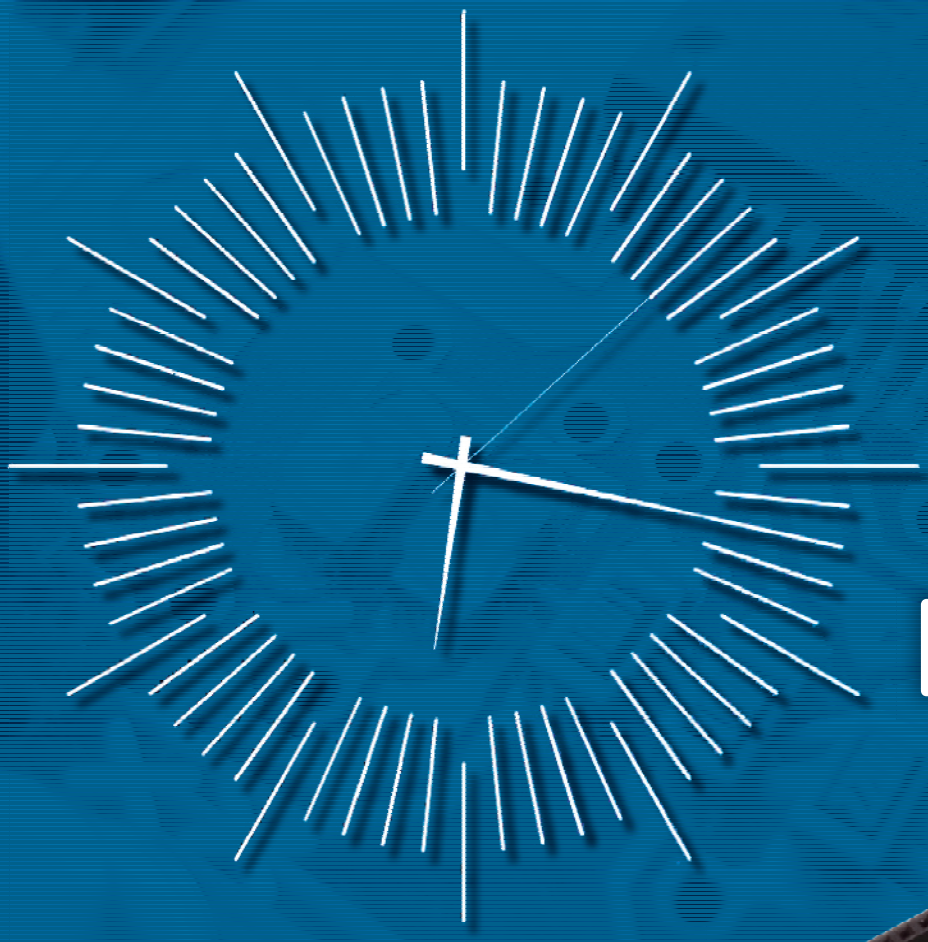
18 : 17

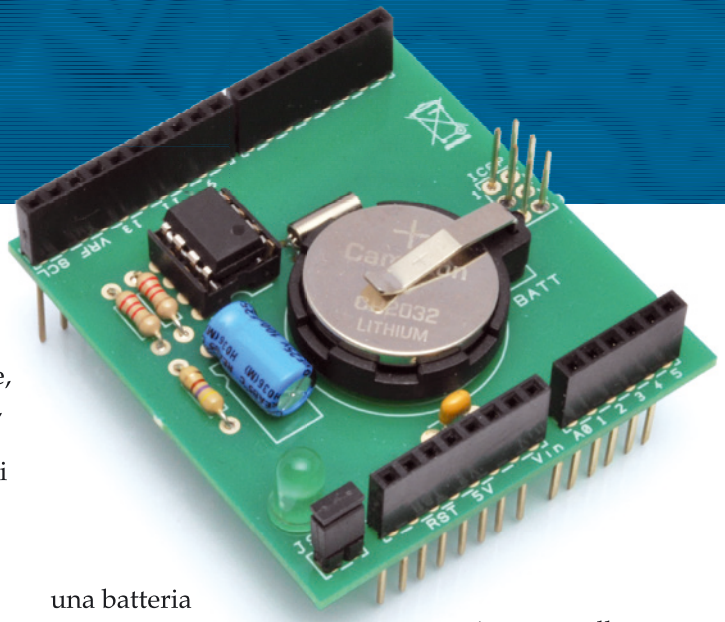
4/11/2011

FRIDAY
4
NOVEMBER

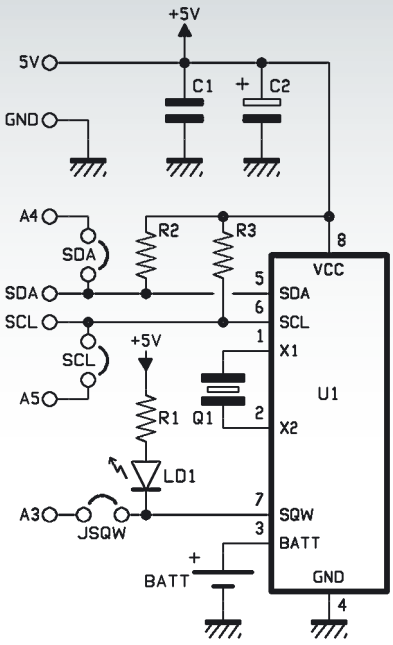


Modulo Real Time Clock per le applicazioni in cui ad Arduino serve mantenere un'ora di sistema precisa.

RTC SHIELD



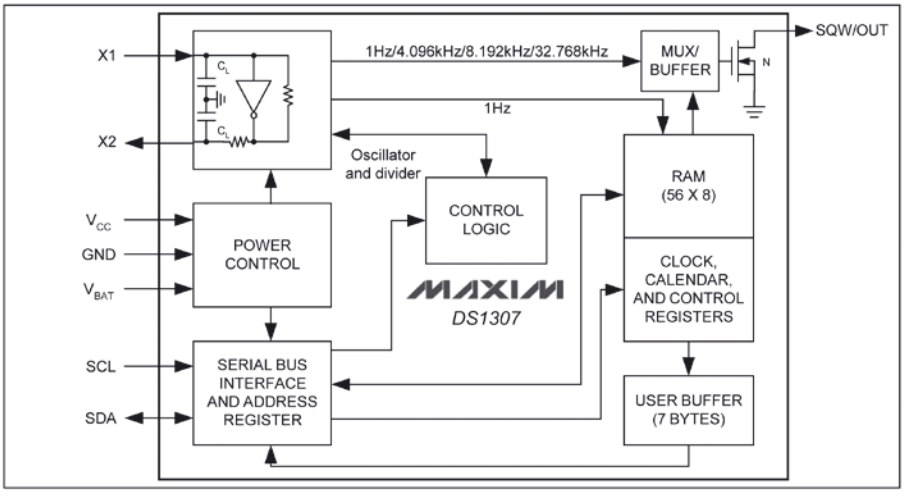
Avere un orologio di sistema è sempre utile: apparati come controlli accessi, cartellini orari elettronici ad RFID o badge magnetico, centralini telefonici, impianti di controllo e gestione delle riprese video, etichettatrici, ad esempio, hanno bisogno di conoscere o memorizzare insieme agli eventi l'orario e la data corrispondenti. Lavorando con i microcontrollori, l'orologio di sistema può essere implementato con un'apposita routine, che, mediante un timer, fa avanzare secondi, minuti, ore, giorni, mesi ed eventualmente anni; resta però il problema di mantenere l'orario quando il micro viene privato dell'alimentazione. Ciò si risolve scrivendo i dati dell'orologio di sistema in una RAM protetta da una batteria tampone. Realizzare la funzione di orologio richiede un certo impegno da parte del microcontrollore, quindi se ci serve l'ora di sistema e le risorse del micro sono già utilizzate al massi-



mo, l'unica alternativa è delocalizzare l'orologio, ossia delegare il compito ad un altro circuito. Per i sistemi basati su Arduino, soprattutto su Ardui-no Duemilanove ed UNO, che sono basati su microcontrollori ATmega 328, quindi dotati di risorse relativamente limitate, un valido aiuto può essere lo shield descritto in queste pagine, che è un completo RTC (Real Time Clock); l'uso di questo circuito consente di avere un orario preciso sollevando il microcontrollore dall'onere di gestire l'orologio di sistema, il che si traduce in due vantaggi: sgravare la CPU dal calcolo e dalla gestione dei dati orari e liberare spazio nella memoria di programma, che può così essere utilizzata per scrivere il codice di altre applicazioni.
Il nostro shield è stato realizzato in modo tale da rendere passanti tutti i pin presenti su Arduino, anche se non utilizzati. Lo shield si monta quindi in modo stabile, specie se viene inserito in una struttura a sandwich con sotto Arduino e sopra un altro shield, così da mantenerne dritti i connettori.

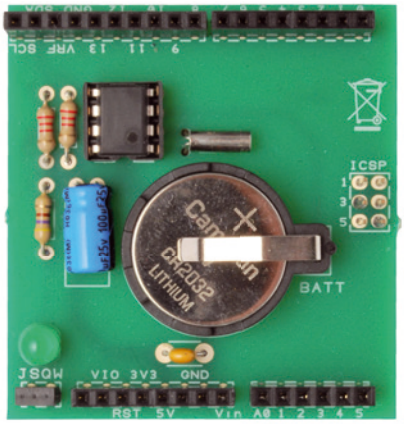
SCHEMA ELETTRICO

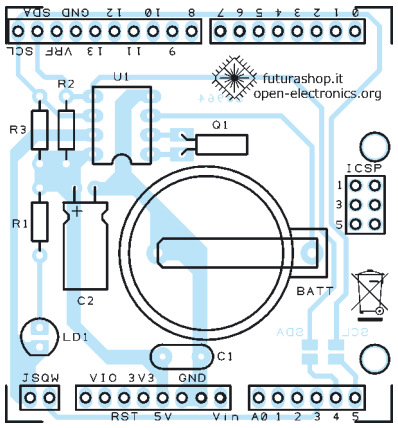
Il nostro orologio è basato sull'integrato DS1307 della Maxim-Dallas, il quale è un contatore BCD (Binary Coded Decimal) a basso consumo, che conta secondi, minuti, ore, giorni, mesi e anni, provvisto di 56 byte di RAM statica non volatile; il componente determina automaticamente quali sono i mesi con meno di 31 giorni e ad effettuare la correzione per l'anno bisestile. L'orologio può operare nelle modalità 12 o 24 ore, con indicazione delle ore antimeridiane (AM) e di quelle pomeridiane (PM). Le informazioni sull'ora e la data vengono collocate in un apposito registro e trasferite all'esterno al microcontrollore di Arduino mediante l'I²C-bus del quale il chip è provvisto; il bus fa capo ai piedini SDA (in alternativa pin A4) e SCL (in alternativa pin A5) di Arduino, il quale fa da unità master dell'I²C-Bus, mentre il chip della Maxim-Dallas è lo slave.
Oltre a ciò, il DS1307 dispone di un'uscita di clock programmabile: più esattamente, rende disponibile un'onda quadra ricavata dalla frequenza di clock dell'orologio (determinata, a sua volta, dal quarzo da 32.768 kHz collegato ai piedini 1 e 2) che, mediante un apposito divisore interno, può essere divisa di frequenza ottenendo 1 Hz, 4.096 kHz, 8.192 kHz o l'intero clock. Le frequenze ottenibili non sono state scelte a caso: per esempio, 1 Hz può servire a far lampeggiare i due punti o il punto dei secondi di un eventuale display che mostra l'ora. La condizione dell'uscita di clock ausiliario (SQWE, piedino 7) si definisce impostando opportunamente lo stato logico dei bit RS0 (0) ed RS1 (1) del registro di controllo, secondo quanto mostrato nell'apposito riquadro; ad esempio, 1 Hz si ottiene con entrambi i bit a zero. Si noti che quando sia il quarto (SQWE) che il settimo bit (OUT) si trovano a zero logico, l'uscita di clock si pone fissa a livello basso; se, invece, il bit 7 è ad uno logico e il 4 a zero, l'uscita assume costantemente lo stato alto. Nel nostro shield l'uscita SQW pilota in modo sink un LED, facendolo pulsare



Schema a blocchi dell'integrato DS1307: notate il contatore che conta il tempo ed il registro tramite cui si può verificarne da seriale lo stato.

alla stessa frequenza dell'onda quadra prodotta; inoltre, tramite il ponticello JSQW possiamo decidere se far leggere o no ad Arduino, tramite la linea A3, il segnale corrispondente.
Come accennato, questo clock ausiliario può servire per attivare dei visualizzatori o scandire certe sequenze: per esempio far suonare un cicalino ogni secondo al raggiungimento di una certa ora, per realizzare una sveglia; il tutto senza impegnare l'ATmega328 di Arduino in routine di temporizzazione.
Il DS1307 dispone anche di un circuito di controllo in grado di verificare la mancanza o l'insufficienza dell'alimentazione principale (Vcc) e fare in modo che la tensione occorrente al proprio funzionamento venga prelevata dalla batteria di backup, nel nostro caso una pila a bottone CR2032 da 3 V, che garantisce un'autonomia di circa 6 mesi in assenza di alimentazione e che deve essere collegata tra il piedino 3 (Vbat) e la massa (pin 4, ossia GND). La sezione di controllo dell'alimentazione è dimensionata in modo da svolgere due compiti: preservare la memoria in cui sono contenuti i dati sull'ora e data attuali e mantenere attivo il contatore dell'orologio. Il primo viene svolto conservando nella RAM le informazioni corrispondenti, mentre il secondo è espletato dall'alimentazione di riserva della batteria tampone collegata fra il piedino 3 e GND. La sezione di controllo dell'alimentazione interviene sulle funzioni del DS1307, bloccando la comunicazione con l'I²C-bus e la produzione dell'eventuale segnale di clock ausiliario SQW, in modo da minimizzare il consumo di energia ed estendere quanto più possibile l'autonomia.





Elenco Componenti:

R1: 470 ohm
R2, R3: 2,2 kohm
C1: 100 nF multistrato
C2: 100 µF 16 VL elettrolitico
Q1: Quarzo 32.768 kHz
LD1: LED 5 mm verde
U1: DS1307

Varie:
- Zoccolo 4+4
- Strip M/F 3 vie (2 pz.)
- Strip M/F 6 vie
- Strip M/F 8 vie (2 pz.)
- Strip M/F 10 vie
- Strip maschio 2 vie
- Jumper
- Porta batteria CR2032
- Batteria CR2032
- Circuito stampato

L'intervento della protezione avviene quando la tensione letta tra il piedino dell'alimentazione principale (8) e massa (4) è minore di 1,25 volte la tensione della pila. Per conoscere l'ora e la data, Arduino deve interrogare il DS1307 mediante l'I²C-bus di cui il chip è provvisto; allo scopo occorre implementare un semplicissimo sketch (visibile nel **Listato 1**) che attivi un I²C-Bus. Ad ogni interrogazione, il DS1307 risponde inviando all'ATmega (sempre lungo il suo bus I²C) la risposta e le informazioni su ora e data.
Il modulo RTC, che deve essere alimentato con una tensione continua ben stabilizzata del valore di 5 volt, assorbe una corrente (davvero esigua) dell'ordine di 1,5 milliampere, che scende a 500 nA nel funzionamento a pila.

LA COSTRUZIONE

Il modulo RTC si realizza su una basetta ramata monofaccia

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

Organizzazione del registro del DS1307 che contiene i dati orari e formato utilizzato per memorizzare i dati stessi.

Listato 1

```
// Date and time functions using a DS1307 RTC connected via I2C and Wire lib

#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 RTC;

void setup () {
    Serial.begin(57600);
    Wire.begin();
    RTC.begin();
    RTC.sqw(1);           //0 Led off - 1 Freq 1Hz - 2 Freq 4096kHz -
                          //3 Freq 8192kHz - 4 Freq 32768kHz
  if (! RTC.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was compiled
    RTC.adjust(DateTime(__DATE__, __TIME__));
  }
}

void loop () {
    DateTime now = RTC.now();

    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

    Serial.print(" since midnight 1/1/1970 = ");
    Serial.print(now.unixtime());
    Serial.print("s = ");
    Serial.print(now.unixtime() / 86400L);
    Serial.println("d");

    // calculate a date which is 7 days and 30 seconds into the future
    DateTime future (now.unixtime() + 7 * 86400L + 30);

    Serial.print(" now + 7d + 30s: ");
    Serial.print(future.year(), DEC);
    Serial.print('/');
    Serial.print(future.month(), DEC);
    Serial.print('/');
    Serial.print(future.day(), DEC);
    Serial.print(' ');
    Serial.print(future.hour(), DEC);
    Serial.print(':');
    Serial.print(future.minute(), DEC);
    Serial.print(':');
    Serial.print(future.second(), DEC);
    Serial.println();

    Serial.println();
    delay(3000);
}
```

facilmente ottenibile per fotoincisione a partire dalla traccia lato rame scaricabile dal nostro sito *www.futurashop.it*. Ottenuto lo stampato, bisogna disporvi i pochi componenti occorrenti iniziando con le resistenze e procedendo con lo zoccolo per il DS1307 ed il quarzo, cilindrico, che monterete sdraiato per occupare meno spazio in altezza. Terminate con il LED, il condensatore ed il portapila; per il montaggio su Arduino dovete montare i soliti connettori SIL aventi i piedini lunghi almeno 20 mm: ne servono due da sei contatti ciascuno. A fine montaggio, inserite pure il DS1307 nel proprio zoccolo, ricordando che va orientato con il riferimento rivolto al lato esterno dello stampato.

Affinché Arduino possa interagire con il modulo RTC, bisogna caricarvi lo sketch appositamente preparato, visibile nel **Listato 1**; la libreria che abbiamo utilizzato è quella di *ladyada*, scaricabile dal sito *www.ladyada.net/learn/breakoutplus/ds1307rtc.html*. Rispetto alla versione originale di questo sito web, abbiamo apportato alcune modifiche per poter gestire anche la frequenza di lampeggio del LED posto sullo shield, comandato dall'uscita SQW; per l'esattezza abbiamo aggiunto il comando RTC.sqw(*x*) dove al posto della *x* bisogna scrivere:

- 0 per disattivare il LED (sempre spento);
- 1 per il lampeggio del LED ad 1Hz;
- 2 per far pulsare l'uscita a 4.096 kHz;
- 3 per ottenere dall'uscita 8.192 kHz;
- 4 per ottenere dall'uscita 32.768 kHz.

Resta inteso che oltre 1 Hz, il LED si vedrà sempre acceso e l'uscita SQW potrà essere utilizzata come clock per determinate applicazioni o altre unità. La libreria che serve è disponibile insieme ai file del progetto nel nostro sito *www.elettronicain.it* sotto forma di cartella compressa chiamata *RTCLib.zip*, contenente anche alcuni esempi di gestione.

Futura Elettronica
Via Adige 11
21013 - Gallarate (VA)