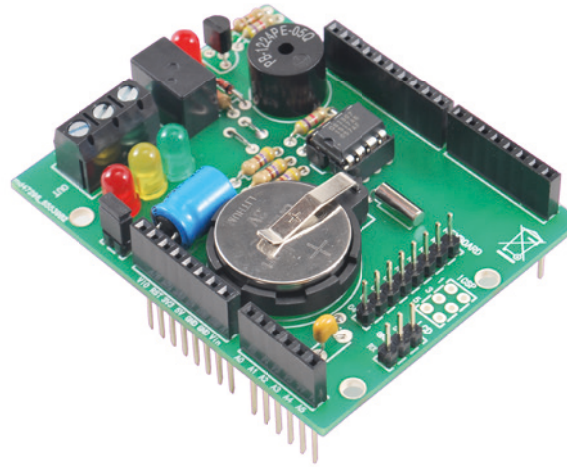
Shield codice OTP attivatore

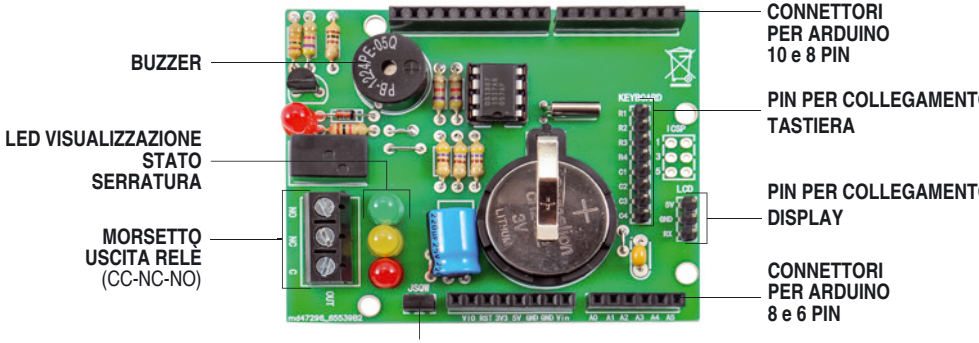
(cod. FT1234K)



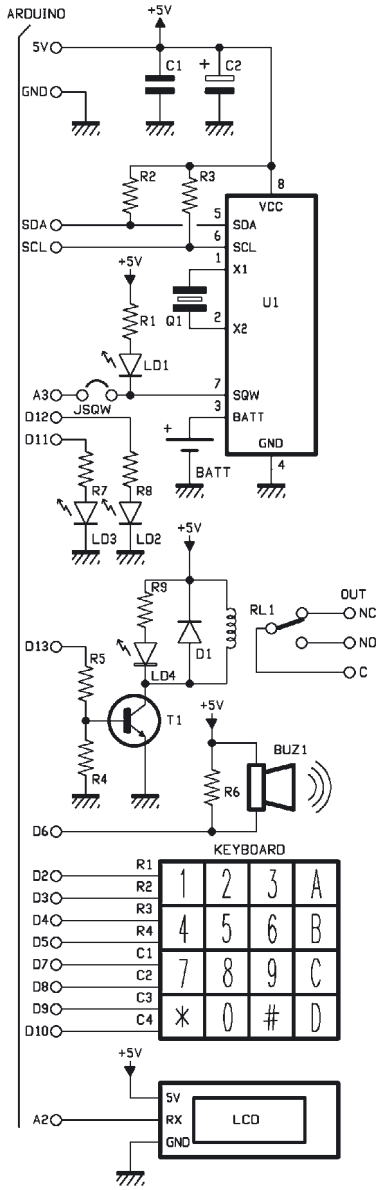
Shield per Arduino che permette di realizzare una serratura a comando elettrico che si apre grazie ad un codice, sempre diverso, generato da un'applicazione da installare sullo smartphone sulla base di un algoritmo che tiene conto dell'orario. Lo shield dispone di un relé per attivare un carico esterno (ad esempio una elettroserratura) se il codice inserito è corretto; un buzzer per generare segnalazioni acustiche; due LED (uno rosso e l'altro verde) per visualizzare lo stato della serratura; un Real Time Clock (con l'integrato DS1307 della Maxim/Dallas) per avere sempre a disposizione un valore di timestamp aggiornato e corretto; una batteria tampone per alimentare l'RTC anche se la scheda non è alimentata. Allo shield andrà collegato un tastierino alfanumerico a membrana per l'inserimento dei codici; un display seriale per visualizzare lo stato della serratura e i codici inseriti.

L'hardware

Il Real Time Clock montato sullo shield è un integrato DS1307 della Maxim/Dallas e si avvale di un quarzo (Q1) da 32.768 kHz che fornisce la frequenza base per l'oscillatore interno; l'integrato è un contatore BCD 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



Schema elettrico



al microcontrollore di Arduino mediante l'I²C-bus del quale il chip è provvisto; Arduino fa da unità master dell'I²C-Bus, mentre il chip della Maxim-Dallas è lo slave.

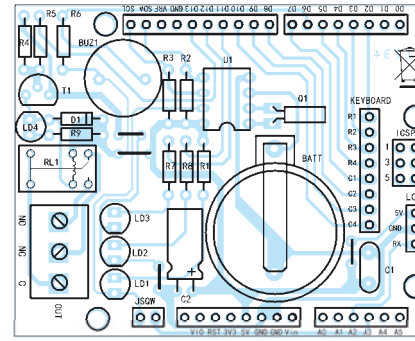
Oltre a ciò, il DS1307 dispone di un'uscita di clock programmabile. La condizione dell'uscita di clock ausiliario (SQWE, piedino 7) si definisce impostando opportunamente lo stato logico dei bit RS0 (0) ed RS1 (1) del registro di controllo; ad esempio, 1 Hz si ottiene con entrambi i bit a zero. Si noti 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. Nello shield l'uscita SQW pilota in modo sink un LED, facendolo pulsare 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, il tutto senza impegnare Arduino in routine di temporizzazione.

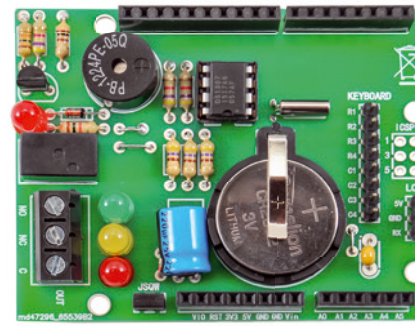
Il DS1307 dispone anche di un circuito di controllo in grado di verificare la mancanza o l'insufficienza dell'alimentazione principale (Vcc) e fare in modo che la tensione occorrente al proprio funzionamento venga prelevata dalla batteria di backup, nel nostro caso una pila a bottone CR2032 da 3 V, che 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

Piano di montaggio

Elenco Componenti:

R1: 470 ohm
R2: 4,7 kohm
R3: 4,7 kohm
R4: 10 kohm
R5: 4,7 kohm
R6: 100 ohm
R7: 470 ohm
R8: 470 ohm
R9: 1 kohm
C1: 100 nF ceramico
C2: 220 μF 16 VL elettrolitico
D1: 1N4148
LD1: LED 5mm verde
LD2: LED 5mm giallo
LD3, LD4: LED 5mm rosso
T1: BC547
RL1: Relé minaitura 5V
BUZ1: Buzzer senza elettronica
Q1: Quarzo 32,768 kHz
BATT: Porta batteria CR2032
U1: DS1307
LCD: Display LCD seriale





Varie:
- Morsetto 3 vie
- Strip maschio 2 vie
- Strip maschio 3 vie
- Strip maschio 8 vie
- Strip Maschio/Femmina 6 vie
- Strip Maschio/Femmina 8 vie (2 pz.)
- Strip Maschio/Femmina 10 vie
- Strip Maschio/Femmina 2x3 vie
- Zoccolo 4+4
- Jumper
- Batteria CR2032
- Tastiera a membrana 4x4
- Circuito stampato S1234

dell'alimentazione è dimensionata in modo da svolgere due compiti: preservare la memoria in cui sono contenuti i dati sull'ora e la data attuali e mantenere attivo il contatore dell'orologio. Il primo viene svolto conservando nella RAM non volatile le informazioni corrispondenti, mentre il secondo è espletato dall'alimentazione di riserva presa dalla pila o batteria tampone collegata fra il piedino 3 e il 4. La sezione di controllo dell'alimentazione interviene sulle altre funzioni del DS1307, bloccando la comunicazione con l'I²C-bus e la generazione dell'eventuale segnale di clock ausiliario SQW, in modo da minimizzare il consumo di energia ed estendere quanto più possibile l'autonomia.

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.
Il modulo RTC, che deve essere alimentato con una tensione continua ben stabilizzata del valore di 5 volt, assorbe una corrente dell'ordine di 1,5 milliampere, che scende a 500 nA quando ad alimentarlo è la pila. I 5 volt vengono prelevati dall'omonima linea di Arduino (5V).
Per conoscere l'ora e la data, Arduino deve interrogare il DS1307 mediante l'I²C-bus di cui il chip è provvisto; ad ogni interrogazione, il DS1307 risponde inviando all'ATmega (sempre lungo il suo bus I²C) la risposta e le informazioni su ora e data. Nel nostro shield l'RTC è quindi collegato ai due pin SDA e SCL di Arduino; le due resistenze da 1 kohm R2 ed R3 fungono da resistenze di pull-up per il bus I²C, che serve al microcontrollore ATmega di Arduino a interrogare il DS1307 per chiedergli l'orario e leggerlo.
Il relé è comandato dal pin 13 di Arduino tramite il transistor T1; mentre il diodo D1 serve a sopprimere la tensione inversa che la bobina del relé genera quando il transistor va in interdizione.
Il LED LD4, collegato con la sua resistenza in serie e a sua volta in parallelo alla bobina del relé, si accende quando RL1 è attivo. Due ulteriori LED, uno rosso e uno giallo, sono collegati ai pin 11 (LD3) e 12 (LD2) di Arduino tramite due resistenze di limitazione della corrente. Il cicalino viene pilotato dal pin 6 di Arduino.
Sullo shield sono presenti due connettori a strip: uno a 8 pin cui collegare la tastiera a membrana e uno a 3 pin a cui collegare il display LCD a interfaccia seriale. La morsettiera tripolare consente invece di collegare un carico esterno al relé: il contatto centrale della morsettiera è il normalmente chiuso (NC), mentre quelli laterali sono il contatto normalmente aperto (NO) e quello comune (C).
È importante ricordare che il relé montato sullo shield permette di gestire utilizzatori che assorbano fino a 1 ampere in circuiti funzionanti al massimo a 60 Vcc: quindi elettroserrature che tipicamente vanno a 12/24 Vcc o ca. Se si intende gestire carichi di maggior potenza è necessario utilizzare un servo relé di cui alimentare la bobina attraverso lo scambio C/NO del relé posto sullo shield, che verrà utilizzato quindi come interruttore. L'isolamento galvanico sarà assicurato dal servo relé.

Lo sketch

Lo sketch da installare in Arduino è scaricabile dalla scheda on-line del prodotto. Per la sua compilazione, è necessario installare tre librerie:

- keypad;
- RTClib;
- TOTP Library.

Tutte le librerie posso essere installate tramite il Library Manager incluso nelle versioni più recenti dell'IDE di Arduino; dal menu Sketch impartiamo il comando *Include Library –> Manage Libraries*. Inserire il nome della libreria nel campo di ricerca e, una volta trovata, cliccare sul pulsante Install come indicato in **Fig. 2**.

Setup iniziale

Per poter utilizzare la serratura, per prima cosa è necessario installare sul proprio smartphone un'applicazione in grado di generare codici OTP con l'algoritmo scelto.
Per iOS ed Android è possibile installare dal relativo market l'applicazione Google Authenticator, mentre per Windows Phone si può utilizzare Authenticator+. Tutte le applicazioni sono gratuite; ecco di seguito gli indirizzi web:
• Google Authenticator per iOS: *https://itunes.apple.com/it/app/google-authenticator/id388497605?mt=8*;
• Google Authenticator per Android: https*://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl=it*;
• Authenticator+ per Windows Phone: *https://www.microsoft.com/en-us/store/apps/authenticator/9nblggh08h54*.

Fig. 2

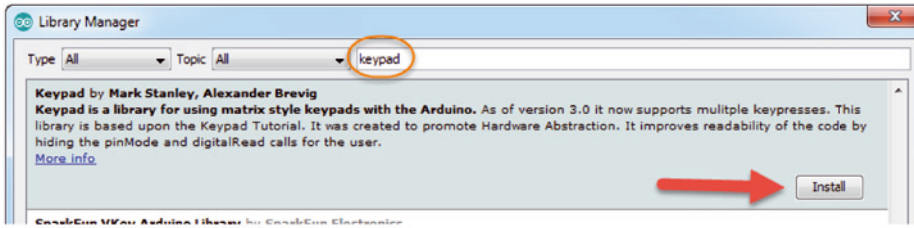


Fig. 3

Fig. 4

È necessario che applicazione e sketch su Arduino condividano la medesima chiave privata per poter generare gli stessi codici; per semplificare la configurazione abbiamo preparato un'applicazione web all'indirizzo *http://www.lucadentella.it/OTP/*.
Dopo aver inserito una password di dieci caratteri a piacere, premere il pulsante GO: apparirà l'array da inserire in Arduino e codice o QRCode per configurare l'app. È anche possibile cambiare il nome dell'account che sarà visualizzato dall'app al di sotto dei codici generati (**Fig. 3**). Il valore del campo Arduino HEX array va riportato nel file CONFIG.h dello sketch (**Fig. 4**). Mentre per configurare l'app sarà possibile –dal menu di aggiunta nuovo account– decidere di inserire manualmente il codice o di inquadrare con la fotocamera del telefono il QRCode per una configurazione automatica (**Fig. 5**). In entrambi i casi, una volta ultimata la configurazione l'applicazione inizierà a generare i codici (**Fig. 6**). Dobbiamo infine verificare l'orario presente sull'RTC, ricordando che deve essere quanto più possibile sincronizzato con quello dello smartphone e secondo il fuso orario UTC (non è invece necessario che lo smartphone sia configurato su tale fuso orario perché è l'applicazione stessa a fare la corretta conversione rispetto all'ora locale). Lo sketch consente di

Fig. 5

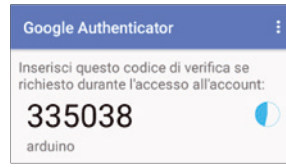


Fig. 6

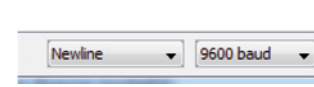


Fig. 7

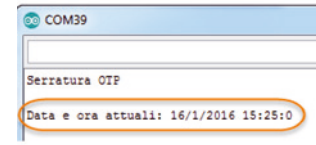


Fig. 8

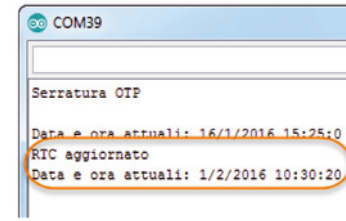


Fig. 9

configurare l'RTC tramite comandi seriali: apriamo il Serial Monitor e configuriamolo per 9.600 baud con Newline come line ending (**Fig. 7**). Se inviamo il carattere "?", lo sketch determinerà la visualizzazione dell'orario corrente (**Fig. 8**). Se tale orario non è corretto, è possibile aggiornarlo inviando il comando **!ggMMaahhmmss**. Ad esempio per configurare il giorno 01/02/2016 alle 10:30:20 il comando da inviare sarà **!010216103020**. Se il comando è corretto, Arduino risponderà con la conferma della configurazione e visualizzerà il nuovo orario (**Fig. 9**).

Configurazione serratura

Lo sketch scritto per il progetto consente alcune personalizzazioni, effettuabili modificando il file CONFIG.h. (**listato 1**). Per prima cosa, è possibile

modificare il comportamento della serratura scegliendo tra due diverse modalità alternative l'una all'altra:
• MODO_IMPULSO = la serratura rimane aperta per tot millisecondi e poi si chiude automaticamente;
• MODO_STABILE = la serratura rimane aperta fino alla pressione di un tasto.
Se è stata scelta la modalità IMPULSO, è possibile cambiare la costante TEMPO_APERTURA per definire il tempo (specificare il numero di millisecondi) in cui la serratura rimarrà aperta a seguito di ogni attivazione del relé. Se invece si è scelta la modalità STABILE, è possibile cambiare la costante TASTO_CHIUSURA per scegliere quale tasto sarà utilizzato per chiudere la serratura (molto importante è che il tasto scelto sia sempre indicato tra apici singoli!).

Infine è possibile disattivare il suono del buzzer commentando la riga #define BUZZER_ON.

L'articolo completo del progetto è stato pubblicati su: Elettronica In n. 203.

Listato 1

```
// chiave privata per la generazione dei codici OTP
uint8_t hmacKey[] = {0x65, 0x6c, 0x65, 0x74, 0x74, 0x72, 0x6f, 0x2d, 0x69, 0x6e};

// buzzer on/off (commentare la riga sottostante per disabilitarlo
#define BUZZER_ON

// modalità di funzionamento della serratura
// MODO_IMPULSO = se il codice è ok, la serratura rimane aperta per tot
// millisecondi (definito sotto)
// MODO_STABILE = se il codice è ok, la serratura rimane aperta fino alla
// pressione di un tasto (definito sotto)
#define MODO_IMPULSO  0
#define MODO_STABILE  1
byte modoSerratura = MODO_IMPULSO;

// tempo di apertura della serratura (in millisecondi) se MODO_IMPULSO
#define TEMPO_APERTURA 2000

// tasto per chiudere la serratura se MODO_STABILE
#define TASTO_CHIUSURA 'D'

// ----- normalmente non è necessario modificare nulla sotto questa riga -----
// definizione dei PIN
#define PIN_BUZZER      6
#define PIN_LEDROSSO    11
#define PIN_LEDVERDE    12
#define PIN_RELAY       13

// configurazione della tastiera
#define KEY_ROWS        4
#define KEY_COLUMNS     4
char keys[KEY_ROWS][KEY_COLUMNS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[KEY_ROWS] = {2,3,4,5};
byte colPins[KEY_COLUMNS] = {7,8,9,10};
```

A tutti i residenti nell'Unione Europea. Importanti informazioni ambientali relative a questo prodotto

Questo simbolo riportato sul prodotto o sull'imballaggio, indica che è vietato smaltire il prodotto nell'ambiente al termine del suo ciclo vitale in quanto può essere nocivo per l'ambiente stesso. Non smaltire il prodotto (o le pile, se utilizzate) come rifiuto urbano indifferenziato; dovrebbe essere smaltito da un'impresa specializzata nel riciclaggio. Per informazioni più dettagliate circa il riciclaggio di questo prodotto, contattare l'ufficio comunale, il servizio locale di smaltimento rifiuti oppure il negozio presso il quale è stato effettuato l'acquisto.

Prodotto e distribuito da:
FUTURA GROUP SRL
Via Adige, 11 - 21013
Gallarate (VA)
Tel. 0331-799775
Fax. 0331-778112
Web site: *www.futurashop.it*
Info tecniche: www.futurashop.it/Assistenza-Tecnica