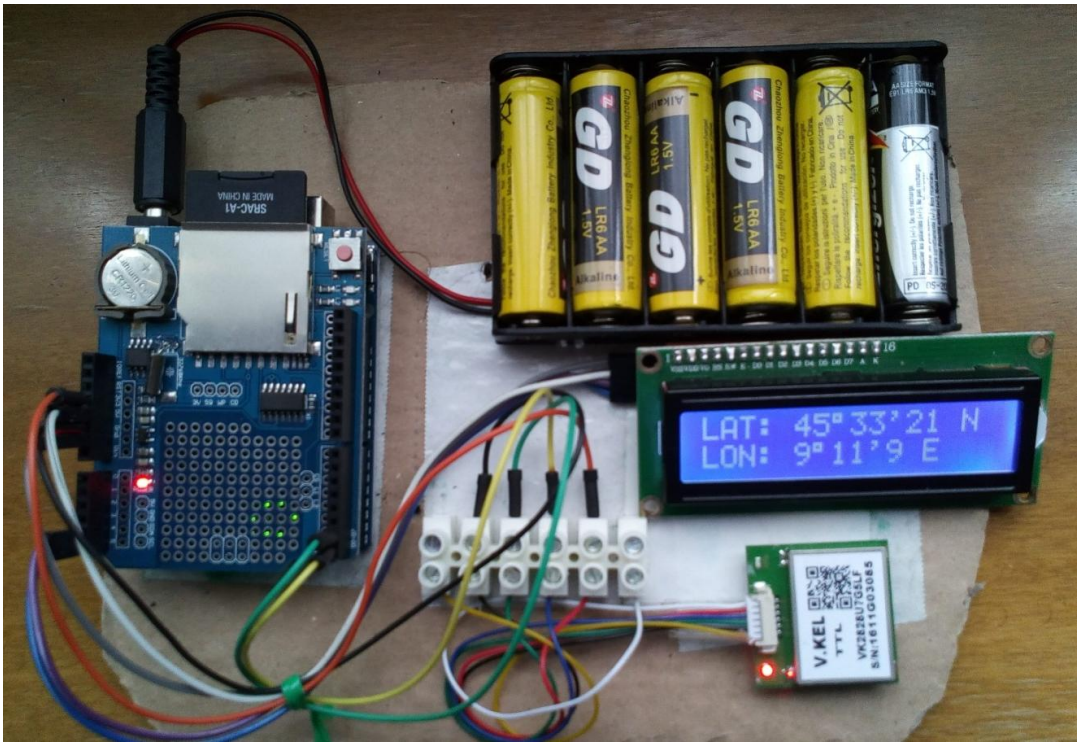
pS- tracciatore di percorso - path recorder (some notes at section end)



Un tracker e' un sistema che registra i dati un percorso al fine di consentirne poi la trasposizione su di una mappa. Sul web esistono svariati programmi di tracciatura che girano sui sistemi android (e quindi sui normali smartphone), per cui il sistema proposto in questa nota non e' particolarmente innovativo.

E' comunque un buon progetto, non troppo complicato ma interessante sia dal punto di vista dell'hardware che dal punto di vista del software. In questo progetto Arduino utilizza un rilevatore gps ed un data logger per fornire le principali funzioni offerte da un tracker.

Piu' in dettaglio il sistema rileva, ogni 3 secondi, la posizione geografica (latitudine longitudine) e la registra, in formato "gps traking key pro text", su di una scheda secure digital.

Una volta completato il percorso, il file presente sulla scheda puo' essere convertito, tramite l'applicazione "GPS babel" (gratuitamente reperibile sul web) in un file GPX e quindi utilizzato per disegnare il percorso su di una mappa, mediante l'uso di una delle numerose app, anch'esse facilmente e gratuitamente reperibili sul web.

Attenzione: il sistema gps utilizza le porte seriali (TX ed RX, collocate sui pin digitali 0 e 1) per comunicare i dati ad Arduino. Poiche' tali porte sono anche utilizzate dall'IDE per caricare il programma, al momento della compilazione sara' necessario togliere l'alimentazione all'antenna GPS, al fine di evitare conflitti.

Il "gps traking key pro text" e' un protocollo di registrazione di dati geografici abbastanza semplice da generare. Questa e' la sua struttura:

data *ora* *longitudine* *latitudine*
dd-mm-aaaa,hh:mm:ss,E/W gg°gm'gs",N/S gg°gm'gs"

N/S = N oppure S (nord oppure sud) a seconda che la latitudine sia positiva o negativa
E/W = E oppure W (est oppure ovest) a seconda che la longitudine sia positiva o negativa
gg = gradi
gm = minuti sessagesimali
gs = secondi sessagesimali

questo e' un esempio degli item di un valido file di tipo "gps traking key pro text":
06-01-2017,13:16:16,N 45°37'14",E 7°59'2",
06-01-2017,13:18:0,N 45°37'16",E 7°59'4",
06-01-2017,13:19:36,N 45°37'16",E 7°59'4",
06-01-2017,13:23:10,N 45°37'18",E 7°59'1",
06-01-2017,13:24:11,N 45°37'18",E 7°59'0",
06-01-2017,13:25:5,N 45°37'19",E 7°59'1",

Il programma registra su file la data e l'ora di Arduino che deve pero' essere preventivamente sincronizzata con la data e l'ora fornita da un sistema RTC. Il data logger utilizzato in questo progetto e' dotato di un sistema RTC gestito dal chip DS3231, le cui caratteristiche e le cui modalita' di settaggio sono descritte nella scheda tecnica del progetto 44 – orologio digitale, reperibile qui

E' quindi necessario procedere preventivamente alla regolazione della data e dell'ora del sistema RTC mediante il programma presente nella suddetta scheda tecnica in modo che Arduino, al momento dell'avvio si sincronizzi con la data e l'ora presente in detto sistema

Prima di procedere alla compilazione del programma devono essere installate, se non gia' presenti, le seguenti librerie:

- TinyGPS.h reperibile qui
- timeLib.h reperibile qui
- DS3232RTC.h reperibile qui
- LiquidCrystal_I2C.h reperibile qui

Per installare le librerie e' necessario seguire una procedura sintetizzabile in:

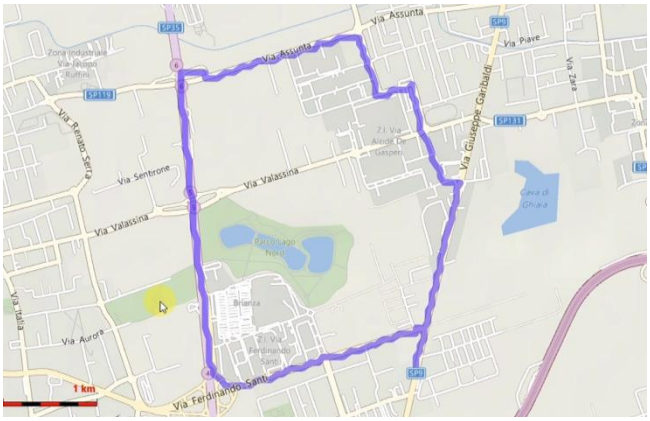
- download della libreria in formato compresso
- installare la nuova libreria andando in IDE-> sketch-> includes Library-> add .zip library
- verificare l'avvenuta installazione (andando in IDE-> sketch-> includes Library-> Contributed library)

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



A tracker is a system that records a path, to be transposed onto a map. On web there are several tracking programs, running on Android systems, so this project is not innovative.

It is, however, a good project, not too complicated but interesting from hardware and software point of view. In this project Arduino uses a gps antenna and a data logger to provide the main features offered by a tracker.

In detail, system detects, every 3 seconds, the geographic position (latitude and longitude) and registers it, as a "gps traking key pro text" format, on a SD card.

Once path is completed, file on SD card can be converted, using the "GPS babel" application (available for free on web) into a GPX file and then used to draw path on a map, by using a GPX file reader, which are freely available on web.

Warning: The gps antenna uses the serial bus (ports TX and RX, placed on 0 and 1 pins) to communicate data to Arduino. Since this bus is also used by IDE, during compilation you must power off the GPS antenna to avoid conflicts.

The "gps traking key pro text" is a fairly easy to generate geographic data logging protocol. This is his structure:

date *time* *longitude* *latitude*
dd-mm-aaaa,hh: mm: ss,E/Wgg °gm'gs", N/S gg°gm'gs",

N/S = N or S (north or south) depending on whether the latitude is positive or negative
E/W = E or W (east or west) depending on whether the longitude is positive or negative
gg = degrees
gm = sexagesimal minutes
gs = sexagesimal seconds

here some items of a valid “gps traking key pro text” file:
06-01-2017,13:16:16,N 45°37'14",E 7°59'2",
06-01-2017,13:18:0,N 45°37'16",E 7°59'4",
06-01-2017,13:19:36,N 45°37'16",E 7°59'4",
06-01-2017,13:23:10,N 45°37'18",E 7°59'1",
06-01-2017,13:24:11,N 45°37'18",E 7°59'0",
06-01-2017,13:25:5,N 45°37'19",E 7°59'1",

The program records on output file the Arduino's date and time, which must be previously synchronized with date and time provided by an RTC (Real Time Clock) system.

Data logger used in this project is equipped with an RTC system, managed by the DS3231 chip, whose features and set-up modes are described in the technical datasheet of the "44-digital clock" project, available here.

So you must proceed in advance to sets date and time in RTC system, by using program found in the above mentioned technical data sheet.

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

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

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

- library download in compressed form;
- Installation via IDE-> sketch-> includes Library-> add .zip library
- After installation please verify the library. It 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.

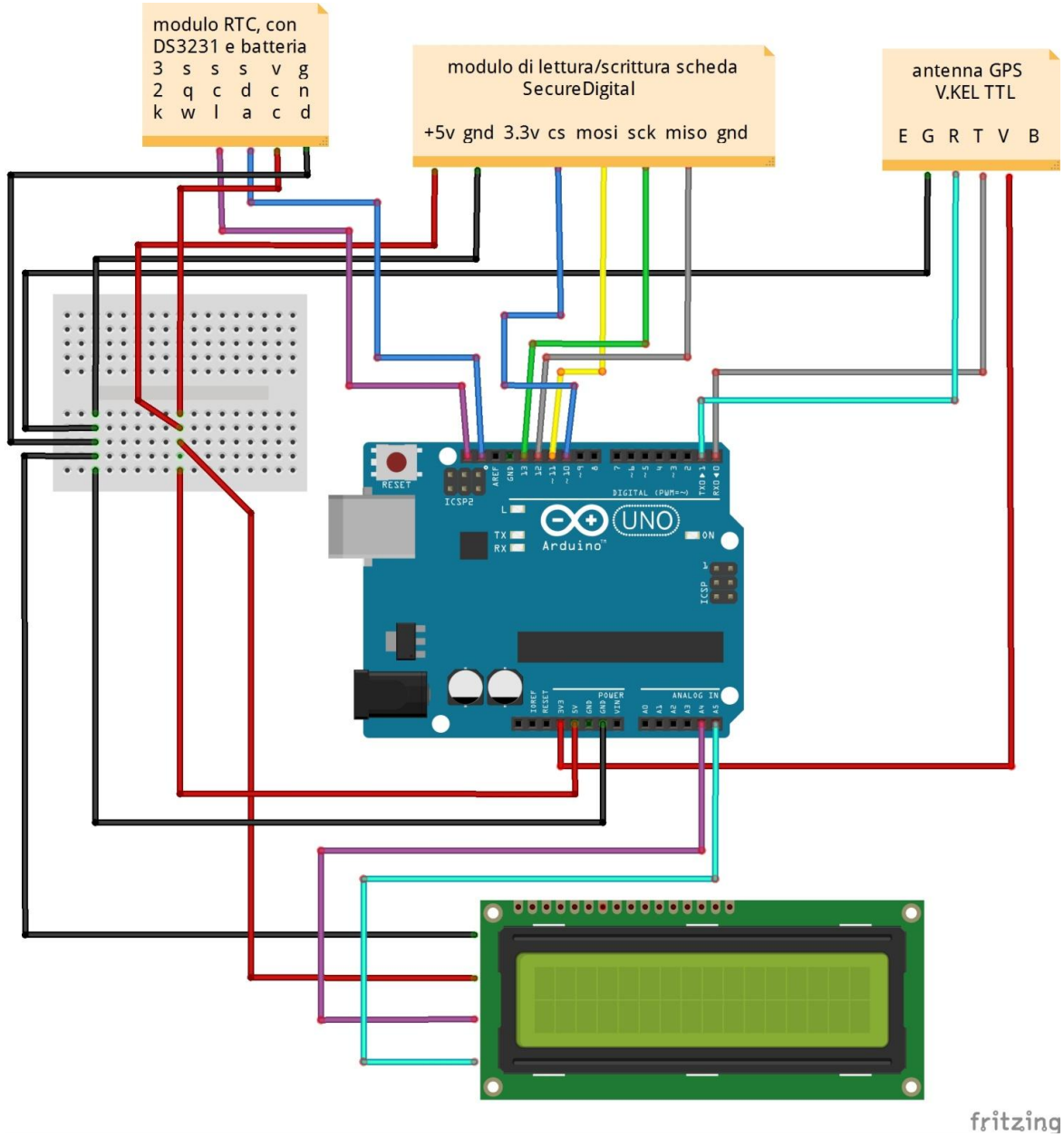
- Projects collection
- Movies on youtube
- About Arduino and components (italian; pdf will be downloaded in your download area
- Quick programming guide (almost english; pdf will be downloaded in your download area)

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 data logger con sistema RTC di tipo DS3231 (oppure un data logger e, separatamente) un sistema RTC di tipo DS3231
- Un'antenna GPS di tipo VK2828U7G5LF (attenzione: l'antenna GPS deve essere alimentata con una tensione di 3,3 volt)
- Un display lcd tipo 1602 con driver I2C
- Un po' di cavetteria

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.
  questo programma utilizza un'antenna GPS ed un data logger per registrare i dati del percorso

  Warning: cut&paste from PDF to IDE loses formatting. To restore it press CTRL + T.
  this program uses a GPS antenna and a Datalogger to records some path data

  Collegamento del data logger - datalogger connections
    ..........arduino uno.....Arduino mega
    GND ......non usato........non usato
    MISO .........12 ................50
    SCK  .........13 ................52
    MOSI .........11 ................51
    CS..... 10 ................53
    3.3v .....non usato ........non usato
    GND ..........GND ..............GND
    +5V ..........5v ................5v

  Collegamento del display lcd con driver I2C - LCD connections
    GND..........GND ..............GND
    VCC .........5v ................5v
    SDA .........A4 ................20
    SCL .........A5 ................21

  Collegamento dell'antenna GPS - GPS antenna connections
    G .........GND
    R .........digital pin 1
    T .........difital pin 0
    V .........3.3v
*/
#include <TinyGPS.h>           //https://github.com/mikalhart/TinyGPS/archive/v13.zip
#include <SPI.h>
#include <DS3232RTC.h>         //http://github.com/JChristensen/DS3232RTC
#include <TimeLib.h>           //http://playground.arduino.cc/Code/Time
#include <SD.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h> // https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
//---------------------addr  en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // definisce la tipologia del display
File filesd;
TinyGPS gps;
time_t t;
tmElements_t tm;
long lat;
long lon;
unsigned long fix_age;
int c;
int deglat;              // gradi sessagesimali latitudine
int minutilat;           // minuti sessagesimali latitudine
int secondilat;          // secondi sessagesimali latitudine
int deglon;              // gradi sessagesimali longitudine
int minutilon;           // minuti sessagesimali longitudine
int secondilon;          // secondi sessagesimali longitudine
char slat;               // segno della latitudine (E oppure w)
char slon;               // segno della longitudine (N oppure S)
float coordinata;        // usata nella routine di conversione gradi decimali in gradi sessagesimali
int gradi;               // usata nella routine di conversione gradi decimali in gradi sessagesimali
int minuti;              // usata nella routine di conversione gradi decimali in gradi sessagesimali
int secondi;             // usata nella routine di conversione gradi decimali in gradi sessagesimali
long tempocorrente = 0;
long tempoprec     = 0;
int  intervallo    = 3000; // intervallo di registrazione (una registrazione ogni tre secondi)
//
//********************** routine di scrittura su secure digital *************************
// ************************ writing on SD routine ************************************
//
void scrivi (void)
{
  filesd = SD.open("traccia.txt", FILE_WRITE); //File in scrittura
  if (filesd)                                  // Se il file è stato aperto correttamente
  {
    filesd.print (day());
    filesd.print('-');
```

```
    filesd.print (month());
    filesd.print('-');
    filesd.print(year());
    filesd.print (',');
    filesd.print(hour());
    filesd.print (':');
    filesd.print (minute ());
    filesd.print (':');
    filesd.print (second());
    filesd.print (',');
    filesd.print (slon);
    filesd.print (' ');
    filesd.print (deglon);
    filesd.write ('°');
    filesd.print (minutilon);
    filesd.print ("'");
    filesd.print (secondilon);
    filesd.print ('"');
    filesd.print (',');
    filesd.print (slat);
    filesd.print (' ');
    filesd.print (deglat);
    filesd.write ('°');
    filesd.print (minutilat);
    filesd.print ("'");
    filesd.print (secondilat);
    filesd.print ('"');
    filesd.print (",");
    filesd.println("");                 // Scrivo su file i dati ricevuti
    filesd.close();                     // Chiude il file su sd
  }
}
//
// ****** routine di conversione da gradi milionesimali gradi sessagesimali **********
// ********* degree conversion routine (from millionesimal to sexagesimal) ***********
//
void converti ()
{
  gradi = coordinata / 1000000;              //i gradi sono la parte intera
  coordinata = coordinata - gradi * 1000000; //restano i milionesimi di grado
  coordinata = coordinata / 100;             // da milionesimi di grado a decimillesmi di grado
  secondi = (coordinata * 3600) / 10000;     // da decimillesimi di grado a secondi di grado
  minuti = secondi / 60;                     // calcola i minuti di grado
  secondi = secondi - minuti * 60;           // il residuo sono i secondi di grado
}
//
//*************  routine di esposizione della latitudine (in gradi sessagesimali) *************
//************************* shows latitude (in sexagesimal degrees) ****************************
//
void LAT()
{
  slat = 'N';
  if (lat < 0)
    slat = 'S';
  lat = abs (lat);
  coordinata = lat;
  converti ();
  deglat = gradi;
  minutilat = minuti;
  secondilat = secondi;
  lcd.setCursor(0, 0);
  lcd.print("LAT:");
  lcd.print (" ");
  lcd.print(deglat);
  lcd.write(0xDF);
  lcd.print(minutilat);
  lcd.print("'");
  lcd.print(secondilat);
  lcd.print (" ");
  lcd.print (slat);
  lcd.print ("  ");
}
//
// ************ routine di esposizione della longitudine (in gradi decimali) ***********
// ************************ shows longitude (in sexagesimal degrees) ******************
//
void LON()
{
```

```
  slon = 'E';
  if (lon < 0)
    slon = 'W';
  lon = abs (lon);
  coordinata = lon;
  converti ();
  deglon     = gradi;
  minutilon  = minuti;
  secondilon = secondi;
  lcd.setCursor(0, 1);
  lcd.print("LON:");
  lcd.print (" ");
  lcd.print(deglon);
  lcd.write(0xDF);
  lcd.print(minutilon);
  lcd.print("'");
  lcd.print(secondilon);
  lcd.print (" ");
  lcd.print (slon);
  lcd.print ("  ");
}
//
//
void setup()
{
  Serial.begin(9600);           // inizializza il monitor seriale
  lcd.begin (20, 4);            // inizializza il dispaly lcd
  lcd.backlight();              // illumina il display lcd
  pinMode (10, OUTPUT);         // il pin CS e' collegato alla porta 10
  pinMode (13, OUTPUT);         // led che evidenzia la frequenza di lettura delle coordinate
  lcd.clear ();
  setSyncProvider(RTC.get);     // sincronizza il timer di Arduino con i dati presenti sul modulo RTC
  if (timeStatus() != timeSet) // verifica se la sincronizzazione e' andata  a  buon fine
    lcd.print(" orologio non"); // clock not
  else
    lcd.print("   orologio");   // clock
  lcd.setCursor (0, 1);
  lcd.print  (" sincronizzato");// syncronized
  delay (1500);
  lcd.clear();
  if (!SD.begin(10))             // verifica la presenza della secure digital
    lcd.print ("SD KO");
  else
    lcd.print ("SD OK");
  delay (1500);
  tempoprec = millis ();        // memorizza il momento di inizio lavoro
}
//
//
void loop()
{
  while (Serial.available())
  {
    digitalWrite (13, HIGH);
    c = Serial.read();          // legge i dati provenienti dal modulo GPS
    if (gps.encode(c))          // apparentemente inutile, forse decodifica i dati
    {
      // inserire qui una eventuale codifica di trattamento del segnale ricevuto
    }
  }
  digitalWrite (13, LOW);
  gps.get_position(&lat, &lon, &fix_age); // rileva lat/long (segnati) in milionesimi di grado
  LAT();                                  // decodifica ed espone latitudine
  LON();                                  // decodifica ed espone longitudine
  tempocorrente = millis ();
  if ((tempocorrente - tempoprec) > intervallo) // se e' trascorso il tempo previsto in intervallo
  {
    tempoprec = tempocorrente;
    scrivi ();                          // scrive un item su sd
  }
}
```