ARDUINO WORKSHOP

Bologna, 30 Maggio 2017

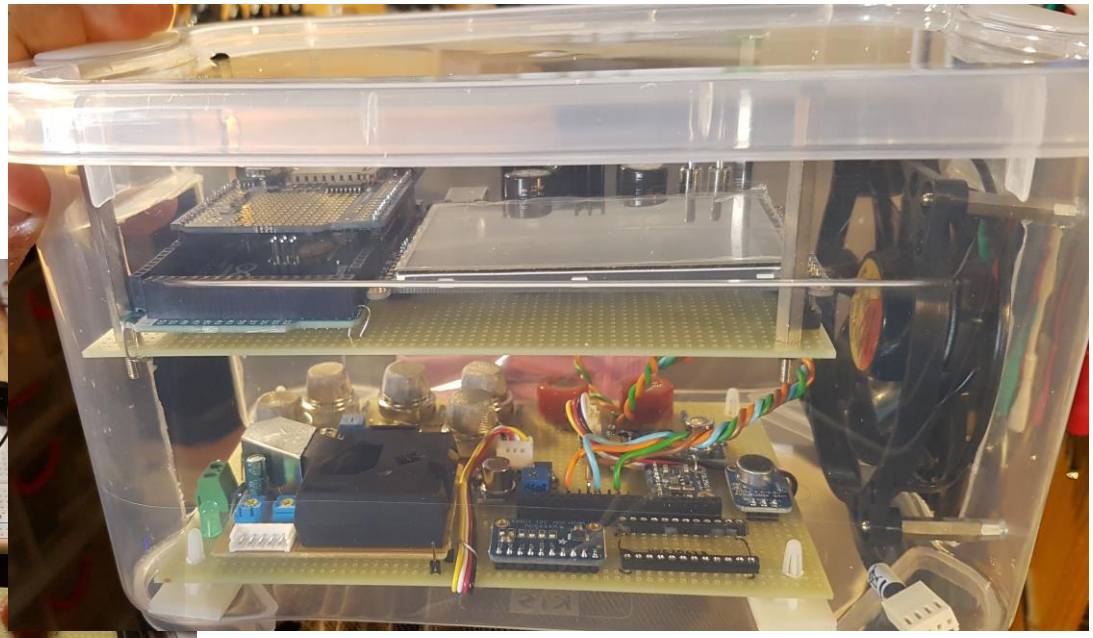
Relatore: Ing. Paolo Guidorzi

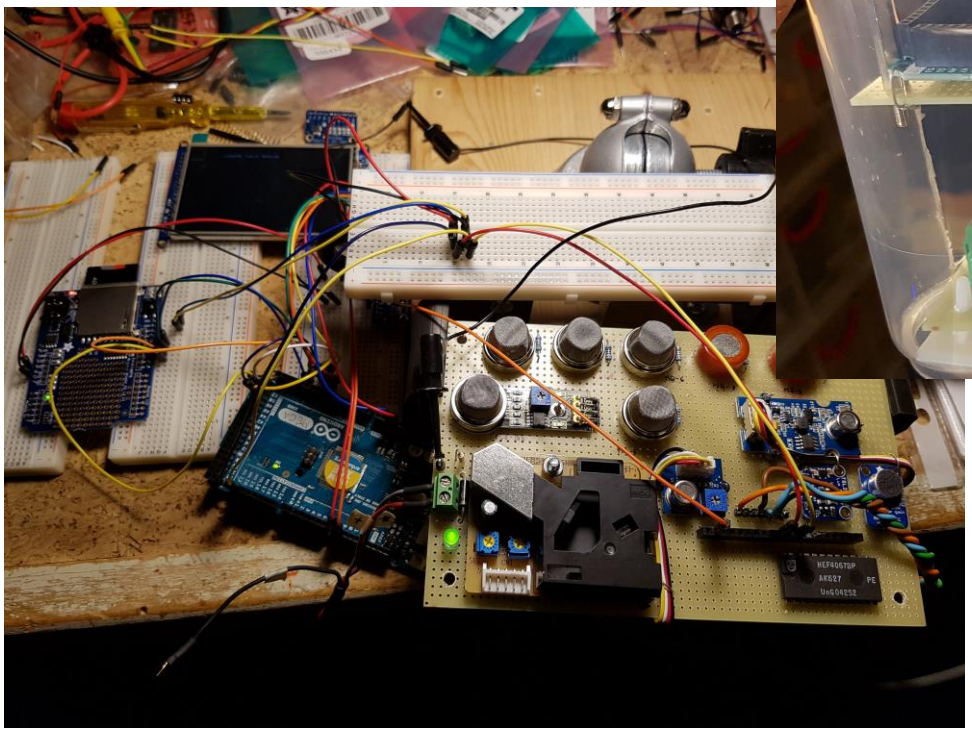
Terza parte

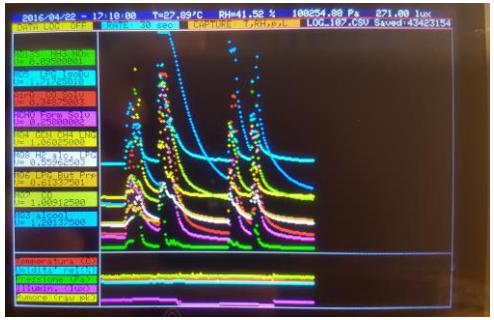
- Una realizzazione completa: il naso elettronico
- Display grafico
- ADC 16 bit
- Sensore pressione temperatura umidità
- Salvataggio dati su microSD e orologio real-time
- Multiplexer
- Sensori di gas

ARDUINO WORKSHOP

Il «naso elettronico» – Piattaforma multi-sensore integrata basata su Arduino







ARDUINO WORKSHOP

SENSORI:

TEMPERATURA (Celsius)

UMIDITA' RELATIVA (%)

PRESSIONE (Pa)

Luminosità (Lux)

MQ-3: Alcool

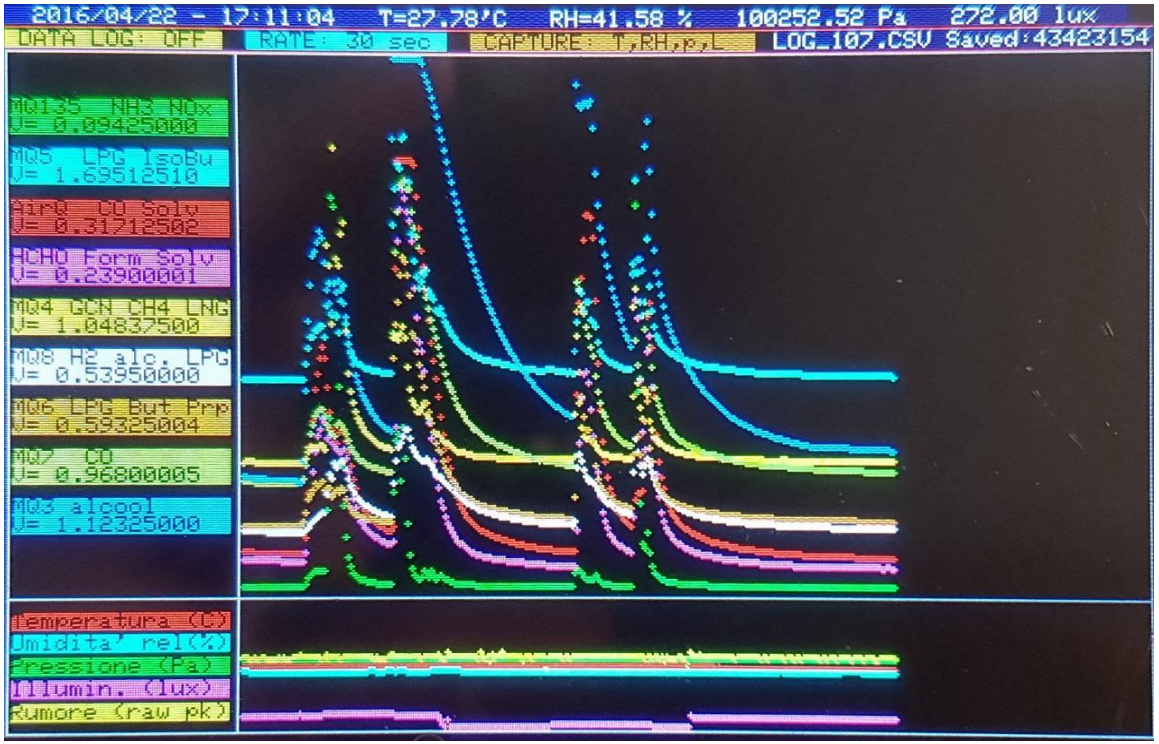
MQ-4: CH_4 metano e gas naturali

MQ-5: LPG, gas naturali

MQ-6: LPG, iso-butano, propano

MQ-7: CO

MQ-8: Idrogeno H_2

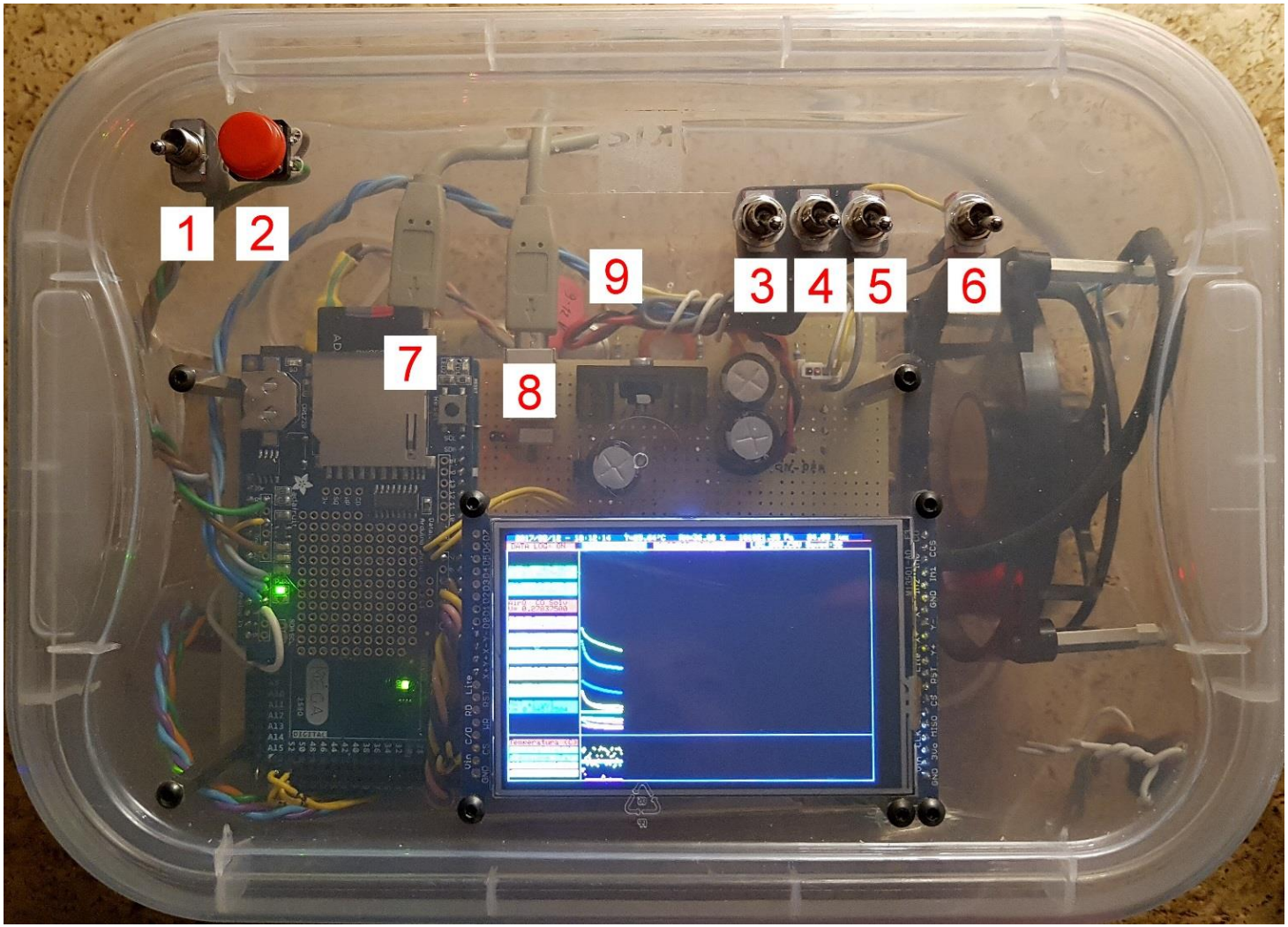


MQ-135: Ammoniaca NH_3, NO_x, alcool, benzene, fumi, CO_2, ecc

AIR-Q (MP-503): Alcool e fumi

HCHO (WSP2110): Gas organici, toluene, benzene, metanolo

ARDUINO WORKSHOP



ARDUINO WORKSHOP

Lo strumento può funzionare o tramite alimentatore esterno (9V o 12V), da attaccare alla presa rossa “9”, oppure usando un normale powerbank USB portatile. Nel caso di alimentazione da USB, entrambe le prese “7” e “8” vanno collegate al powerbank.

1: ON/OFF: (nel caso di utilizzo dell’alimentatore 9V o 12V)

2: tasto RESET: Utilizzare se all’accensione lo schermo appare bianco o quando si vuole reiniziare il log di misura su un nuovo file (filename incrementale assegnato automaticamente)

3: data log ON/OFF: abilita o disabilita la scrittura dati su file. La scrittura dati può essere abilitata e disabilitata più volte durante la misura, senza bisogno di resettare lo strumento. I dati verranno aggiunti al file corrente quando riabilitata.

4: FAST/SLOW capture: cambia il periodo di campionamento dati. Quando la modalità è SLOW vengono scritti a display i valori numerici di ogni singolo sensore, ogni ciclo, quando la modalità è FAST solo il grafico è aggiornato

5: Seleziona il campionamento da tutti i sensori di gas più Temperatura, umidità relativa, Pressione e Luminosità, oppure solo Temperatura, umidità relativa, Pressione e Luminosità. Solo i dati salvati sono mostrati nel grafico.

6: accende o spegne la ventola per la circolazione forzata dell’aria

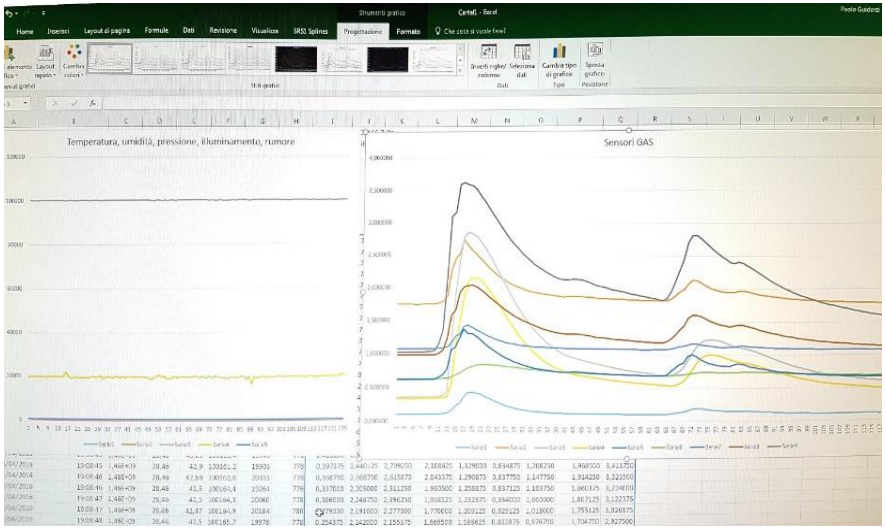
7: alimentazione USB Arduino / collegamento Arduino a PC (solo per programmazione)

8: alimentazione USB sensori. Usare 7 e 8 insieme per alimentare da pacco batteria

9: presa rossa volante: alimentazione 9V o 12V

ARDUINO WORKSHOP

I dati sono salvati sulla scheda SD. I dati salvati sono in formato di testo, importabili direttamente su Excel. All'accensione il software effettua una verifica di corretto funzionamento dei componenti del sistema. In caso venga mostrato "errore sensori" probabilmente manca l'alimentazione alla scheda sensori "8" e si sta alimentando via USB. Nel caso di utilizzo della presa "9" (non USB) entrambi i cavi USB possono essere rimossi. All'accensione viene anche mostrata la dimensione corrente della SD e il nome del file di log. A ogni accensione il numero associato al nome viene incrementato di uno, a partire dal numero dell'ultimo file salvato precedentemente. Vuotare la scheda per far ripartire la numerazione da zero.



Nota: i dati misurati di Temperatura, umidità relativa, Pressione e Luminosità sono rappresentati dal valore corretto. I dati provenienti dai sensori di gas richiedono invece una calibrazione successiva che dipende dal tipo di sensore e anche dalla temperatura e umidità corrente, come evidenziato dai datasheet dei singoli sensori.

ARDUINO WORKSHOP

_15_Naso §

```
/*************************************************
DANTE 1.0
*************************************************/

#include <SPI.h>
#include "Adafruit_GFX.h"
#include "Adafruit_HX8357.h"
#include <Wire.h>
#include <Adafruit_ADS1015.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include "RTClib.h"
#include <SD.h>
#include <Adafruit_TSL2561_U.h>

RTC_DS1307 rtc;

Adafruit_ADS1115 ads1115(0x48); // Construct an ads1115 at the default address: 0x48

// Pin del TFT
#define TFT_CS 9
#define TFT_DC 8
#define TFT_RST 7 // RST can be set to -1 if you tie it to Arduino's reset
Adafruit_HX8357 tft = Adafruit_HX8357(TFT_CS, TFT_DC, TFT_RST);

// SD
Sd2Card card;
SdVolume volume;
SdFile root;

#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // I2C

```

Inclusione delle librerie richieste per i vari sensori usati.

Inizializzazione dei sensori

ARDUINO WORKSHOP


```
34
35  // char array to print to the screen
36  char sensorPrintout[20];
37
38  float Voltage;
39  int16_t adc0,adc1;
40
41  #define black           0x0000      /*   0,   0,   0 */
42  #define navy            0x000F      /*   0,   0, 128 */
43  #define darkgreen       0x03E0      /*   0, 128,   0 */
44  #define darkcyan        0x03EF      /*   0, 128, 128 */
45  #define maroon          0x7800      /* 128,   0,   0 */
46  #define purple          0x780F      /* 128,   0, 128 */
47  #define olive           0x7BE0      /* 128, 128,   0 */
48  #define lightgrey       0xC618      /* 192, 192, 192 */
49  #define darkgrey        0x7BEF      /* 128, 128, 128 */
50  #define blue            0x051F      /*   0,   0, 255 */
51  #define green           0x07E0      /*   0, 255,   0 */
52  #define cyan            0x07FF      /*   0, 255, 255 */
53  #define red             0xF800      /* 255,   0,   0 */
54  #define magenta         0xF81F      /* 255,   0, 255 */
55  #define yellow          0xFFE0      /* 255, 255,   0 */
56  #define white           0xFFFF      /* 255, 255, 255 */
57  #define orange          0xFD20      /* 255, 165,   0 */
58  #define greenyellow     0xAFE5      /* 173, 255,  47 */
59  #define pink                        0xF81F
60
61  const int chipSelect = 10;
62  File logfile;
63
64  // address multiplexer
65  int A_zero =  2;
66  int A_one =   3;
67  int A_two =   4;
68  int A_three = 5;
69
70  Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_LOW, 12345);
```

```
71
72  int InitialDelays=200;
73  int scendigiu=18;
74
75  char filename[] = "LOG_000.CSV";
76
77  String sensorVal1,oldsensorVal1="";
78  String Orologio;
79  String Anno,Mese,Giorno,Ora,Minuto,Secondo,Unix;
80  int Switch0,Switch1,Switch2;
81
```

Definizione di variabili e costanti

ARDUINO WORKSHOP


```
83  void setup() {
84
85    tft.begin(HX8357D);
86    tft.setRotation(1);
87    tft.fillScreen(black);
88    tft.setCursor(0, 0);
89
90    tft.setTextColor(greenyellow);
91    tft.setTextSize(1);
92    tft.println("DANTE - Electronic multisensorial acquisition device. Vr 1.0");
93    tft.println("Designed and engineered by Paolo Guidorzi");
94    tft.println("paolo.guidorzi@unibo.it");
95    tft.println();
96
97    tft.setTextColor(white);
98    tft.println("---- SYSTEM CHECK ----");
99
100   tft.println("OK TFT display!");
101   delay(InitialDelays);
102
103   String oldsensorVall="";
104
105 // Initialize adsl115 ADC
106     adsl115.begin();
107     adsl115.setGain(GAIN_ONE);
108
109   if (! adsl115.getGain()==GAIN_ONE) { tft.println("Error ADC adsl115"); }
110   else { tft.println("OK adsl115 (A/D Converter)!"); }
111   delay(InitialDelays);
112
113 // Sensore Luminosita'
114   /* Initialise the sensor */
115   if(!tsl.begin()) { tft.print("Could not find a valid TSL2561 sensor!");}
116   else { tft.println("OK TSL2561 (Light sensor)!");   }
117   delay(InitialDelays);
```

Setup: inizializzazione sensori e variabili

ARDUINO WORKSHOP


```
  /* You can also manually set the gain or enable auto-gain support */
  // tsl.setGain(TSL2561_GAIN_1X);      /* No gain ... use in bright light to avoid sensor saturation */
  // tsl.setGain(TSL2561_GAIN_16X);     /* 16x gain ... use in low light to boost sensitivity */
  tsl.enableAutoRange(true);            /* Auto-gain ... switches automatically between 1x and 16x */

  /* Changing the integration time gives you better sensor resolution (402ms = 16-bit data) */
  // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);      /* fast but low resolution */
   tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_101MS);  /* medium resolution and speed   */
  // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_402MS);  /* 16-bit data but slowest conversions */

// RTC (orologio)
  if (! rtc.begin()) { tft.println("Couldn't find RTC!"); }
  else { tft.println("OK RTC (Real Time clock)!");   }
  delay(InitialDelays);

// REGOLA OROLOGIO - tenere disattivato se non per regolare
//rtc.adjust(DateTime(__DATE__, __TIME__));

// Sensore p, T, RH
  if (!bme.begin()) {
    tft.println("Could not find a valid BME280 sensor!");

    tft.println("");
    tft.setTextSize(2);
    tft.setTextColor(red,yellow);
    tft.println("                                    ");
    tft.println(" --- WARNING: SENSOR BOARD FAILURE --- ");
    tft.println("                                    ");
    delay(5000);
    tft.setTextSize(1);
    tft.setTextColor(white);
   }
  else { tft.println("OK BME280 (p, T, RH) sensor!"); }
  delay(InitialDelays);
```

Setup: inizializzazione sensori e variabili

ARDUINO WORKSHOP


```
167   int Switch_0= analogRead(A3);
168   tft.println(Switch_0);
169   tft.print("Unix time: ");
170   DateTime now = rtc.now();
171   tft.println(now.unixtime());
172
173 // SD
174   tft.print("\nInitializing SD card...");
175   pinMode(53, OUTPUT);
176   delay(InitialDelays);
177
178   if (!card.init(SPI_HALF_SPEED, 10, 11, 12, 13)) {
179     tft.println("initialization failed. Things to check:");
180     tft.println("* is a card is inserted?");
181   }
182   delay(InitialDelays);
183
184   // print the type of card
185   tft.print("\nCard type: ");
186   switch(card.type()) {
187     case SD_CARD_TYPE_SD1:
188       tft.println("SD1");
189       break;
190     case SD_CARD_TYPE_SD2:
191       tft.println("SD2");
192       break;
193     case SD_CARD_TYPE_SDHC:
194       tft.println("SDHC");
195       break;
196     default:
197       tft.println("Unknown");
198   }
199   delay(InitialDelays);
```

Setup: inizializzazione sensori e variabili

ARDUINO WORKSHOP


```
201   // Now we will try to open the 'volume'/'partition' - it should be FAT16 or FAT32
202   if (!volume.init(card)) {
203     tft.println("Could not find FAT16/FAT32 partition.\nMake sure you've formatted the card");
204     tft.println("You cannot run the software without SD Card");
205     if (Switch_0>500) {while(1) { };}
206   }
207
208   // print the type and size of the first FAT-type volume
209   uint32_t volumesize;
210   tft.print("\nVolume type is FAT");
211   tft.println(volume.fatType(), DEC);
212   tft.println();
213
214   volumesize = volume.blocksPerCluster();    // clusters are collections of blocks
215   volumesize *= volume.clusterCount();       // we'll have a lot of clusters
216   volumesize *= 512;                            // SD card blocks are always 512 bytes
217   volumesize /= 1024;
218   tft.print("Volume size (Mbytes): ");
219   volumesize /= 1024;
220   tft.println(volumesize);
221
222   root.openRoot(volume);
223
224   // list all files in the card with date and size
225   // root.ls(LS_R | LS_DATE | LS_SIZE);
226
227   // see if the card is present and can be initialized:
228   if (!SD.begin(10, 11, 12, 13)) {
229     tft.println("Card failed, or not present");
230     // don't do anything more:
231     //return;
232   }
233   tft.println("card initialized.");
```

Setup: inizializzazione sensori e variabili

ARDUINO WORKSHOP


```
235   // crea il nome file LOG_XYZ.CSV
236   for (uint8_t i = 0; i < 1000; i++) {
237     sprintf(filename, "LOG_%03d.CSV", i);
238     tft.print(".");
239     if (! SD.exists(filename)) {   // appena non esiste un file con numero XYZ, lo genera
240       // only open a new file if it doesn't exist
241       logfile = SD.open(filename, FILE_WRITE);
242       break;  // leave the loop!
243     }
244   }
245     tft.println("");
246     tft.print("filename: "); tft.println(filename);
247     delay(InitialDelays);
248
249   if (! logfile) {
250     tft.println("couldnt create file");
251   }
252
253   tft.print("Logging to: ");
254   tft.println(filename);
255   delay(1000);
256
257   Anno=String(now.year());
258   Mese=String(now.month()); if (Mese.toInt()<10)    { Mese="0"+Mese;}
259   Giorno=String(now.day()); if (Giorno.toInt()<10) { Giorno="0"+Giorno;}
260   Ora=String(now.hour()); if (Ora.toInt()<10) { Ora="0"+Ora;}
261   Minuto=String(now.minute()); if (Minuto.toInt()<10) { Minuto="0"+Minuto;}
262   Secondo=String(now.second()); if (Secondo.toInt()<10) { Secondo="0"+Secondo;}
263   Unix=String(now.unixtime());
264   Orologio=Anno+"/"+Mese+"/"+Giorno+" "+Ora+":"+Minuto+":"+Secondo+" "+Unix;
265   sensorVall = String()+ "  T= "+bme.readTemperature()+" 'C  RH= "+bme.readHumidity()+" %  "+bme.readPressure()+" Pa ";
266   Orologio=Orologio+sensorVall;
267
268   logfile.println(Orologio);
269   logfile.println("YYYY/MM/DD HH:MM:SS UNIXTIME## TT.00 RH.00 PRESSI.00 NOISE LUX.00 0.SENSOR01 0.SENSOR02 0.SENSOR03 0.SENSOR
270
271   logfile.flush();
```

Assegnazione del nome con numerazione progressiva del file di LOG

Setup: creazione del file di LOG e salvataggio della prima riga.

Il comando «flush()» forza la scrittura, creando così il file

ARDUINO WORKSHOP


```
// Multiplexer
  pinMode(A_zero, OUTPUT);      // sets the digital pin "A_zero" as output
  pinMode(A_one, OUTPUT);       // sets the digital pin "A_one" as output
  pinMode(A_two, OUTPUT);       // sets the digital pin "A_two" as output
  pinMode(A_three, OUTPUT);     // sets the digital pin "A_three" as output

  delay(3*InitialDelays);

  tft.setTextSize(2);
  tft.println("");
  tft.setTextColor(pink);
  tft.print("Starting acquisition");
  tft.setTextColor(white);
  delay(2*InitialDelays);
  tft.setTextSize(1);

// preparazione schermata di acquisizione
  tft.fillScreen(black);  // pulisce schermo
  tft.setTextWrap(false);

  tft.drawLine(0,8,480,8,red);
  tft.drawLine(0,18,480,18,red);

  tft.drawLine(0,19,480,19,lightgrey);
  tft.drawLine(0,260,480,260,lightgrey);
  tft.drawLine(95,19,95,319,lightgrey);
  tft.drawLine(0,19,0,319,lightgrey);
  tft.drawLine(0,319,479,319,lightgrey);
  tft.drawLine(478,19,478,319,lightgrey);

// Sensore 1
 tft.setCursor(2, scendigiu+22);
 tft.setTextColor(black,green);
 tft.print("MQ135  NH3 NOx ");
 tft.setCursor(2, scendigiu+22+8*1);
 tft.print("Val            ");
```

```
// Sensore 2
 tft.setCursor(2, scendigiu+6+22+8*2);
 tft.setTextColor(black,cyan);
 tft.print("MQ5  LPG IsoBu ");
 tft.setCursor(2, scendigiu+6+22+8*3);
 tft.print("Val            ");

// Sensore 3
 tft.setCursor(2, scendigiu+12+22+8*4);
 tft.setTextColor(black,red);
 tft.print("AirQ  CO Solv  ");
 tft.setCursor(2, scendigiu+12+22+8*5);
 tft.print("Val            ");

// Sensore 4
 tft.setCursor(2, scendigiu+18+22+8*6);
 tft.setTextColor(black,magenta);
 tft.print("HCHO Form Solv ");
 tft.setCursor(2, scendigiu+18+22+8*7);
 tft.print("Val            ");
```

Creazione della grafica del display

ARDUINO WORKSHOP

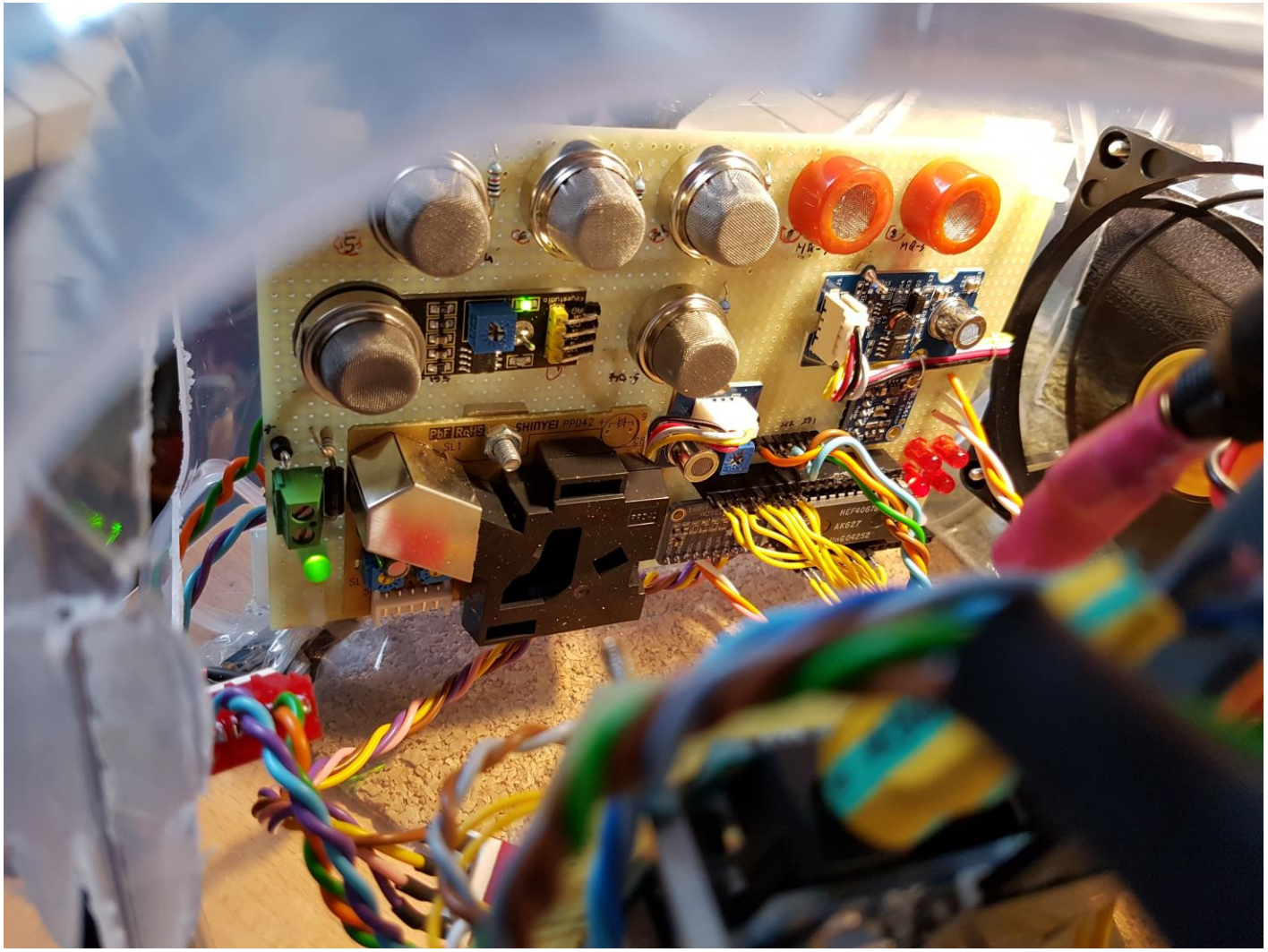

```
// Sensore 5
 tft.setCursor(2, scendigiu+24+22+8*8);
 tft.setTextColor(black,yellow);
 tft.print("MQ4 GCN CH4 LNG");
 tft.setCursor(2, scendigiu+24+22+8*9);
 tft.print("Val            ");

// Sensore 6
 tft.setCursor(2, scendigiu+30+22+8*10);
 tft.setTextColor(black,white);
 tft.print("MQ8 H2 alc. LPG");
 tft.setCursor(2, scendigiu+30+22+8*11);
 tft.print("Val            ");

// Sensore 7
 tft.setCursor(2, scendigiu+36+22+8*12);
 tft.setTextColor(black,orange);
 tft.print("MQ6 LPG But Prp");
 tft.setCursor(2, scendigiu+36+22+8*13);
 tft.print("Val            ");

// Sensore 8
 tft.setCursor(2, scendigiu+42+22+8*14);
 tft.setTextColor(black,greenyellow);
 tft.print("MQ7  CO        ");
 tft.setCursor(2, scendigiu+42+22+8*15);
 tft.print("Val            ");

// Sensore 9
 tft.setCursor(2, scendigiu+48+22+8*16);
 tft.setTextColor(black,blue);
 tft.print("MQ3 alcool     ");
 tft.setCursor(2, scendigiu+48+22+8*17);
 tft.print("Val            ");
```

```
// Temperatura
 tft.setCursor(2, 100+22+8*18);
 tft.setTextColor(black,red);
 tft.print("Temperatura (C)");

// Umidita'
 tft.setCursor(2, 102+22+8*19);
 tft.setTextColor(black,cyan);
 tft.print("Umidita' rel(%)");

// Pressione
 tft.setCursor(2, 104+22+8*20);
 tft.setTextColor(black,green);
 tft.print("Pressione (Pa) ");

// Luminosita'
 tft.setCursor(2, 106+22+8*21);
 tft.setTextColor(black,magenta);
 tft.print("Illumin. (lux) ");

// Rumore
 tft.setCursor(2, 108+22+8*22);
 tft.setTextColor(black,yellow);
 tft.print("Rumore (raw pk)");
}
```

Creazione della grafica del display

FINE della parte di SETUP

ARDUINO WORKSHOP



ARDUINO WORKSHOP


```
399 boolean datalogging,ratefastslow,ambientegas;
400 int16_t xPos = 96;
401 int16_t yPos;
402
403 boolean scrivivalori=true;
404 int16_t lumin;
405 int16_t pressio;
406 int16_t temper;
407 static float f_val = 123.6794;
408 static char outstr[15];
409
410 float sens1,sens2,sens3,sens4,sens5,sens6,sens7,sens8,sens9;
411 String sens1S,sens2S,sens3S,sens4S,sens5S,sens6S,sens7S,sens8S,sens9S;
412
413 int16_t DatiScrittiSuSD = 0;
414
415 void loop(void) {
416
417  sensors_event_t event;
418  tsl.getEvent(&event);
419
420  DateTime now = rtc.now();
421  tft.setCursor(0, 0);
422
423  Anno=String(now.year());
424  Mese=String(now.month()); if (Mese.toInt()<10)    { Mese="0"+Mese;}
425  Giorno=String(now.day()); if (Giorno.toInt()<10) { Giorno="0"+Giorno;}
426  Ora=String(now.hour()); if (Ora.toInt()<10) { Ora="0"+Ora;}
427  Minuto=String(now.minute()); if (Minuto.toInt()<10) { Minuto="0"+Minuto;}
428  Secondo=String(now.second()); if (Secondo.toInt()<10) { Secondo="0"+Secondo;}
429  Unix=String(now.unixtime());
430  Orologio="  "+Anno+"/"+Mese+"/"+Giorno+" - "+Ora+":"+Minuto+":"+Secondo;
```

Dichiarazione di alcune variabili e inizio della parte LOOP

ARDUINO WORKSHOP


```
sensorVall = String()+ "   T="+bme.readTemperature()+"'C   RH="+bme.readHumidity()+" %   "+bme.readPressure()+" Pa   ";
Orologio=Orologio+sensorVall;

if (event.light) {
 Orologio=Orologio+event.light+" "+"lux  ";
 lumin=event.light;
 };

if (scrivivalori==true) {
 tft.setTextSize(1);
 tft.setTextColor(white,navy);
 tft.print(Orologio);
 }

if ( bme.readTemperature()>120 || bme.readHumidity()==0)
{
 tft.setTextSize(2);
 tft.setTextColor(red,yellow);
 tft.setCursor(0, 100);
 tft.println("                                        ");
 tft.println(" --- WARNING: SENSOR BOARD FAILURE --- ");
 tft.println("                                        ");
 tft.setTextSize(1);
}

 xPos = xPos + 1;
 if(xPos>=tft.width()-2) {
 xPos=96;
 tft.fillRect(96,20,382,239,black);
 tft.fillRect(96,261,382,58,black);
}
```

Scrittura della prima riga nel display (temperatura, umidità, pressione, orario,…)

ARDUINO WORKSHOP


```
// Sensore 1
// Select address 0000  =0
 if (ambientegas==true) {
  digitalWrite(A_zero, LOW);digitalWrite(A_one,  LOW);digitalWrite(A_two,  LOW);digitalWrite(A_three,LOW);
  adc0 = ads1115.readADC_SingleEnded(0);
  dtostrf(adc0*4.096/32768,10, 8, outstr);
  sens1S=outstr;
  if (scrivivalori==true) {
    tft.setTextColor(black,green);
    tft.setCursor(2, scendigiu+22+8*1);
    tft.print(String()+"V= "+outstr);
    }
  tft.fillCircle(xPos,260-adc0/137,1,green);
 } else {sens1S="";}

// Sensore 2
// Select address 0001  =1
 if (ambientegas==true) {
  digitalWrite(A_zero, HIGH);digitalWrite(A_one,  LOW);digitalWrite(A_two,  LOW);digitalWrite(A_three,LOW);
  adc0 = ads1115.readADC_SingleEnded(0);
  dtostrf(adc0*4.096/32768,10, 8, outstr);
  sens2S=outstr;
  if (scrivivalori==true) {
    tft.setTextColor(black,cyan);
    tft.setCursor(2, scendigiu+6+22+8*3);
    tft.print(String()+"V= "+outstr);
    }
  tft.fillCircle(xPos,260-adc0/137,1,cyan);
 } else {sens2S="";}
```

Acquisizione dati dai vari sensori di gas, scrittura sul display del valore (se l'opzione è abilitata) e plot grafico

ARDUINO WORKSHOP


```
// Sensore 3
// Select address 0010  =2
 if (ambientegas==true) {
  digitalWrite(A_zero, LOW);digitalWrite(A_one, HIGH);digitalWrite(A_two, LOW);digitalWrite(A_three, LOW);
  adc0 = ads1115.readADC_SingleEnded(0);
  dtostrf(adc0*4.096/32768,10, 8, outstr);
  sens3S=outstr;
  if (scrivivalori==true) {
    tft.setTextColor(black,red);
    tft.setCursor(2, scendigiu+12+22+8*5);
    tft.print(String()+"V= "+outstr);
    }
  tft.fillCircle(xPos,260-adc0/137,1,red);
 } else {sens3S="";}

// Sensore 4
// Select address 0011  =3
 if (ambientegas==true) {
  digitalWrite(A_zero, HIGH);digitalWrite(A_one, HIGH);digitalWrite(A_two, LOW);digitalWrite(A_three, LOW);
  adc0 = ads1115.readADC_SingleEnded(0);
  dtostrf(adc0*4.096/32768,10, 8, outstr);
  sens4S=outstr;
  if (scrivivalori==true) {
    tft.setTextColor(black,magenta);
    tft.setCursor(2, scendigiu+18+22+8*7);
    tft.print(String()+"V= "+outstr);
    }
  tft.fillCircle(xPos,260-adc0/137,1,magenta);
 } else {sens4S="";}
```

Acquisizione dati dai vari sensori di gas, scrittura sul display del valore (se l'opzione è abilitata) e plot grafico

ARDUINO WORKSHOP


```
// Sensore 5
// Select address 0100  =4
 if (ambientegas==true) {
  digitalWrite(A_zero, LOW);digitalWrite(A_one, LOW);digitalWrite(A_two, HIGH);digitalWrite(A_three, LOW);
  adc0 = ads1115.readADC_SingleEnded(0);
  dtostrf(adc0*4.096/32768,10, 8, outstr);
  sens5S=outstr;
  if (scrivivalori==true) {
    tft.setTextColor(black,yellow);
    tft.setCursor(2, scendigiu+24+22+8*9);
    tft.print(String()+"V= "+outstr);
    }
  tft.fillCircle(xPos,260-adc0/137,1,yellow);
 } else {sens5S="";}

// Sensore 6
// Select address 0101 =5
 if (ambientegas==true) {
  digitalWrite(A_zero, HIGH);digitalWrite(A_one, LOW);digitalWrite(A_two, HIGH);digitalWrite(A_three, LOW);
  adc0 = ads1115.readADC_SingleEnded(0);
  dtostrf(adc0*4.096/32768,10, 8, outstr);
  sens6S=outstr;
  if (scrivivalori==true) {
    tft.setTextColor(black,white);
    tft.setCursor(2, scendigiu+30+22+8*11);
    tft.print(String()+"V= "+outstr);
    }
  tft.fillCircle(xPos,260-adc0/137,1,white);
 } else {sens6S="";}
```

Acquisizione dati dai vari sensori di gas, scrittura sul display del valore (se l'opzione è abilitata) e plot grafico

ARDUINO WORKSHOP


```
555 // Sensore 7
556 // Select address 0110 =6
557  if (ambientegas==true) {
558   digitalWrite(A_zero, LOW);digitalWrite(A_one, HIGH);digitalWrite(A_two, HIGH);digitalWrite(A_three, LOW);
559   adc0 = ads1115.readADC_SingleEnded(0);
560   dtostrf(adc0*4.096/32768,10, 8, outstr);
561   sens7S=outstr;
562   if (scrivivalori==true) {
563     tft.setTextColor(black,orange);
564     tft.setCursor(2, scendigiu+36+22+8*13);
565     tft.print(String()+"V= "+outstr);
566     }
567   tft.fillCircle(xPos,260-adc0/137,1,orange);
568  } else {sens7S="";}
569
570 // Sensore 8
571 // Select address 0111 =7
572  if (ambientegas==true) {
573   digitalWrite(A_zero, HIGH);digitalWrite(A_one, HIGH);digitalWrite(A_two, HIGH);digitalWrite(A_three, LOW);
574   adc0 = ads1115.readADC_SingleEnded(0);
575   dtostrf(adc0*4.096/32768,10, 8, outstr);
576   sens8S=outstr;
577   if (scrivivalori==true) {
578     tft.setTextColor(black,greenyellow);
579     tft.setCursor(2, scendigiu+42+22+8*15);
580     tft.print(String()+"V= "+outstr);
581     }
582   tft.fillCircle(xPos,260-adc0/137,1,greenyellow);
583  } else {sens8S="";}
```

Acquisizione dati dai vari sensori di gas, scrittura sul display del valore (se l'opzione è abilitata) e plot grafico

ARDUINO WORKSHOP


```
585 // Sensore 9
586 // Select address 1000 =8
587  if (ambientegas==true) {
588   digitalWrite(A_zero, LOW);digitalWrite(A_one, LOW);digitalWrite(A_two, LOW);digitalWrite(A_three, HIGH);
589   adc0 = ads1115.readADC_SingleEnded(0);
590   dtostrf(adc0*4.096/32768,10, 8, outstr);
591   sens9S=outstr;
592   if (scrivivalori==true) {
593     tft.setTextColor(black,blue);
594     tft.setCursor(2, scendigiu+48+22+8*17);
595     tft.print(String()+"V= "+outstr);
596     }
597   tft.fillCircle(xPos,260-adc0/137,1,blue);
598  } else {sens9S="";}
599
600 // End multiplexing
601   // Select address 0000  =0
602  digitalWrite(A_zero, LOW);digitalWrite(A_one,  LOW);digitalWrite(A_two,  LOW);digitalWrite(A_three,LOW);
603
604
605 // Temperatura
606   temper=bme.readTemperature();
607   if (temper<-15)  { temper=-15;}
608   if (temper>45) { temper=45;}
609   tft.fillCircle(xPos,318-temper,1,red);
610
611 // Umidita'
612   tft.fillCircle(xPos, 318-bme.readHumidity()/1.67 ,1,cyan);
```

Acquisizione dati dai vari sensori di gas, scrittura sul display del valore (se l'opzione è abilitata) e plot grafico

ARDUINO WORKSHOP


```
// Pressione
  pressio=bme.readPressure()-101325;
  if (pressio<1) { pressio=1;}
  if (pressio>58) { pressio=58;}
  tft.fillCircle(xPos,318-30-pressio/1,1,green);

// Luminosita'
 if (lumin>5800) {lumin=5800;}
  tft.fillCircle(xPos,318-lumin/50,1,magenta);

// Rumore
  adc1 = ads1115.readADC_SingleEnded(1);
  tft.fillCircle(xPos,320-adc1/580,1,yellow);


// legge interruttori
 Switch0= analogRead(A0);
 if(Switch0>512) {
  datalogging=true;
    tft.setCursor(0, 10);
    tft.setTextColor(black,red);
    tft.print(" DATA LOG: ON  ");
 }
 else
 {
  datalogging=false;
    tft.setCursor(0, 10);
    tft.setTextColor(black,yellow);
    tft.print(" DATA LOG: OFF ");
 }
```

```
Switch1= analogRead(A1);
if(Switch1>512) {
 ratefastslow=true;
 scrivivalori=true;

   tft.setCursor(100, 10);
   tft.setTextColor(black,cyan);
   tft.print(" SLOW/SHOW VL ");
}
else
{
 ratefastslow=false;
 scrivivalori=false;
   tft.setCursor(100, 10);
   tft.setTextColor(black,orange);
   tft.print(" FAST/PLOT OPT");
}
```

```
Switch2= analogRead(A2);
if(Switch2>512) {
 ambientegas=true;
   tft.setCursor(194, 10);
   tft.setTextColor(black,magenta);
   tft.print(" CAPTURE: GAS + Amb ");
}
else
{
 ambientegas=false;
   tft.setCursor(194, 10);
   tft.setTextColor(black,orange);
   tft.print(" CAPTURE: T,RH,p,L  ");
}
```

Grafico dei valori dagli altri sensori. Lettura degli interruttori per l'abilitazione delle varie opzioni

ARDUINO WORKSHOP


```
678     tft.setCursor(322, 10);
679     tft.setTextColor(white,navy);
680     tft.print(String()+filename+" Saved:"+DatiScrittiSuSD);
681
682    if (datalogging==true) {
683     Orologio=Anno+"/"+Mese+"/"+Giorno+" "+Ora+":"+Minuto+":"+Secondo+" "+Unix;
684     sensorVall = String()+ " "+bme.readTemperature()+" "+bme.readHumidity()+" "+bme.readPressure()+" "+adc1+" ";
685     Orologio=Orologio+sensorVall;
686     Orologio=Orologio+event.light+" ";
687
688     logfile.print(Orologio);
689
690     if (ambientegas==true) {
691       logfile.println(sens1S+" "+sens2S+" "+sens3S+" "+sens4S+" "+sens5S+" "+sens6S+" "+sens7S+" "+sens8S+" "+sens9S );
692     } else {logfile.println("");}
693
694     DatiScrittiSuSD++;
695     logfile.flush();
696    }
697
698 }
```

Scrittura dati su SD (se abilitata)

FINE del programma

ARDUINO WORKSHOP

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Ing. Paolo Guidorzi

Dipartimento di Ingegneria Industriale

paolo.guidorzi@unibo.it

http://acustica.ing.unibo.it/Staff/paolo/index.html