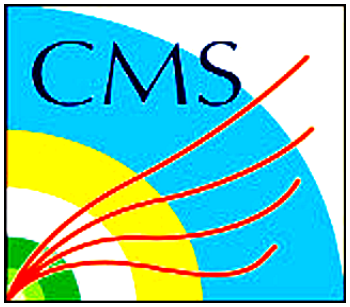
Costruiamo il termometro del Big Bang



1 Intoduzione

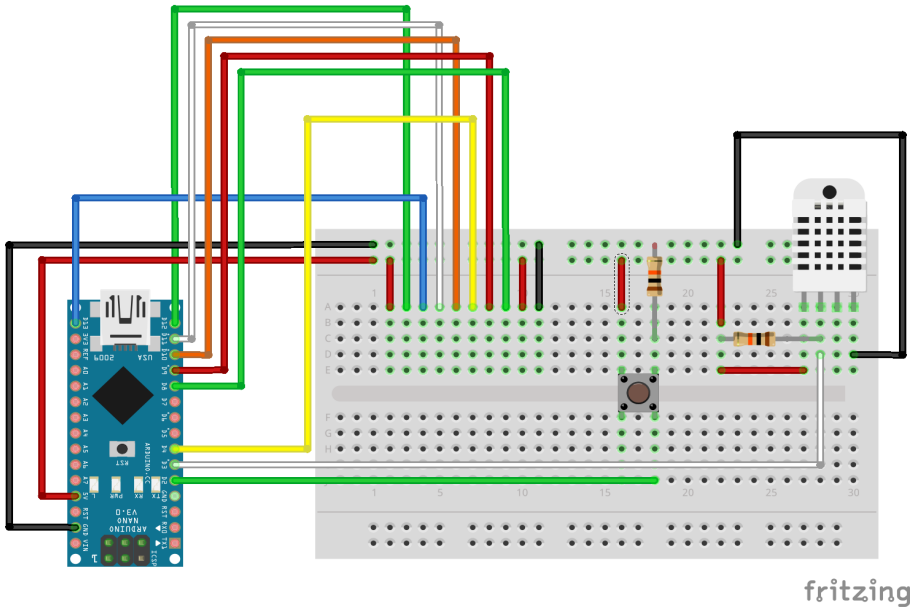
Per questa attività, utilizziamo una scheda a micro-controllore arduino nano (*basata sul circuito integrato ATmega328*), uno schermo TFT-LCD (*thin-film-transistor liquid crystal display*) e un sensore di umidità e temperatura DHT11.
Lo scopo è di visualizzare i valori della temperatura (T), dell'umidità (RH) e del punto di rugiada (che si può calcolare a partire da T e RH) e farne un grafico in tempo reale.

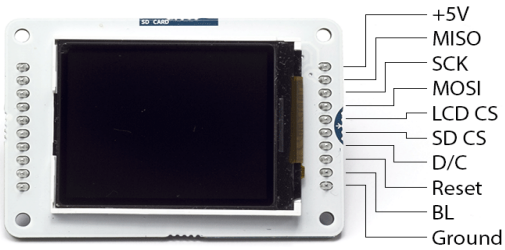
2 Perchè si chiama "termometro del Big-Bang" ?

In questa esperienza, il "valore aggiunto" da CMS (per ricordarvi che il *Compact Muon Solenoid* è un esperimento di fisica delle particelle) è legato ad una considerazione che di solito non siamo abituati a fare nella nostra vita quotidiana ma che nello studio del comportamento quantistico delle particelle elementari risulta piuttosto comune: la temperatura e' una quantità che ci assicura che il tempo sta scorrendo (che dell'informazione sta fluendo dal o nel sistema che si considera). Nell'attuale modello cosmologico standard, l'universo che osserviamo oggi ha avuto inizio da una fluttazione chiamata "Big Bang" avvenuta circa 15 miliardi di anni fa. Dopo un periodo chiamato "inflazione" (durato circa 1 milione di anni) la luce che oggi vediamo, ha incominciato a viaggiare indisturbata nel cosmo. In questo scenario, la temperatura può sostituire la nostra unità di tempo più convenzionale (i secondi) nel misurare il tempo trascorso da quella gigantesca fluttuazione poichè l'universo è andato raffreddandosi sempre più, di pari passo con la sua espansione.
In prima approssimazione dopo t secondi dal "Big-Bang" temperatura (T) ed energia (E) sono date dalle seguenti relazioni:
$T = 10^{10}/\sqrt{t}$ $E = 860000/\sqrt{t}$ rispettivamente misurate in Kelvin ed electron-Volts (eV).
Misurando una temperatura possiamo quindi stimare dopo quanto tempo dal "Big-Bang" il nostro universo aveva raggiunto tale temperatura e a quale energia questa corrisponde. Per esempio, si vedrà che a 20 ^{o}C ($\simeq$ 293 K) corrispondono poche decine di milli electron-Volt $\simeq$ 20 meV. Per capire a cosa questo corrisponda, si tenga presente che per strappare un singolo elettrone ad un atomo (energia di prima ionizzazione) ci vogliono al minimo $\simeq$5 eV cioè energie centinaia di volte maggiori. Nonostante la temperatura ambiente (20 ^{o}C) corrisponda a bassi valori di energia, si vedrà che ci saranno voluti $\simeq$4 milioni di anni all'universo per raffreddarsi ad una tale temperatura. Attualmente l'universo ha raggiunto una temperatura vicina allo zero assoluto di circa 3 K (-270 ^{o}C).
Negli acceleratori di particelle, si fa il contrario: accelerando, cioè fornendo energia a delle particelle, si risale alle condizioni primordiali delle energie disponibili nell'universo subito dopo il "Big-Bang". Con l'acceleratore LHC si portano dei protoni ad un'energia di 6.5 TeV (Tera-electronVolts; 1 TeV = $10^{12} eV$) e facendoli collidere frontalmente si accede alla spaventosa energia di 13 TeV nel centro di massa delle collisioni; mettendo questa cifra nella formula di cui sopra, si scopre che si risale a: $t = (8.6\ 10^5/13\ 10^{12})^{1/2} \simeq 8\ 10^{-4}\ s$ cioè a circa un millesimo di secondo dopo il "Big-Bang".
Grazie alla relazione di Einstein (equivalenza tra massa ed energia): $E{=}mc^2$ (dove $c{=}3\ 10^8\ m/s$ è la velocità della luce) tutta questa energia è disponibile per la creazione di particelle non facilmente osservabili al giorno d'oggi ma che erano comuni al tempo del "Big-Bang". Ed è proprio analizzando le particelle prodotte in queste collisioni che l'esperimento CMS le ritrova (artificialmente riprodotte) anche ai giorni nostri.
Questo è solo un esempio di come studiando l'infinitesimamente piccolo (le particelle) si abbia accesso all'infinitamente grande (l'universo e la sua storia) e ci è sembrato bello che tutte queste considerazioni si possano fare semplicemente a partire dalla temperatura fornita da un termometro... ma non uno qualsiasi, il termometro del Big-Bang appunto.

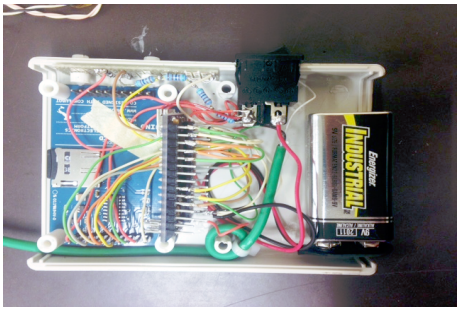
3 Montiamo il termometro

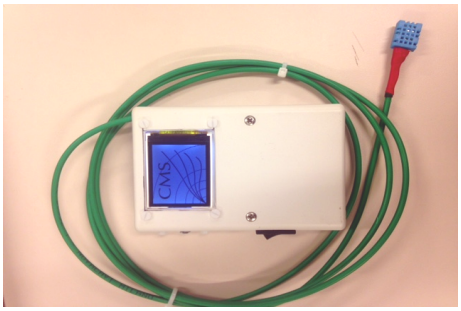
Consigliamo vivamente di realizzare prima un montaggio temporaneo per verificare la funzionalità di tutte le componenti. Bisogna armarsi di pazienza e realizzare il circuito riportato in figura (qui sotto a sinistra) a mezzo dei fili di connessione temporanea e di una basetta forata ("bread-board") che facilita la realizzazione di prototipi poichè alcuni fori lungo le righe o colonne della *bread-board* sono tutti in contatto elettrico tra di loro.

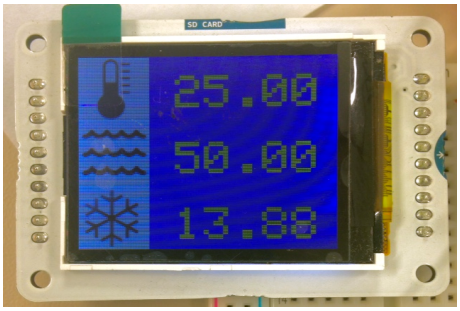




La parte più complicata del circuito è costituita dai 10 fili che servono a gestire lo schermo. Questi 10 fili sono preparati in modo tale che si possa direttamente collegare lo schermo alla *bread-board* secondo lo schema di connessione riportato nella foto dello schermo a destra (con la nomenclatura dei vari pin). Si tenga presente che i +5V dello schermo vanno collegati al filo rosso più a sinistra mentre il *Ground* (GND) dello schermo a quello nero più a destra. In pratica lo schermo va ruotato di un anolo retto in verso anti-orario e collegato alla *bread-board* nella riga sotto a quella dei suoi fili, ma questo non è rappresentato nella figura dello schema elettrico.
Il resto del circuito è semplice e contiene i 3 fili necessari per la connessione del sensore DHT11 e la connessione del tasto che verrà programmato per agire da "volta pagina": dal logo di CMS al display di Temperatura, RH umidità, DP punto di rugiada al grafico in tempo reale (in una scala arbitraria) di questi valori per finire con il display dell'equivalente della temperatura in meV e della relativa stima dell'epoca, in milioni di anni dal "Big-Bang", a cui questa temperatura era raggiunta nell'universo.
Nel circuito, tutte le resistenze sono da 10 kHom.
Solo dopo avere caricato nell'arduino nano il programma riportato di seguito (vedi sezione 4) ed avere verificato che tutto funzioni correttamente potrete, se ne avete la possibilità, procedere alla saldatura di tutte le connessioni e alla costruzione di una scatola nella quale alloggiare comodamente il termometro, come illustrato nelle figure qui sotto. Notare che nel caso in cui si voglia rendere il termometro indipendente da un cavo di alimentazione, bisognerà provvedere a collegare una batteria (quella da 9 V andrà benissimo, come da noi suggerito nella figura del montaggio).







4 Programmiamo l'arduino nano

Di seguito diamo per scontato che qualcuno vi abbia aiutato a configurare l'ambiente di programmazione dell'arduino (Arduino IDE). Prima di procedere alla compilazione e al caricamento di questo programma nell'arduino nano, è necessario installare alcune librerie specifiche per gestire:

- (#include <SD.h>) una scheda di memoria micro SD che viene inserita nell'apposito alloggiamento dietro allo schermo tft-lcd e che serve per archiviare alcune foto (il logo di CMS e i disegni delle schermate per la temperatura nelle varie modalità -classica e "Big-Bang"-). Le foto sono contenute nel folder di files che viene fornito per questa attività. Sono in formato "bitmap" e vanno semplicemente copiate sulla scheda micro SD che deve essere formattata preventivamente nel formato fat16.
- (#include <SPI.h> e include <TFT.h>) per gestire lo schermo tft-lcd
- (#include <dht.h>) per gestire il sensore DHT11

Queste librerie vanno scaricate dal sito ufficiale di Arduino e spacchettate/installate nella sotto-directory *libraries* del vostro folder *sketchbook* (presente in tutte le installazioni dell'Arduino IDE).
Per il resto, come vedrete, il programma è piuttosto semplice e nel loop principale non fa altro che occuparsi di controllare quale schermata deve essere visualizzata. In base a quale schermata va visualizzata, viene richiamata di volta in volta la routine specifica che la visualizza.
L'unica osservazione da aggiungere è relativa all'algoritmo impiegato per il calcolo del punto di rugiada: ci siamo avvalsi di un algoritmo NOAA documentato in [ref.[1]] che calcola anche la pressione di vapor saturo. Questa variabile, nel codice allegato, rimane privata all'interno della routine che calcola il punto di rugiada ma potreste implementarne voi stessi la visualizzazione per esercitarvi.

[1] *http://wahiduddin.net/calc/density_algorithms.htm* e *http://www.colorado.edu/geography/weather_station/Geog_site/about.htm*

5 Listato del programma

```
/*
================================================================================
================================================================================
 CMS Outreach gadjet illustrating the
 use of a DHT11 sensor (temperature and humidity) to:
 -- compute the dew-point using NOAA Environmental Research Laboratories algortim
 -- the use of natural units to display
    the equivalent in milli-eV of the actual temperature
    how many milions year since the Big Bang this temperature would correspond to
 -- display the online values for Temperature, Relative Humidity and Dewpoint

 The user can swith displays by pushing a button
 Exhibit based on Arduino Nano and Arduino TFT LCD SPI Screen
 The background images are loaded from a micro-SD card

 Credits:
 CERN
 15 December 2015
 by Francesco Palmonari
 email: palmo4u@gmail.com
================================================================================
================================================================================
*/

// include the necessary libraries
#include <SPI.h>   // Arduino LCD library
#include <SD.h>    // for the micro-sd card
#include <TFT.h>   // Arduino LCD library
#include <dht.h>   // DHT11 sensor library

#define dht_dpin 3 // DATA sensor pin

// pin definition for the Uno/Nano
#define sd_cs  4
#define lcd_cs 10
#define dc     9
#define rst    8

TFT screen = TFT(lcd_cs, dc, rst);  // screen
PImage lastimg;                     // this variable represents the image to be drawn on screen

dht DHT;  // sensor

const int  buttonPin = 2;    // the pin that the pushbutton is attached to
int buttonState = 0;         // variable for reading the pushbutton status
int currentState = 0;        // current state of the button

char sensort[8], sensorh[8], sensord[8];  // char array to print to the screen

void setup() {
  // for debug only
  Serial.begin(9600);

  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);

  // initialize and clear the LCD screen
  screen.begin();

  // will start with CMS logo
  // try to access the SD card.
  if (!SD.begin(sd_cs)) {
    return;
  }
}
```

```
//================================================================================
void loop() {

  checkState();

  if (currentState == 0) {
    cmslogo();
  }
  if (currentState == 1) {
    sensor();
  }
  if (currentState == 2) {
    onlinegraph();
  }
  if (currentState == 3) {
    cms();
  }
}
//================================================================================
void checkState() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    currentState++;
  }

  if (currentState > 3) {
    currentState = 0;
  }

}
//================================================================================
void checkdisplay() {
  if (!lastimg.isValid()) {
    //Serial.println(F("error while loading image"));
  }
  // don't do anything if the image wasn't loaded correctly.
  if (lastimg.isValid() == false) {
    screen.background(0, 0, 0);
    return;
  }
}
//================================================================================
void sensor() {
  screen.background(0, 0, 255);
  lastimg = screen.loadImage("scale.bmp");
  checkdisplay();
  screen.image(lastimg, 0, 0);
  screen.setTextSize(3);

  int initialCurrentState = currentState;

  while (currentState == initialCurrentState) {

    checkState();

    // read the DHT11 sensor
    DHT.read11(dht_dpin);

    String t = String(DHT.temperature);
    String h = String(DHT.humidity);
    String d = String(dewPoint(DHT.temperature, DHT.humidity));
    t.toCharArray(sensort, 5);
    h.toCharArray(sensorh, 5);
```

```
    d.toCharArray(sensord, 5);

    screen.stroke(255, 255, 255);
    screen.text(sensort, 60, 10);
    screen.text(sensorh, 60, 54);
    screen.text(sensord, 60, 98);

    checkState();
    delay(800);
    checkState();

    screen.stroke(0, 0, 255);
    screen.text(sensort, 60, 10);
    screen.text(sensorh, 60, 54);
    screen.text(sensord, 60, 98);

  }
}
//================================================================================
void cms() {
  screen.background(255, 255, 0);
  lastimg = screen.loadImage("scalecms.bmp");
  checkdisplay();
  screen.image(lastimg, 0, 0);
  screen.stroke(0, 0, 0);
  screen.setTextSize(3);

  int initialCurrentState = currentState;

  while (currentState == initialCurrentState) {

    checkState();

    // read the DHT11 sensor
    DHT.read11(dht_dpin);

    // temperature in meV = 1 eV/1000
    float evt = (DHT.temperature + 273.15) / 11.605;
    String t = String( evt );
    t.toCharArray(sensort, 5);
    screen.text(sensort, 60, 20);

    checkState();

    // Milion years since the Big Bang to have this average temperature
    float years = (861733.24 / evt * 1000) * (861733.24 / evt * 1000) / 31536000000000;
    String d = String(years);
    d.toCharArray(sensord, 5);
    screen.text(sensord, 60, 80);

    checkState();
    delay(800);
    checkState();

    screen.stroke(255, 255, 0);
    screen.text(sensort, 60, 20);
    screen.text(sensord, 60, 80);
    screen.stroke(0, 0, 0);
  }
}
//================================================================================
void cmslogo() {
  lastimg = screen.loadImage("symbol.bmp");
  checkdisplay();
  screen.image(lastimg, 0, 0);

  int initialCurrentState = currentState;
```

```
  while (currentState == initialCurrentState) {

    checkState();

  }
}
//================================================================================
void onlinegraph() {
  double yt, yh, yd;
  int x = 0, _OY = 64;
  double minT = 5, maxT = 45;
  double minH = 10, maxH = 80;
  double minD = -10, maxD = 20;
  double scaleT = 128 / (maxT - minT), scaleH = 128 / (maxH - minH), scaleD = 128 / (maxD - minD);
  int initialCurrentState = currentState;

  screen.background(0, 0, 0); // black the screen
  screen.setTextSize(1);

  while (currentState == initialCurrentState) {

    checkState();

    yt = 0; yh = 0; yd = 0; //zero the coordinates

    // read the DHT11 sensor
    DHT.read11(dht_dpin);

    // (128 - XXX) because the screen origin is on the top left corner
    yt = 128 - scaleT * DHT.temperature;
    yh = 128 - scaleH * DHT.humidity;
    yd = 128 - scaleD * dewPoint(DHT.temperature, DHT.humidity);

    String t = String(DHT.temperature);
    String h = String(DHT.humidity);
    String d = String(dewPoint(DHT.temperature, DHT.humidity));
    t.toCharArray(sensort, 4);
    h.toCharArray(sensorh, 4);
    d.toCharArray(sensord, 4);

    screen.stroke(255, 0, 0); // temperature == RED
    screen.point(x, int(yt) - 1 );
    screen.point(x, int(yt) );
    screen.point(x, int(yt) + 1 );
    screen.text("T", x+5, yt-5);
    screen.text(sensort, x + 15, yt-5 );

    screen.stroke(0, 0, 255); // humidity == BLU
    screen.point(x, int(yh) - 1 );
    screen.point(x, int(yh) );
    screen.point(x, int(yh) + 1 );
    screen.text("H", x+5, yh+5);
    screen.text(sensorh, x + 15, yh+5 );

    screen.stroke(255, 255, 0); // dewpoint == YELLOW
    screen.point(x, int(yd) - 1 );
    screen.point(x, int(yd) );
    screen.point(x, int(yd) + 1 );
    screen.text("D", x+5, yd);
    screen.text(sensord, x + 15, yd );

    delay(250);
    screen.stroke(0, 0, 0);
    screen.text("T", x+5, yt-5);
    screen.text("H", x+5, yh+5);
    screen.text("D", x+5, yd);
```

```
    screen.text(sensort, x + 15, yt-5 );
    screen.text(sensorh, x + 15, yh+5 );
    screen.text(sensord, x + 15, yd );

    x++;
    if (x > 160) {
      x = 0;
      screen.background(0, 0, 0); // black the screen
    }
  }
}

//================================================================================
// dewPoint function NOAA
// reference (1) : http://wahiduddin.net/calc/density_algorithms.htm
// reference (2) : http://www.colorado.edu/geography/weather_station/Geog_site/about.htm
//
double dewPoint(double celsius, double humidity)
{
  // (1) Saturation Vapor Pressure = ESGG(T)
  double RATIO = 373.15 / (273.15 + celsius);
  double RHS = -7.90298 * (RATIO - 1);
  RHS += 5.02808 * log10(RATIO);
  RHS += -1.3816e-7 * (pow(10, (11.344 * (1 - 1 / RATIO ))) - 1) ;
  RHS += 8.1328e-3 * (pow(10, (-3.49149 * (RATIO - 1))) - 1) ;
  RHS += log10(1013.246);

  // factor -3 is to adjust units - Vapor Pressure SVP * humidity
  double VP = pow(10, RHS - 3) * humidity;

  // (2) DEWPOINT = F(Vapor Pressure)
  double T = log(VP / 0.61078); // temp var
  return (241.88 * T) / (17.558 - T);
}
```