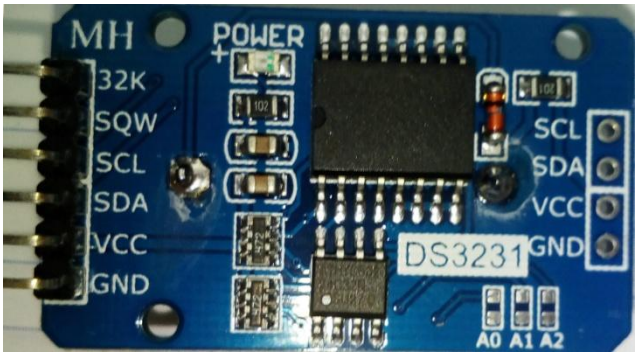
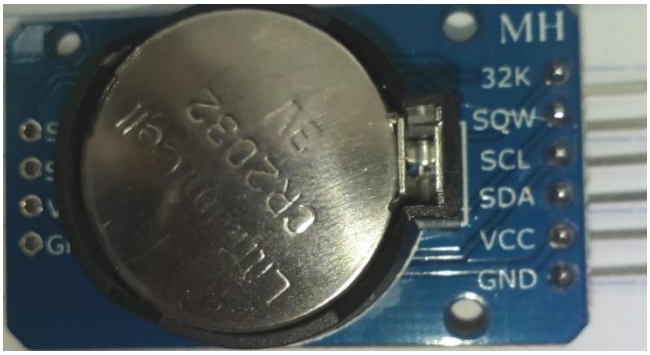
44 - RTC DS3231- orologio digitale - digital clock (see notes at end of this section)





Arduino e' uno smemorato. Se si toglie l'alimentazione mantiene memorizzato il programma, ma perde ogni altra informazione acquisita nel corso di ogni sua precedente attivita'. Tra le informazioni "volatili" c'e' anche la data e l'ora. Per superare quest'ultimo problema e' possibile utilizzare un modulo RTC (Real Time Clock) e cioe' un orologio che tiene aggiornata e memorizza data ed ora in sua vece. Il modulo RTC DS3231 rappresentato in figura e' dotato di una sua autonoma fonte di energia (una pila del tipo CR2032 da 3 volt) ed e' in grado di fornire informazioni su data, ora e temperatura ambientale.

Grazie alla pila di cui e' dotato, il modulo, una volta istruito con data ed ora corrente, continua a segnare il tempo, anche quando non e' in uso o e' addirittura tolto dal circuito.

E' anche in grado di gestire due allarmi e quindi di fornire ad Arduino un segnale digitale nel momento in cui matura il momento di un allarme. Questo segnale puo' essere utilizzato per attivare eventuali funzioni a tempo o, addirittura, per risvegliare Arduino dallo stato di "sleep" in cui puo' essere messo al fine di limitare il consumo di energia.

Il modulo fornisce le informazioni attraverso il bus SDA/SCL, il cui protocollo e' gestito dalle porte A4 ed A5 e, in replica, nelle porte a fianco della porta AREF di Arduino

In questo esercizio non ci limiteremo ad utilizzare il modulo RTC DS3231, ma utilizzeremo anche un display lcd gestito da un driver I2C e quindi avremo contemporaneamente in funzione due componenti che utilizzano il bus SDA/SCL. Per comodita' di assemblaggio il modulo RTC DS3231 e' stato collegato alle porte A4 ed A5 mentre il display lcd e' stato collegato alle due porte a fianco del porta AREF (come, con un po' di difficolta' si puo' anche evincere dalla figura qua' sopra).

Il programma presentato in questo esercizio e' anche in grado di "settare" il modulo RTC con data ed ora correnti. Per farlo e' sufficiente resettare Arduino, aprire il monitor seriale ed inserire i dati tramite la tastiera del pc (con Arduino ovviamente collegato al pc tramite cavo USB) nella forma:

aa,mm,gg,hh,mm,ss

dove:

aa	anno (ultimi due caratteri (e quindi 16 per indicare 2016)
mm	mese (da 1 a 12)
gg	giorno (da 1 a 31)
hh	ora (da 00 a 23)
mm	minuto (da 00 a 59)

ss secondo (da 00 a 59)

Prima di procedere alla compilazione del programma e' necessario installare, se non gia' fatto, le seguenti librerie:

- timeLib.h reperibile qui;
- DS3232RTC.h reperibile qui'.
- LiquidCrystal_I2C.h reperibile qui'.

Per le modalita' di installazione si rimanda all'iter gia' illustrato nei precedenti esercizi e comunque riassumibile in:

- download delle librerie in forma compressa (normalmente ogni sito che tratta una libreria ha un punto dal quale si puo' attivare il download);
- installazione via IDE->sketch->include library->add .zip library
- verifica di avvenuta installazione (la libreria deve essere nell'elenco presente in IDE->sketch->include library->contributed library)

Il filmato di questo esercizio e' reperibile qui'.

Nota: Questo esercizio e questa nota sono parte di una serie che vede protagonisti Arduino ed alcuni dei componenti ad esso collegabili. Per la maggior parte degli esercizi e' anche disponibile un filmato su youtube.

- Esercizi facenti parte della raccolta
- Filmati presenti su youtube
- Informazioni su arduino e sui componenti collegabili (PDF scaricato nell'area di download)
- Breve manuale di programmazione (PDF scaricato nell'area di download)

Per eventuali chiarimenti o suggerimenti sul contenuto di questa scheda scrivere a giocarduino@libero.it

Here some notes about this project, translated by google translator

Arduino is a dazed. If power is off, it keeps the program, but loses any other information generated in the previous activity. Among the lost information there are also date and time. To overcome this problem we can use a RTC (Real Time Clock), an external watch that maintain date and time in his stead.

The DS3231 module shown in the figure is equipped with an autonomous power source (a battery CR2032 3v type) and can provide information about date, time and ambient temperature. Once trained with current date and time, continue to mark time, even when not in use or even removed from the circuit.

It also can handle two alarms, and then to provide, at Arduino, a digital signal when matures an alarm. This signal can be used to start a scheduled activity or, even, to awaken Arduino from the state of "sleep" in which can be put, in order to save energy.
The module provides information via "SDA/SCL" bus, whose protocol runs on A4 and A5 pins and on pins near (on the left) the AREF pin.

This project will not only uses a DS3231 device, but also uses an LCD display, both controlled by an I2C driver, that uses "SDA/SCL" bus.

The program in this example is also able to set the RTC module with current date and time. To do that is sufficient reset Arduino, open the serial monitor and enter the data via the PC keyboard (obviously connected via USB cable) in the form: **yy,mm,dd,hh,mm,ss**

Before proceeding to program compilation must be installed, if not already done, the libraries:

- ds3232rtc.h found here.
- timelib.h found here
- LiquidCrystal_I2C.h found here

For library installation, see process shown in previous examples, and summarized in:

- library download in compressed form;
- Installation via IDE-> sketch-> includes Library-> add .zip library
- After installation need verification: the library must be present in IDE-> sketch-> includes Library-> Contributed library

Note: This project and this note is part of a series that sees, as main characters, Arduino and some of connectable components. For most projects there is also a video on youtube.

- Projects collection
- Movies on youtube
- About Arduino and components (italian; pdf will be downloaded in your download area
- Quick programming guide (almost english; pdf will be downloaded in your download area)

For any questions or suggestions about this note (and on its english translation), please write to giocarduino@libero.it (simple words and short sentences, please)

Materiali

- Un modulo RTC DS3231
- Un display lcd con driver I2C (non obbligatorio – not mandatory)
- Un po' di cavetteria

Nota: il display lcd non e' indispensabile e puo' essere sostituito dal monitor seriale. In questo caso e' necessario:

- modificare il prefisso "lcd." con il prefisso "Serial." in ogni istruzione "lcd.print"
- sostituire ogni istruzione "lcd.clear (0, 0)" oppure "lcd.setCursor (0, 1)" con "Serial.println (" ");"
- eliminare le seguenti istruzioni:

```
#include <LiquidCrystal_I2C.h>  // libreria di gestione del display lcd
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
```

- eliminare ogni altra istruzione residua, avente il prefisso "lcd"

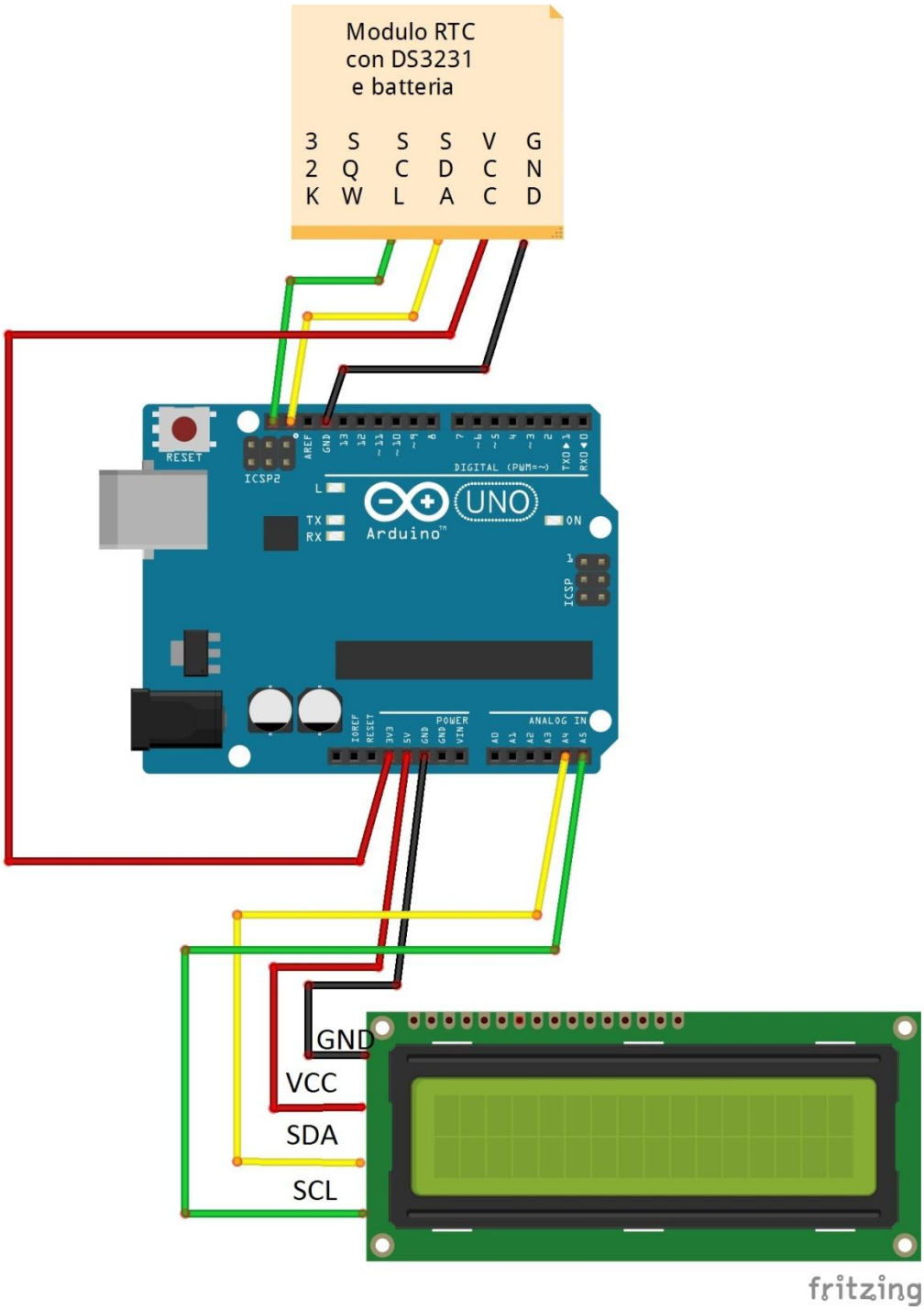
Note: The LCD display is not mandatory and can be replaced by serial monitor. In this case:

- change each "lcd" prefix in "Serial" prefix, in each statement "lcd.print"
- replace each instruction "lcd.clear ()" or "lcd.setCursor (0, 1)" with "Serial.println (" ");"
- delete:

```
#include <LiquidCrystal_I2C.h>  // libreria di gestione del display lcd
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
```

- delete any other statement which prefix is"lcd"

Schema



Programma

```
/*  Attenzione: facendo il copia/incolla dal PDF all'IDE si perde la formattazione del
 *  testo. Per rendere piu' facilmente leggibile il programma e' opportuno formattarlo
 *  subito dopo il trasferimento nell'IDE, premendo CTRL+T.
 *
 *     Programma di utilizzo del modulo RTC DS3231
 *
 *  Questo programma sincronizza il timer interno di arduino con la
 *  data e l'ora fornita da un orologio esterno (un modulo RTC con
 *  DS3231) ed espone data, ora e temperatura su di un display 1620
```

```
 *  governato da un circuito I2C
 *
 *  Prima di procedere con questo esercizio e' necessario, se non
 *  gia' fatto, caricare sul modulo RTC la data e l'ora corrente
 *  utilizzando la tastiera del pc collegato ad arduino via cavo USB.
 *
 *  I dati devono essere inseriti sottoforma di stringa cosi' formata:
 *
 *                  aa,mm,gg,hh,mm,ss
 *  dove
 *     aa  anno, sottoforma di due caratteri (2016 = 16)
 *     mm  mese, da 1 a 12
 *     gg  giorno, da 1 a 31
 *     hh  ora, da 00 a 23
 *     mm  minuti, da 0 a 59
 *     ss  secondi, da 0 a 59
 *
 *  Nel momento in cui viene immessa la stringa (nel momento in cui si
 *  preme enter), l'orologio inizia a computare il tempo partendo dai
 *  dati immessi e non avra' piu' bisogno di essere settato (a meno
 *  che non si scarichi la pila o la si tolga).
 *
 *  ATTENZIONE:
 *  il programma non esegue alcun controllo di validita' dei dati immessi,
 *  per cui se si inserscono dati errati o fuori dai limiti i risultati
 *  potrebbero essere imprevedibili.
 *
 * ----------------------------------------------------------------------------------------------
 * WARNING: cut&paste from PDF to IDE loses formatting. to restore it press CTRL + T.
 *
 *                      DS3231 Real time clock
 *
 * This program synchronizes the arduino's internal timer with the date and time provided by an
 * external clock (RTC DS3231) and displays the date, time and temperature on a 1620 lcd display,
 * managed by an I2C circuit.
 *
 * Before proceeding with this program it's necessary, if not already done, upload the RTC
 * appliance with current date and time, using the PC keyboard connected to Arduino via USB cable.
 * Data must be entered in the form: yy, mm, dd, hh, mm, ss
 *
 * The moment you enter the string (when you press enter), the clock begins work and does not have
 * needs to be set again (unless you drain or takes out battery).
 *
 * WARNING: the program does not perform any checks of validity on input data, so if you enter
 * incorrect data, results may be unpredictable.
 *
 *-----------------------------------------------------------------------------------------------
 */
#include <DS3232RTC.h>        //http://github.com/JChristensen/DS3232RTC
#include <TimeLib.h>          //http://playground.arduino.cc/Code/Time (presente in gestione
// librerie, ma da installare - already present in ide library managere, but not yet installed
#include <Wire.h>             //http://arduino.cc/en/Reference/Wire (vedi sopra - see above)
#include <LiquidCrystal_I2C.h>  // libreria di gestione del display lcd
//--------------------addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
//
time_t t;
tmElements_t tm;
//
float temperatura = 0;    // zona di memorizzazione della temperatura fornito da DS3231
float temp1       = 0;    // zona di memorizzazione della temperatura in gradi celsius
int digits        = 0;    // zona di memorizzazione dei minuti e dei secondi da esporre
//                           sul display lcd
char tabmesi [37] =  "genfebmaraprmaggiulugagosetottnovdic";  // tabella descrizione mesi
// above: italian month description, on three digits
int mese          = 0;    // zona di memorizzazione del mese corrente
int lavoro        = 0;    // zona di lavoro, utilizzata per calcoli intermedi
//
//********* routine di esposizione di data, ora e temperatura sul display lcd
//*************** time and temperature on lcd display **********************
//
void esponidati(void)
{
  // esposizione dei dati rilevati sul timer di arduino
  lcd.clear();
  lcd.print("   ");
  lcd.print(day());
  lcd.print(' ');
```

```
  mese = month();                          // memorizza il mese
  mese = mese - 1;                         // diminuisce di 1 il valore del mese poiche'
  //                                          la descrizione del primo mese (gen) inizia
  //                                          all'indirizzo zero della tabella mesi
  lcd.print (tabmesi [(mese * 3)]);        // espone la prima lettera del mese
  lcd.print (tabmesi [(mese * 3 + 1)]);    // espone la seconda lettera del mese
  lcd.print (tabmesi [(mese * 3 + 2)]);    // espone la terza lettera del mese
  lcd.print(' ');
  lcd.print(year());
  lcd.setCursor(0, 1);
  lcd.print(hour());
  digits = (minute());
  printDigits();                           // routine esposizione zeri non significativi
  digits = (second());
  printDigits();                           // routine esposizione zeri non significativi
  lcd.print (" t:");
  lcd.print (temp1);
}

//
//****routine visualizzazione minuti e secondi comprensivi di zeri non significativi*****
// ************* displays minutes and seconds, including non significant zeroes ***********
//
void printDigits()
{
  lcd.print(':');
  if (digits < 10)
    lcd.print('0');
  lcd.print(digits);
}

void setup(void)
{
  Serial.begin (9600);        // inizializza il monitor seriale
  lcd.begin(16, 2);           // inizializza il display e accende
  lcd.backlight();            // lo sfondo
  lcd.print (" buongiorno"); // goodday
  delay (1500);
  lcd.clear ();
  setSyncProvider(RTC.get);  // sincronizza il timer di Arduino con i dati presenti
  //                            sul modulo RTC
  lcd. clear ();
  if (timeStatus() != timeSet)  // verifica se la sincronizzazione e' ok
    lcd.print(" orologio non"); // "clock not"
  else
    lcd.print(" orologio");     // "clock"
  lcd.setCursor (0, 1);
  lcd.print  (" sincronizzato"); // " syncronized"
  delay (1500);
  temperatura = RTC.temperature();  // rileva per la prima volta la temperatura
}

void loop(void)
{
  if (Serial.available() == 17)   // controlla se e' in arrivo una stringa da 17
    //  caratteri (aa,mm,gg,hh,mm,ss)
  { // acquisisce la stringa con data ed ora
    int y = Serial.parseInt();
    tm.Year = y2kYearToTm(y);
    tm.Month = Serial.parseInt();
    tm.Day = Serial.parseInt();
    tm.Hour = Serial.parseInt();
    tm.Minute = Serial.parseInt();
    tm.Second = Serial.parseInt();
    t = makeTime(tm);
    RTC.set(t);          // aggiorna (setta) il modulo RTC
    setTime(t);          // aggiorna (setta) il modulo RTC
  }                      // fine del ciclo di acquisizione data ed ora

  digits = (second());   // verifica se sono passati 10 secondi dal precedente
  //                        rilevamento della temperatura
  lavoro = digits % 10;  // calcola il resto del secondo corrente diviso 10
  if (lavoro == 9)       // se si e' al nono secondo di ogni decade di minuto (se si e'
    //                      al secondo 9, 19, 29, 39, 49 o 59 di ogni minuto)
    //                      if we are at the 9,19,29... secondo of each minutes,
    temperatura = RTC.temperature();  // rileva la temperatura - catch temperature
  temp1 = temperatura / 4.0;          // trasforma la temperatura in gradi celsius
  esponidati();                       // lancia la routine di esposizione dei dati
```

```
  delay (1000);                     // aspetta un secondo prima di andare ad esporre
  //                                   nuovi dati, al fine di consentire al display
  //                                   una visualizzazione stabile
}
```