trovano posto l'alimenta-

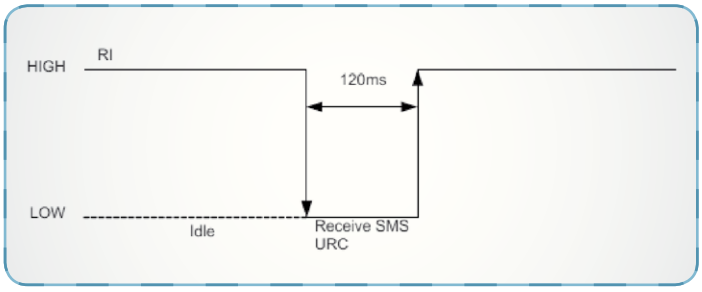


Fig. 6 *- Il segnale sulla linea RI del GSM all'arrivo di un SMS.*

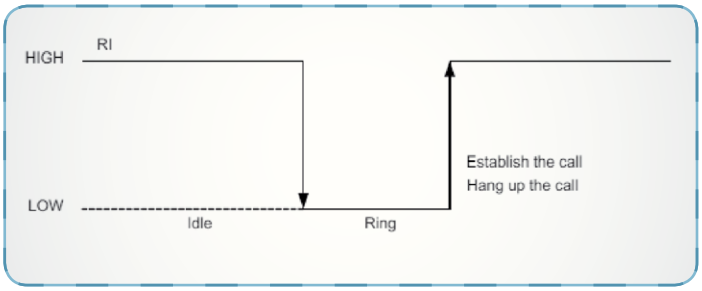
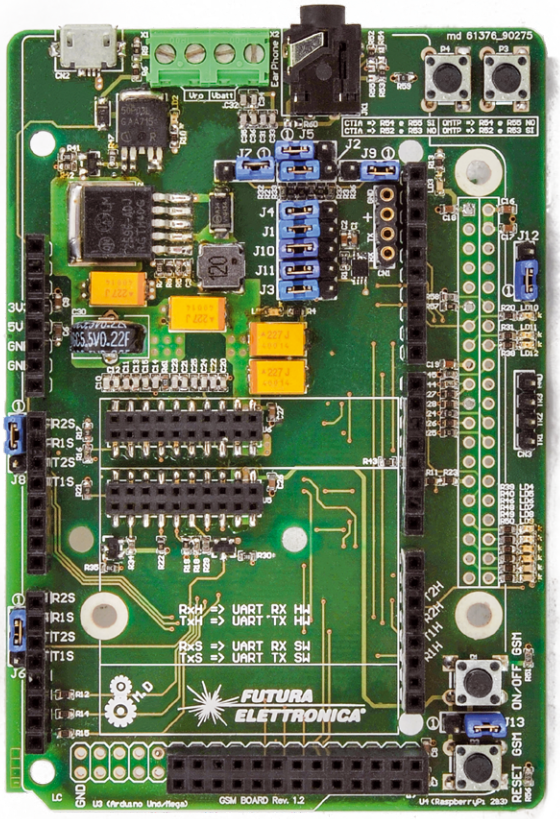
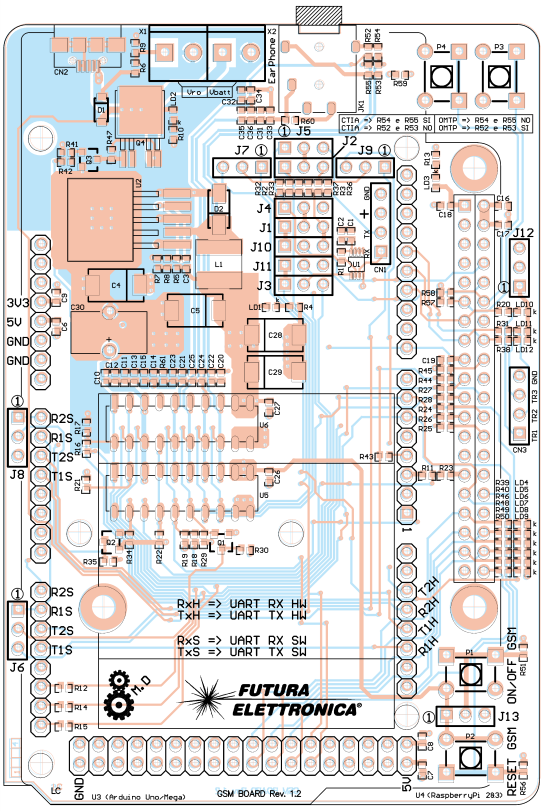


Fig. 7 *- Il segnale sulla linea RI del GSM all'arrivo di un telefonata.*

[piano di **MONTAGGIO**]

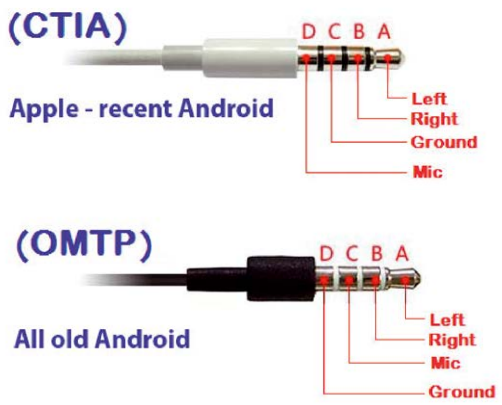


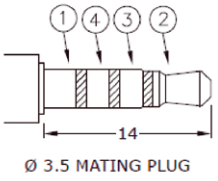


Elenco Componenti:

R1: 100 kohm 1% (0603)
R2, R3: 0 ohm (0603)
R4, R6, R10: 1,2 kohm 1% (0603)
R5: 5,6 kohm 1% (0603)
R7: 2,2 kohm 1% (0603)
R8: 200 ohm 1% (0603)
R9: 10 kohm 1% (0603)
R11, R12: 220 ohm 1% (0603)
R13: 1,2 kohm 1% (0603)
R14, R15: 220 ohm 1% (0603)
R16, R18: 5,6 kohm 1% (0603)
R17, R19: 10 kohm 1% (0603)
R20: 1,2 kohm 1% (0603)
R21, R22: 10 kohm 1% (0603)
R23: 220 ohm 1% (0603)
R24. R25: 0 ohm (0603)
R26÷R28: 220 ohm 1% (0603)
R29, R30: 10 kohm 1% (0603)
R31: 1,2 kohm 1% (0603)
R32, R33: -
R34, R35: 10 kohm 1% (0603)
R36, R37: -
R38÷R40: 1,2 kohm 1% (0603)
R39, R40: 1,2 kohm 1% (0603)
R41, R45: 10 kohm 1% (0603)
R42: 36 kohm 1% (0603)
R43: 220 ohm 1% (0603)
R44: 5,6 kohm 1% (0603)
R46, R48÷R50: 1,2 kohm 1% (0603)
R47: 22 kohm 1% (0603)
R51: 10 kohm 1% (0603)
R52, R53, R59: -
R54, R55: 0 ohm (0603)
R56÷R58: 10 kohm 1% (0603)
R60: 0 ohm (0603)
R61÷R64: 10 kohm 1% (0603)
C1: 100 nF 16 VL ceramico (0603)
C2: 10 µF 16 VL ceramico (0603)
C3: 10 nF 16 VL ceramico (0603)
C4, C5: 220 µF 16 VL tantalio (E)
C6, C7: 100 nF 16 VL ceramico (0603)
C8, C9: 10 µF 16 VL ceramico (0603)
C10, C11: 33 pF 16 VL ceramico (0603)
C12, C13, C15: 10 pF 16 VL ceramico (0603)
C14: 33 pF 16 VL ceramico (0603)
C16÷C19: 10 µF 16 VL ceramico (0603)
C20, C21: 10 pF 16 VL ceramico (0603)
C22÷C23: 33 pF 16 VL ceramico (0603)
C24: 10 pF 16 VL ceramico (0603)
C25: 33 pF 16 VL ceramico (0603)
C26, C27: 100 nF 16 VL ceramico (0603)
C28, C29: 220 µF 16 VL tantalio (E)
C30: 0,22 F 5 VL elettrolitico
C31, C32: 33 pF 16 VL ceramico (0603)
C32: 33 pF 16 VL ceramico (0603)
C33, C34: 10 pF 16 VL ceramico (0603)
C35: 33 pF 16 VL ceramico (0603)
C36: 10 pF 16 VL ceramico (0603)
C37, C38: 10 µF 16 VL ceramico (0603)
LD1, LD6, LD7: LED giallo (0603)
LD2, LD4, LD5: LED rosso (0603)
LD3, LD8, LD9: LED verde (0603)
LD10÷LD12: LED rosso (0603)
P1÷P4: Microswitch
D1: MMSD4148T1G
D2: B340A-13-F
Q1, Q2: BSS123
Q3: BC817
Q4: SPD50P03LG
Q5, Q6: BSS123
U1: SN74LVC2G00
U2: LM2596DSADJG
U3: Arduino Mega 2560 rev.3
U4: Raspberry Pi 3 B+
U5: Modulo GSM/GPS SIM928
U6: Modulo GSM G510/M95/SIM900
L1: VLS6045EX-150M

Varie:
- Connettore Micro-USB B
- Presa Jack 3.5mm da CS
- Strip maschio 3 vie (13 pz.)
- Strip maschio 4 vie (2 pz.)
- Jumper (13 pz.)
- Connettore 2x10 poli passo 2 mm femmina (2 pz.)
- Strip M/F Arduino 8 vie (5 pz.)
- Strip M/F Arduino 10 vie
- Strip M/F Arduino 2x18 vie
- Strip M/F Raspberry Pi 2x20 vie
- Strip femmina 4 vie
- Morsetto 4 vie
- Circuito stampato S1416 (110x158mm)





Model No.	SJ-43514	OMTP	CTIA
Schematic			
PIN			
1	sleeve	MICROPHONE	GND
2	tip	LEFT SPEAKER	LEFT SPEAKER
3	ring 1	RIGHT SPEAKER	RIGHT SPEAKER
4	ring 2	GND	MICROPHONE

***Fig. 8** - A sinistra, spinotti audio tipo CTIA e OMTP; qui sopra, piedinatura del jack CUI-INC.*

zione portata dal connettore microUSB, la morsettiera per collegare un'eventuale sistema di backup a batteria, il jack audio e due pulsanti ausiliari per sviluppi futuri;

- ***sezione verde,*** dove trovano posto i jumper di selezione delle varie modalità di comunicazione e spia (trovate una tabella riassuntiva più avanti);
- ***sezione grigia,*** dove trovano posto i due connettori per l'alloggiamento delle breakout board GSM;
- ***sezione rossa,*** dove trovano posto i led di trigger e diagnostica nonché il connettore a pettine per il collegamento di una eventuale sonda oscilloscopio;
- ***sezione blu***, dove trovano posto i due pulsanti usati per l'accensione/spegnimento del motore GSM nonché il pulsante di reset.

Lo shield è disponibile in versione già montata, tuttavia chi vuole cimentarsi nel montaggio dei componenti sulla scheda suggeriamo di partire dall'integrato U11 seguito da Q1 e U2. Poi vanno montati tutte le resistenze e i condensatori SMD in case 0603, i due connettori U5 e U6 e successivamente la rimanente componentistica SMD a partire dai LED. Per ultimo montate il jack audio, la morsettiera e tutti i jumper e pulsanti. Lasciate per ultimi i connettori di interconnessione delle schede Arduino e Raspberry Pi.
La **Tabella 1** riepiloga i jumper e come devono essere impostati.

COMANDI AT DA PC TRAMITE SPIA SERIALE

Molti dei comandi AT disponibili, oltre alla sintassi e al numero di parametri ad essi associati, prevedono delle risposte più o meno articolate, formate da diversi parametri, che necessitano del codice per essere decodificate e rese disponibili all'utente finale per l'implementazione nella propria applicazione. Quindi è fondamentale poter studiare il comportamento dei comandi AT che si vuole utilizzare prima di implementarli nelle proprie applicazioni o librerie. Inoltre è bene sapere che per raggiungere alcuni obiettivi, come ad esempio la gestione delle connessioni GPRS, è necessario impartire diversi comandi AT per attivare la connessione dati. Aggiungiamo che oltre alla sintassi è necessario anche studiare e capire la giusta sequenza di comandi AT da inviare perché alcuni comandi per essere recepiti e accettati necessitano l'invio preventivo di altri comandi AT prima di essi e nella giusta sequenza.
Un esempio di questo lo si ha quando si tenta la connessione a un APN tramite il comando AT+CSTT in una connessione dati di tipo TCP/UDP. Prima di inviare il comando AT di connessione allo APN si deve prima inviare il comando AT+CIPRXGET (***Get Data From Network Manually***) se non si vuole vedere ritornare il codice di errore "3", ovvero operazione non permessa, inviando il comando AT+CIPSTART.
Uno strumento utile allo scopo è l'utilizzo di sof-

	Jumper comunicazione seriale e spia										
Descrizione	**J1**	**J2**	**J3**	**J4**	**J5**	**J6**	**J7**	**J8**	**J9**	**J10**	**J11**
Arduino impostazione jumper											
GSM → RXD1 e TXD1 SW (***No spia PC***)	X	X	X	2-3	2-3	1-2	1-2	X	X	X	X
GSM → RXD1 e TXD1 SW (***Si spia PC***)	1-2	1-2	1-2	1-2	1-2	1-2	1-2	X	X	X	X
GSM → RXD1 e TXD1 HW (***No spia PC***)	X	X	X	2-3	2-3	2-3	2-3	X	X	X	X
GSM → RXD1 e TXD1 HW (***Si spia PC***)	1-2	1-2	1-2	1-2	1-2	2-3	2-3	X	X	X	X
GPS → RXD2 e TXD2 SW (***No spia PC***)	X	X	X	2-3	2-3	X	X	1-2	1-2	X	X
GPS → RXD2 e TXD2 SW (***Si spia PC***)	2-3	2-3	2-3	1-2	1-2	X	X	1-2	1-2	X	X
GPS → RXD2 e TXD2 HW (***No spia PC***)	X	X	X	2-3	2-3	X	X	2-3	2-3	X	X
GPS → RXD2 e TXD2 HW (***Si spia PC***)	2-3	2-3	2-3	1-2	1-2	X	X	2-3	2-3	X	X
RaspberryPi impostazione jumper											
GSM → RX e TX (***No spia PC***)	X	X	X	2-3	2-3	X	X	X	X	1-2	1-2
GSM → RX e TX (***Si spia PC***)	1-2	1-2	1-2	1-2	1-2	X	X	X	X	1-2	1-2
GPS → RX e TX (***No spia PC***)	X	X	X	2-3	2-3	X	X	X	X	2-3	2-3
GPS → RX e TX (***Si spia PC***)	2-3	2-3	2-3	1-2	1-2	X	X	X	X	2-3	2-3
Comandi AT da PC impostazione jumper											
GSM	X	1-2	1-2	2-3	2-3	X	X	X	X	X	X
GPS	X	2-3	2-3	2-3	2-3	X	X	X	X	X	X

Tabella 1

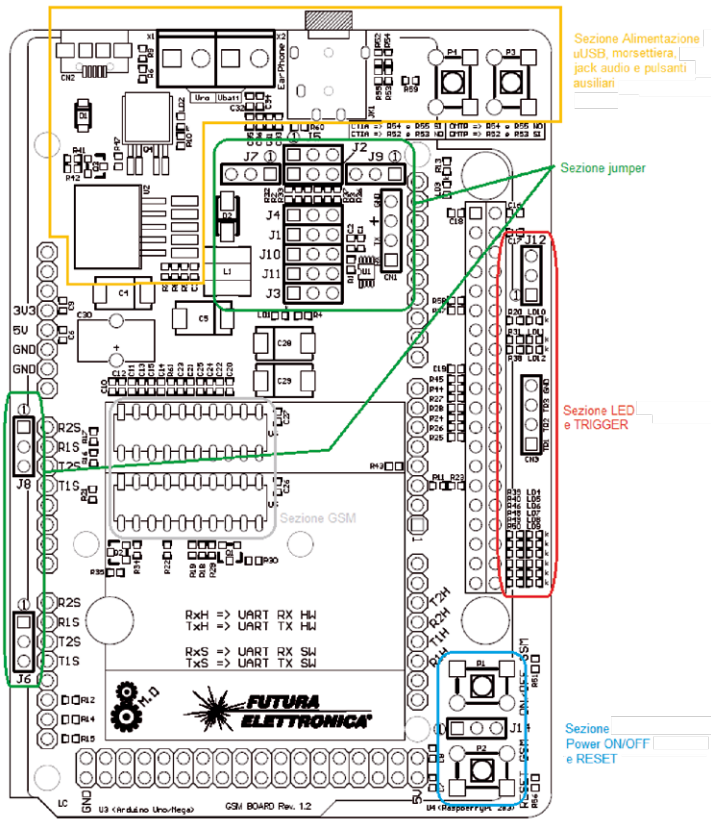


Fig. 9 - *Suddivisione del PCB in aree funzionali.*

tware per l'analisi e la simulazione dei protocolli di comunicazione seriale. Riportiamo di seguito alcuni esempi nei quali mostreremo delle sezioni di spia dei comandi AT scambiati tra le schede Arduino e il modulo GSM e l'invio dei comandi AT da PC per la configurazione del modulo GSM per lavorare in modalità GPRS.

La **Fig. 10** mostra il monitor seriale dell'IDE Arduino durante le fasi di inizializzazione del modulo GSM: in questo caso è un SIMCOM SIM800C.

All'avvio del monitor seriale, Arduino viene resettata e viene dato inizio alla procedura di inizializzazione dei moduli GSM, accensione compresa; tuttavia, dato che si partiva da una condizione di modulo GSM già acceso, l'impulso inviato al motore in realtà lo spegne e questo si deduce dalla ricezione della stringa "***NORMAL POWER DOWN***" (vedere sezione cerchiata in blu). L'algoritmo di inizializzazione è in grado di capire queste situazioni e automaticamente provvede alla riaccensione del modulo GSM, questo si capisce dalla ricezione della stringa "***RDY***" (vedere sezione cerchiata in rosso). Segue poi una serie di comandi AT per inizializzare il motore GSM tra cui l'invio del codice PIN. Le informazioni che si

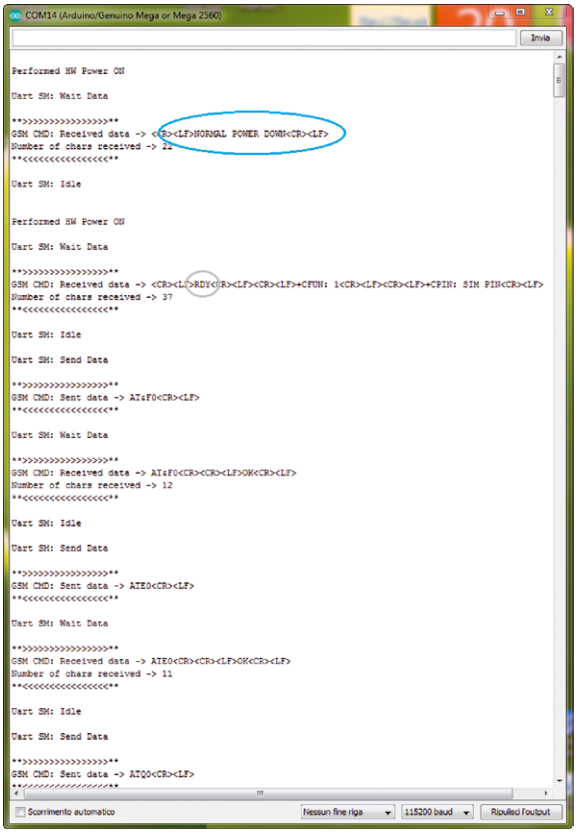


Fig. 10 - *Monitor seriale con il modulo GSM in funzione.*

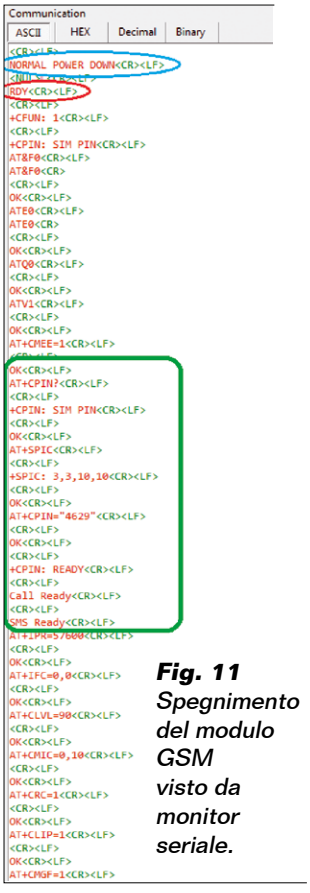


Fig. 11 *Spegnimento del modulo GSM visto da monitor seriale.*

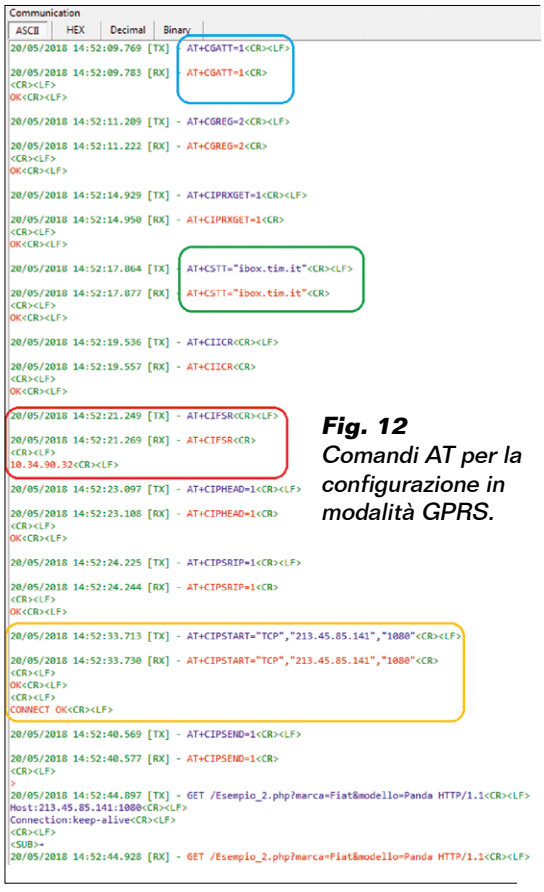


Fig. 12 *Comandi AT per la configurazione in modalità GPRS.*

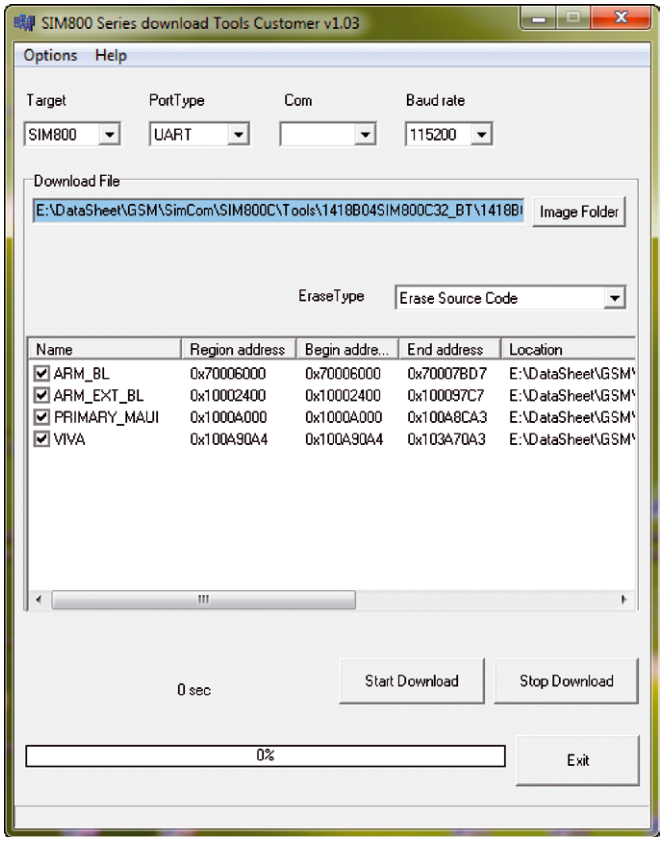


Fig. 13 *- Finestra di aggiornamento.*

vedono nel monitor seriale dello IDE possono essere abilitate/disabilitate via codice, tali informazioni sono molto utili durante le fasi di debug del codice. Tuttavia per avere una visione reale delle informazioni scambiate tra motore e schede Arduino/Raspberry Pi si deve usare una spia seriale in hardware coadiuvata da un apposito software di analisi.
La **Fig. 11** evidenzia, cerchiato in blu, lo spegnimento del modulo GSM da parte della libreria e la successiva riaccensione (in rosso). Cerchiati in verde vedete i comandi AT inviati per verificare se è necessario inviare il codice PIN per attivare la SIM, quanti tentativi abbiamo a disposizione per l'invio del codice PIN e l'avvenuta attivazione della SIM dopo l'invio del codice stesso.
La **Fig. 12** invece mostra l'invio dei comandi AT da PC per la configurazione del modulo GSM in modalità GPRS, in particolare durante una connessione dati verso un server tramite protocollo TCP; vedete evidenziato in blu il comando "***AT+CGATT=1***" usato per connettersi al servizio GPRS, in verde il comando "***AT+CSTT="ibox.tim.it***" usato per la connessione al proprio APN (GPRS Access Point Name), in rosso il comando "***AT+CIFSR***" che ritorna l'indirizzo IP assegnato dal GPRS, in giallo il comando "***AT+CIPSTART***" per la connessione, tramite protocollo TCP o UDP, a un server remoto e relativa porta.

Segue una serie di comandi per l'invio dei dati in particolare sfruttando del codice PHP presente sul server. Quanto appena mostrato evidenzia l'efficacia nell'invio dei comandi AT da PC per lo studio dei passi da seguire per configurare correttamente una connessione dati verso un server remoto. Una volta individuato il giusto flusso di comandi AT si può pensare di implementarli in codice nella propria libreria e relativo sketch.

AGGIORNAMENTO MODULI GSM

Con la connessione diretta al PC è possibile aggiornare il firmware dei moduli GSM; come esempio vediamo come aggiornare il firmware del SimCom SIM800C. Per prima cosa è necessario scaricare il tool, scaricabile ad esempio, dal seguente link:
www.dropbox.com/s/kn73rhcz8v72jhb/SIM800_Series_download_Tools_Customer_v1.06.rar?dl=0.
Non è necessario installare il software in quanto per utilizzarlo è sufficiente eseguire, in modalità amministratore, il file "***SIM800_Series_download_Tools_Customer.exe***" il quale aprirà la finestra di lavoro in **Fig. 13**, dove possiamo selezionare una serie di parametri tra cui il ***Target*** (nel nostro caso, SIM800C), l'interfaccia di comunicazione (***PortType2***, che in questo caso è una UART), la "***Com***" cui si è collegati e infine il ***Baudrate***. La ***Com*** usata nell'esempio è la COM15 e il baud-rate impostato è 115.200.
Il passo successivo consiste nel caricare il firmware, che nel caso dei SimCom è un file a estensione *.cfg*. L'aggiornamento conviene eseguirlo con il solo shield, perché è indispensabile che il GSM sia spento. Cliccate su "***Start Download***" e solo dopo accendere il motore GSM tenendo premuto il pulsante

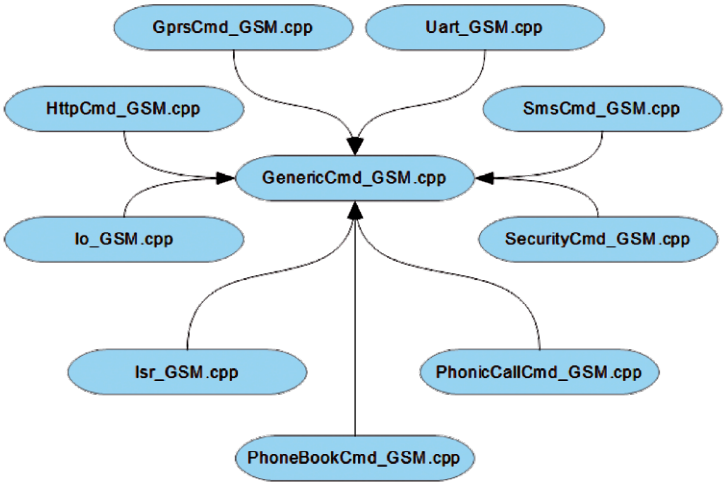


Fig. 14 *- Struttura dei file libreria.*

di accensione P1 per più di 4 secondi o, se preferite, inserendo il jumper J14 nella posizione 1-2 fino a che non parte la procedura di aggiornamento. L'aggiornamento del firmware avviene in tre step.

LIBRERIA GSM PER LE SCHEDE ARDUINO

L'architettura della libreria si basa su una serie di file in C++ e relativi file .h con le definizioni delle variabili, costanti, struttura dati ecc. Il file principale, contenente anche la procedura di accensione e inizializzazione del modulo GSM, serve da base per gli altri file che compongono la libreria (**Fig. 14**).
Ci sono diversi file di contorno a quello principale tra cui:

- gestione delle connessioni UART "***Uart_GSM.cpp***"
- gestione degli I/O "***Io_GSM.cpp***"
- gestione degli interrupt utilizzati dalla libreria "***Isr_GSM.cpp***"
- gestione degli SMS "***SmsCard_GSM.cpp***"
- gestione chiamate foniche "***PhonicCallCmd_GSM.cpp***"
- gestione funzione di sicurezza "***SecurityCmd_GSM.cpp***"
- gestione rubrica telefonica "***PhoneBookCmd_GSM.cpp***"
- gestione connessione GPRS "***GprsCmd_GSM.cpp***"
- gestione comandi HTPP "***HttpCmd_GSM.cpp***".

I file contengono il codice per la gestione di una serie di comandi AT sia per l'invio dei comandi, con i relativi parametri, sia per la decodifica delle risposte ricevute dal GSM compresi i codici di errore. Attualmente sono gestiti soltanto alcuni comandi AT, ma siccome la libreria è in continuo sviluppo verranno aggiunti sempre più comandi. Stesso discorso per la gestione dei codici di errore.
La **Fig. 15** propone la macchina a stati per l'invio dei comandi AT e gestione della risposta da parte del GSM: si inizia con l'inizializzazione del motore GSM per poi entrare in un loop di invio comandi AT. Per ogni comando AT inviato si attende la risposta da parte del motore GSM e se ne decodifica il contenuto. Se la risposta è OK si passa al comando AT successivo, altrimenti il sistema ritorna il codice di errore e a seconda dei casi ritenterà l'invio del comando AT fino a un massimo di tre volte. Se il comando AT non va a buon fine, nemmeno dopo tre tentativi, il sistema resetta il motore GSM e ricomincia da capo re-inizializzando lo stesso per poi ritornare nel loop di invio comandi AT.
Questo è a grandi linee l'approccio che c'è alla base della nostra libreria. Le funzioni per la gestione dei comandi AT che non vengono usate non occupano spazio di memoria Flash, tuttavia per ogni categoria

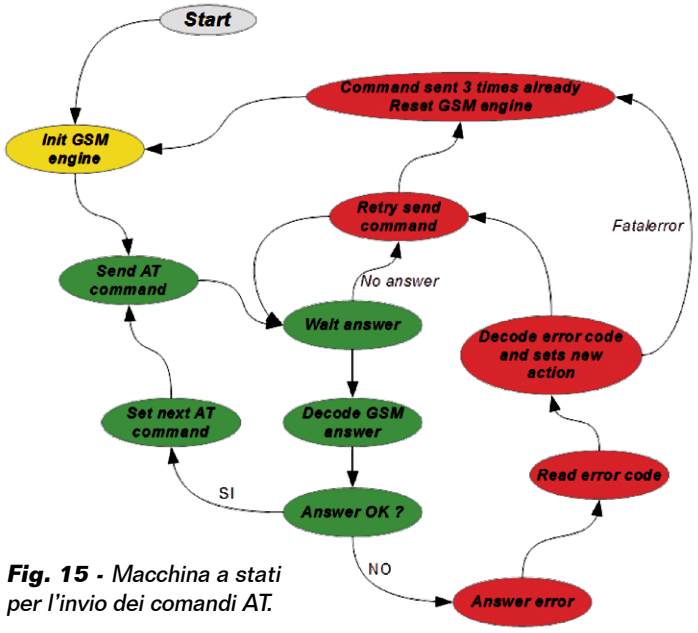
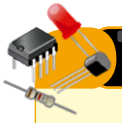


***Fig. 15** - Macchina a stati per l'invio dei comandi AT.*

di comandi AT esiste una propria sezione di gestione delle risposte, la quale è gestita da una opportuna macchina a stati. In questo caso, se abilitate con apposito *#Define*, le routine di gestione delle risposte vengono inserite nella compilazione del codice macchina e quindi occupano spazio in Flash. Il tutto sta nel cercare il giusto equilibrio tra funzioni necessarie alle proprie applicazioni e spazio in memoria occupato dalla libreria.
La libreria supporta le schede Arduino Uno e Fishino Uno, Arduino Mega 2560 o Fishino Mega 2560. In futuro non escludiamo di estendere la compatibilità alle Arduino più evolute, ovvero quelle con microcontrollore a 32 bit. Ciò detto, concludiamo qui questa prima puntata.



per il MATERIALE

Lo shield universale GSM (cod. WWGSMSHIELD) è in vendita presso Futura Elettronica al prezzo di Euro 54,90. Viene fornito montato ad esclusione degli strip line (compresi) che vanno saldati manualmente. Il prezzo è comprensivo di IVA.
Lo shield si può interfacciare con uno dei seguenti moduli GSM: interfaccia con M95 (cod. FT1128M, Euro 39,00), modulo cellulare miniaturizzato con SIM800 (cod. FT1308M, Euro 29,00) e modulo cellulare GSM/GPS con SIM928A (cod. FT1178M, Euro 59,00).

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