# RTC SHIELD PER ARDUINO- IN KIT

Prezzo: 9.02 €

Tasse: 1.98 €

Prezzo totale (con tasse): 11.00 €

Basata sull'integrato DS1307 della Maxim-Dallas, questa shield per Arduino consente di avere un preciso orologio di sistema, sgravando la CPU dal calcolo e dalla gestione dei dati orari e liberando spazio nella memoria di programma, che può così essere utilizzata per scrivere il codice di altre applicazioni. L'integrato DS1307 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.

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 al microcontrollore di Arduino mediante l'I²C-bus. Il bus fa capo ai piedini 5 (SDA) e 6 (SCL) che combaciano con i pin SCL e SDA di Arduino (UnoRev3 e MegaRev3), il quale fa da unità master dell'I²C-Bus, mentre il DS1307 è lo slave.

Per le versioni di Arduino precedenti ad Arduino Uno Rev3 (UnoRev2 o Arduino2009), i segnali SCL e SDA sono disponibili sui pin A4 e A5, per rendere la shield compatibile anche con queste versioni, sul lato rame sono presenti delle piazzole a saldare. Per conoscere l'ora e la data, Arduino deve interrogare il DS1307 mediante l'I²C-bus; allo scopo occorre implementare un semplicissimo sketch 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.

Sui connettori sono disponibili l'RST, i 5V, la massa e gli ingressi analogici A0÷A5. Nella shield l'uscita SQW (SQUARE WAVE) pilota in modo sink un LED, facendolo pulsare alla stessa frequenza dell'onda quadra prodotta; inoltre, tramite il ponticello J1 possiamo decidere se far leggere o no ad Arduino, tramite la linea A3, il segnale corrispondente. Alimentazione: 5 Vdc (ben stabilizzati), assorbimento: 1,5 mA (che scende a 500 nA nel funzionamento a pila). Dimensioni (mm): 55x51x15.

La confezione comprende anche la batteria CR2032.

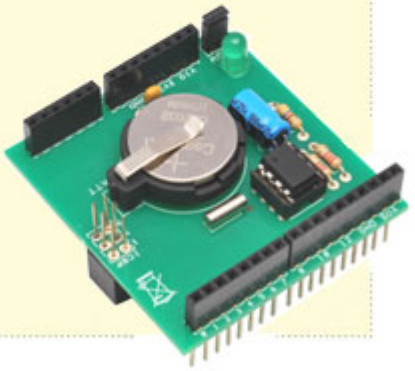
## Lo Sketch (LISTATO 1)

Affinchè Arduino possa interagire con il modulo RTC, bisogna caricarvi lo sketch appositamente preparato, visibile nel Listato 1. Come libreria abbiamo utilizzato quella di ***ladyada***, opportunamente modificata (scaricabile qui ***RTCLib.zip***) 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 1 Hz
- 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

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);
}
```



## SQW/OUT (SQUARE WAVE/OUTPUT DRIVER)

### SQUAREWAVE OUTPUT FREQUENCY

| RS1 | RS0 | SQW OUTPUT FREQUENCY |
|---|---|---|
| 0 | 0 | 1Hz |
| 0 | 1 | 4.096kHz |
| 1 | 0 | 8.192kHz |
| 1 | 1 | 32.768kHz |

Il DS1307 dispone di un’uscita di clock programmabile che 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 - Square Wave Enable, 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 alla stessa frequenza dell’onda quadra prodotta; inoltre, tramite il ponticello J1 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.

## Schema a Blocchi dell’integrato DS1307