ARDUINO WORKSHOP

Bologna, 30 Maggio 2017

Relatore: Ing. Paolo Guidorzi

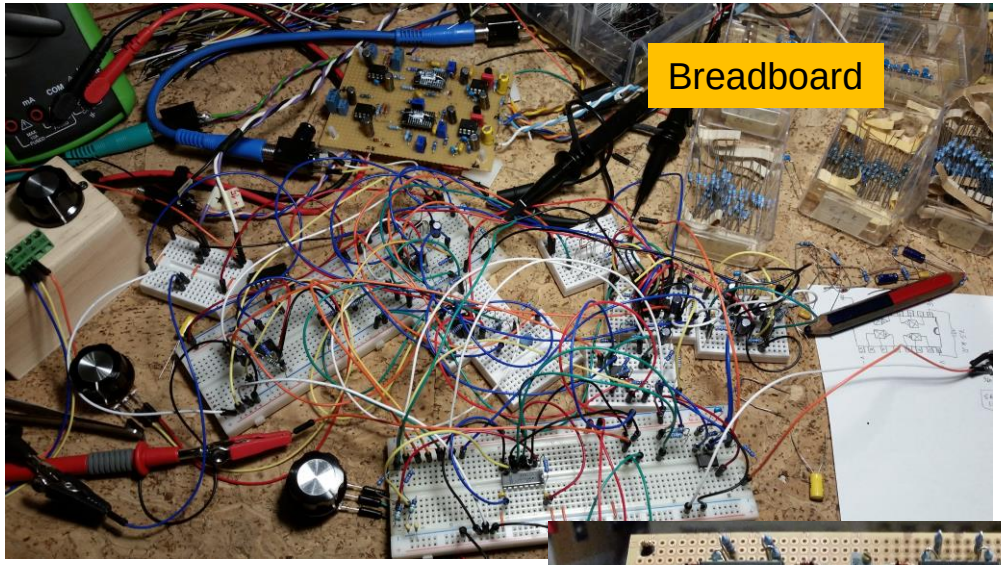
ARDUINO WORKSHOP

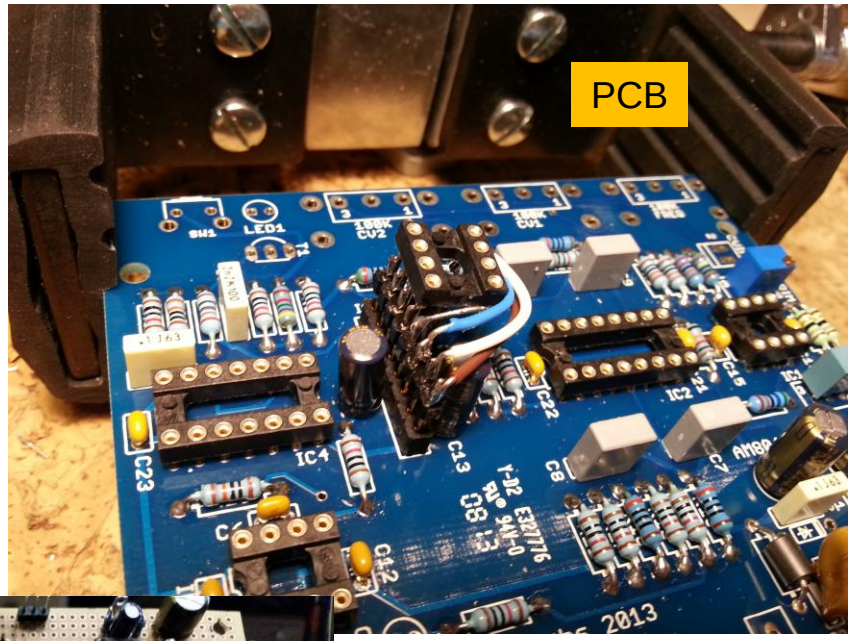
Seconda parte

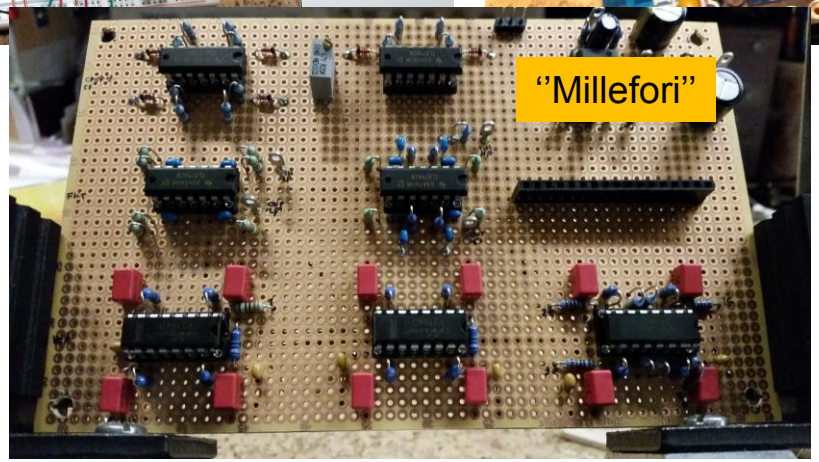
- I primi esperimenti, breadboard e millefori, Arduino Playground
- Lettura del valore di un potenziometro
- Dal valore di un potenziometro all'uscita PWM – luminosità di un LED
- Uscita PWM continuamente variabile
- Dal segnale PWM a una tensione continua. «Poor man DAC»
- Utilizzo di un pulsante. Resistenze di pull-up e pull-down
- Il partitore di tensione
- Utilizzo di sensori di tipo resistivo. La fotocellula
- Utilizzo di sensori di tipo resistivo. Sensore di GAS
- Display a 7 segmenti (seriale)
- Display grafico TFT a colori (2.8'')
- Collegamento di un altro display grafico TFT (1.8'')
- Collegamento di un altro display grafico TFT (2.2'') usando Arduino MEGA2560
- Convertitori DAC e ADC esterni
- Sensore di temperatura e pressione
- Comunicazione dati da Arduino al computer (tramite porta seriale)
- Comunicazione dati da smartphone ad Arduino tramite Bluetooth
- Utilizzo di un Multiplexer

ARDUINO WORKSHOP

Breadboard vs Millefori vs PCB







Lettura del valore di un potenziometro

- Potenziometro tra massa e 5V
- Pin centrale pot. a ingresso A0
- Pin centrale pot. a oscilloscopio

AnalogReadSerial

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

Commenti su più righe

SETUP

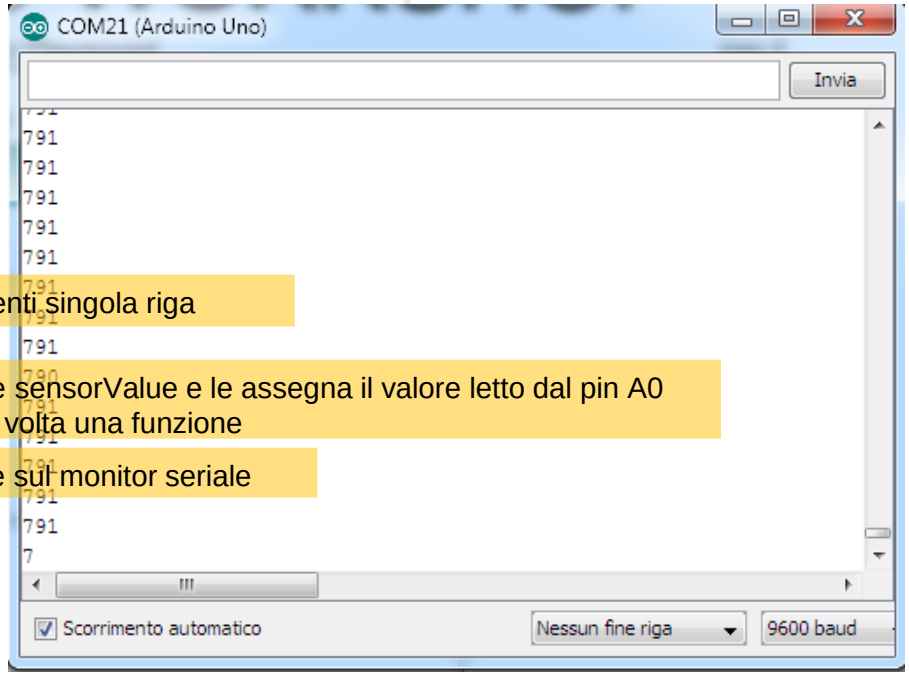
Inizializza trasmissione nel monitor seriale

Commenti singola riga

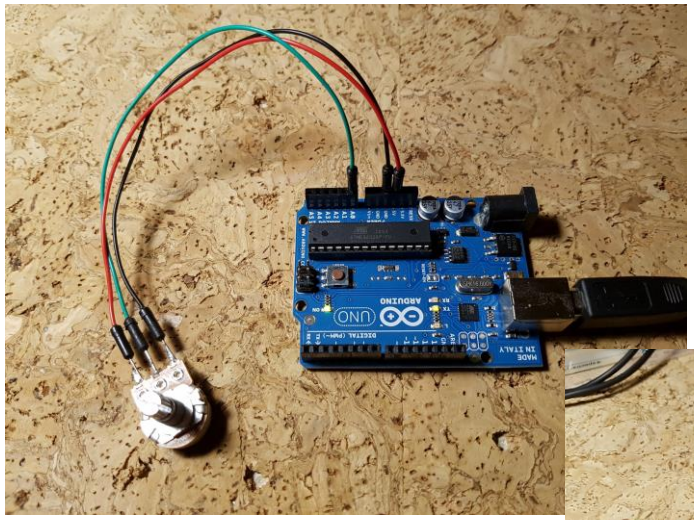
LOOP

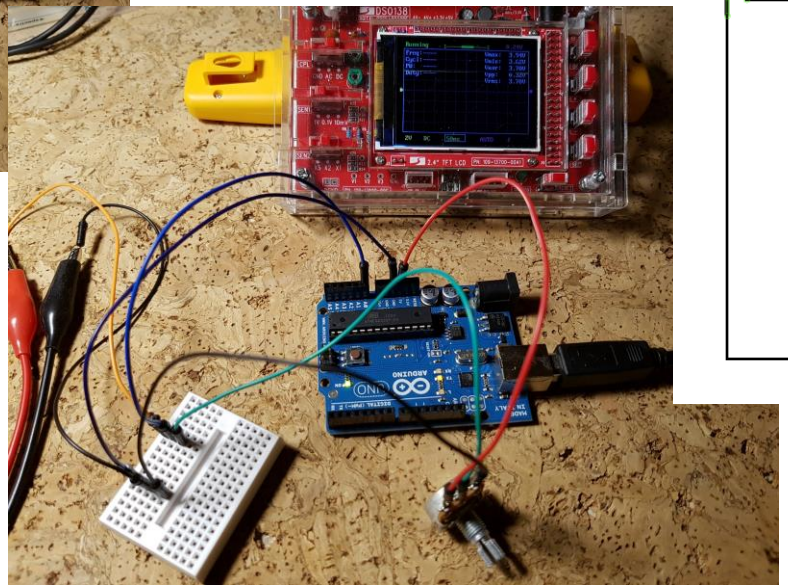
Inizializza la variabile sensorValue e le assegna il valore letto dal pin A0
AnalogRead è a sua volta una funzione

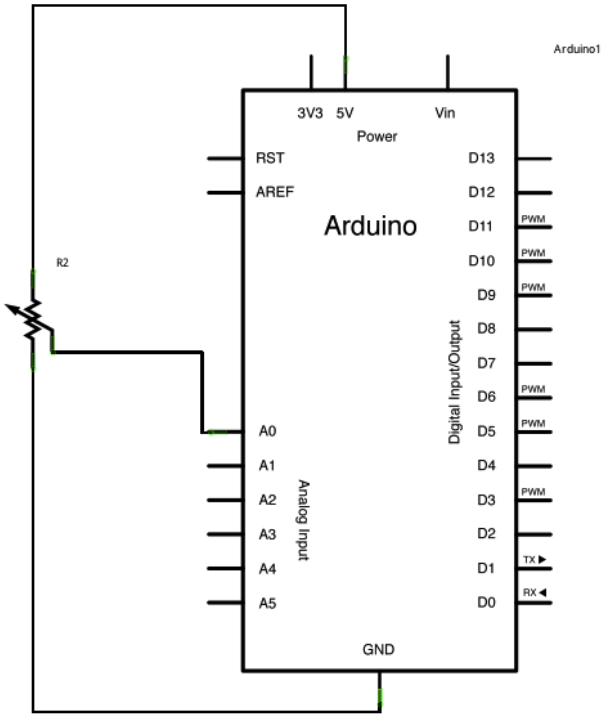
Scrive il valore della variabile sul monitor seriale



Lettura del valore di un potenziometro







ARDUINO WORKSHOP

Dal valore di un potenziometro all'uscita PWM – Variare la luminosità di un LED

AnalogInOutSerial

```
/*
  Analog input, analog output, serial output

 Reads an analog input pin, maps the result to a range from 0 to 255
 and uses the result to set the pulsewidth modulation (PWM) of an output pin.
 Also prints the results to the Serial Monitor.

 The circuit:
 * potentiometer connected to analog pin 0.
   Center pin of the potentiometer goes to the analog pin.
   side pins of the potentiometer go to +5V and ground
 * LED connected from digital pin 9 to ground

 created 29 Dec. 2008
 modified 9 Apr 2012
 by Tom Igoe

 This example code is in the public domain.

 */

// These constants won't change.  They're used to give names
// to the pins used:
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}
```

Scrive il valore di PWM nel pin di uscita, cioè imposta la larghezza dell'impulso generato

```
void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(2);
}
```

Rimappa i valori da 0..1023 a 0..255

print senza «a capo»

\t → tab

print e «a capo»

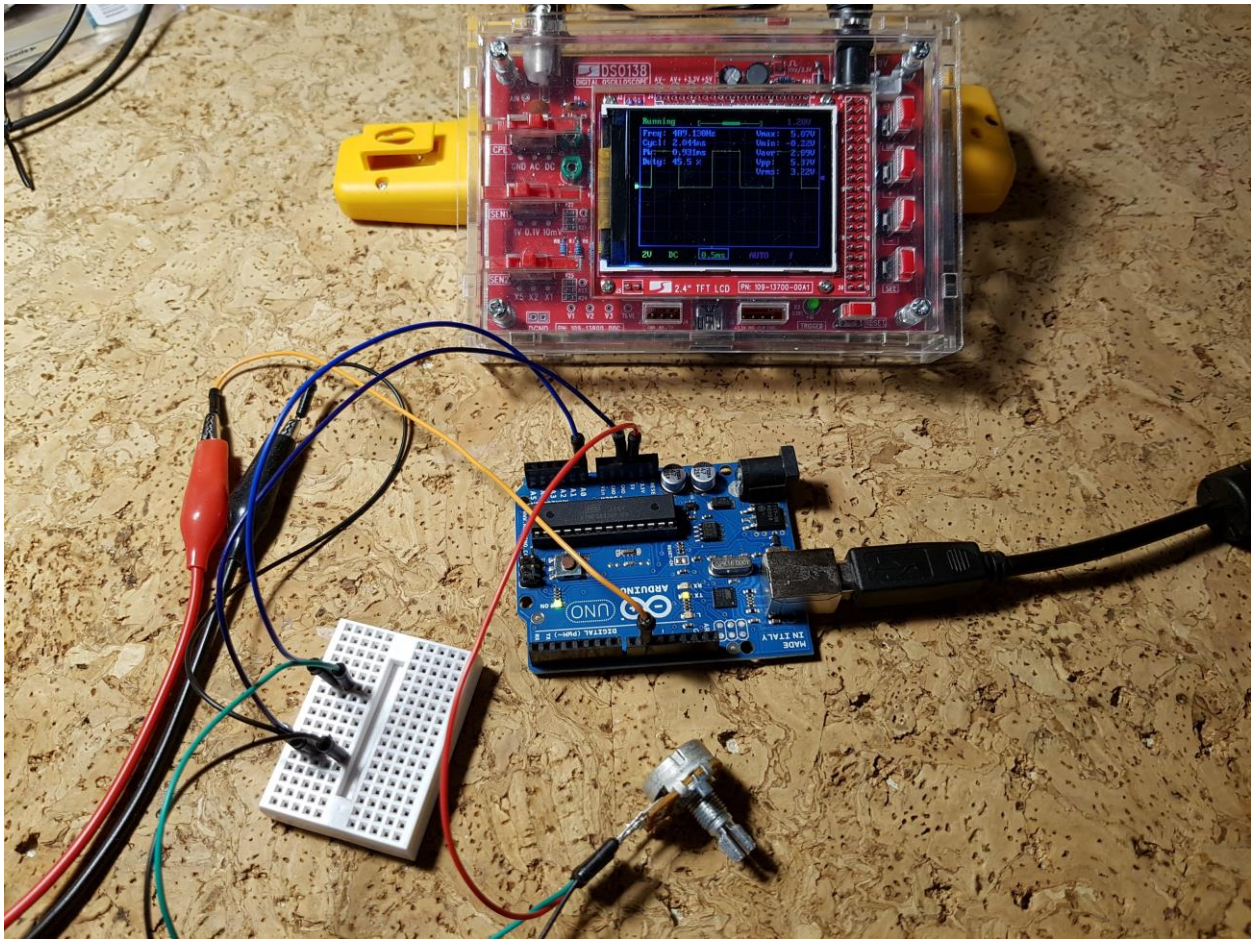
COM21 (Arduino Uno)

```
sensor = 470    output = 117
sensor = 469    output = 116
sensor = 469    output = 116
sensor = 470    output = 117
sensor = 470    output = 117
```

- Potenziometro tra massa e 5V
- Pin centrale pot. a ingresso A0
- Uscita pin 9 verso LED (con resistenza di 1K)
- Uscita 9 verso oscilloscopio

ARDUINO WORKSHOP

Dal valore di un potenziometro all'uscita PWM – Variare la luminosità di un LED



Uscita PWM continuamente variabile

_03_Fade

```
/*
 Fade

 This example shows how to fade an LED on pin 9
 using the analogWrite() function.

 This example code is in the public domain.
 */

int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}
```

Aumenta o diminuisce la luminosità (PWM)

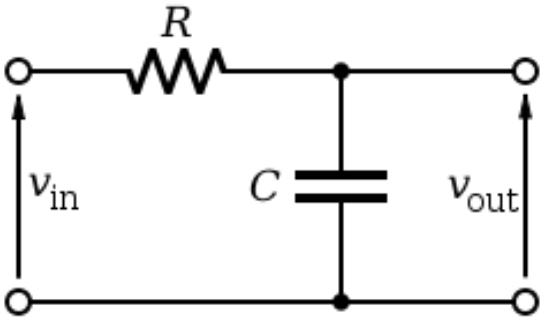
```
// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

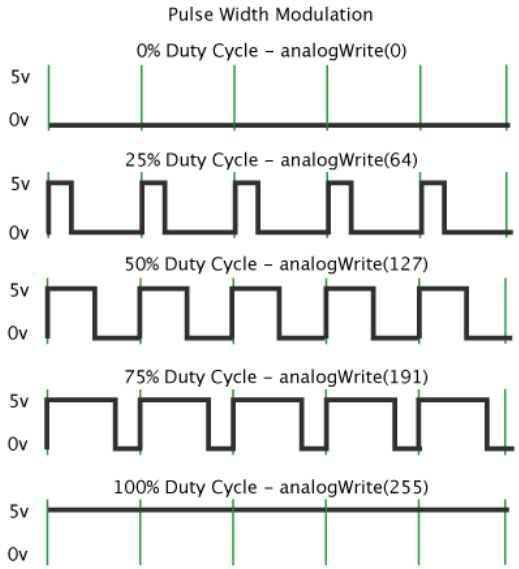
Cambia in positivo o negativo l'incremento di luminosità (PWM)

- Uscita pin 9 verso LED (con resistenza di 1K)
- Uscita 9 verso oscilloscopio

Dal segnale PWM a una tensione continua. «Poor man DAC»

In Arduino (tranne che nel modello DUE) non sono presenti DAC a bordo, ma solo ingressi ADC. Utilizzando le uscite PWM è però possibile ottenere una tensione continua. La frequenza del modulatore PWM di Arduino è di circa 490 Hz, ma può essere modificata tramite alcuni registri interni. Supponendo comunque di lasciare la frequenza a 490 Hz, un filtro passa-basso posto all'uscita del pin permette di ottenere una tensione continua, da utilizzare per usi vari, sopperendo alla mancanza di un vero e proprio DAC (li utilizzeremo a breve), ma con alcune limitazioni. Utilizziamo un filtro passa-basso del primo ordine, costruito nel modo più semplice, ovvero con una resistenza e un condensatore, posto all'uscita di un pin con segnale PWM. Arduino permette di variare la larghezza degli impulsi con una risoluzione di 8 bit (256 valori), da 0% a 100%, ovvero si hanno 256 possibili larghezze dell'impulso. Si può simulare l'effetto del filtraggio passa-basso per capire i limiti di questa soluzione, in questa pagina http://sim.okawa-denshi.jp/en/PWMtool.php





ARDUINO WORKSHOP

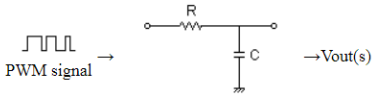
OKAWA Electric Design

Top > Tools > Filters > RC Low-pass Filter Design for PWM > Result

RC Low-pass Filter Design for PWM - Result -

Calculated peak-to-peak ripple voltage and settling time at a given PWM frequency and cut-off frequency or values of R and C.

RC Filter



Transfer Function:

$$G(s)=\frac{666.66666666667}{s+666.66666666667}$$

Cut-off frequency

fc = 106.1032953946[Hz]

Final Vout value of the step response (without a ripple)

f_{PWM}= 490 Hz

Duty Step 0%→ 50 [%]

PWM signal voltage:

V_L = 0 [V] V_H = 5 [V]

R and C values of filter | Cut-off frequency

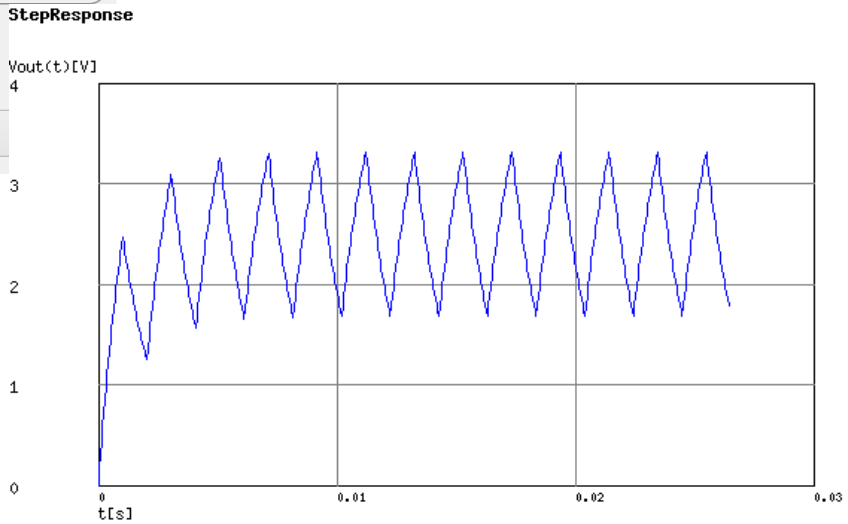
○ Cut-off frequency fc = [Hz]

◉ R and C values

R= 15000 Ω C= 0.1u F

p:pico, n:nano, u:micro, k:kilo, M:mega

Calculate



- Freq. di taglio circa 100 Hz
- veloce ad arrivare a regime
- eccessivo ripple sull'uscita

http://sim.okawa-denshi.jp/en/PWMtool.php

https://en.wikipedia.org/wiki/Low-pass_filter

ARDUINO WORKSHOP

OKAWA Electric Design

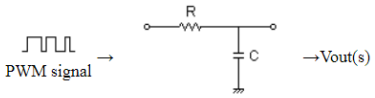
English | Ja

Top > Tools > Filters > RC Low-pass Filter Design for PWM > Result

RC Low-pass Filter Design for PWM - Result -

Calculated peak-to-peak ripple voltage and settling time at a given PWM frequency and cut-off frequency or values of R and C.

RC Filter



Transfer Function:

$$G(s)=\frac{66.666666666667}{s+66.666666666667}$$

Cut-off frequency

fc = 10.61032953946[Hz]

Final Vout value of the step response (without a ripple)

f_{PWM}= 490 Hz

Duty Step 0%→ 50 [%]

PWM signal voltage:

V_L = 0 [V] V_H = 5 [V]

R and C values of filter | Cut-off frequency

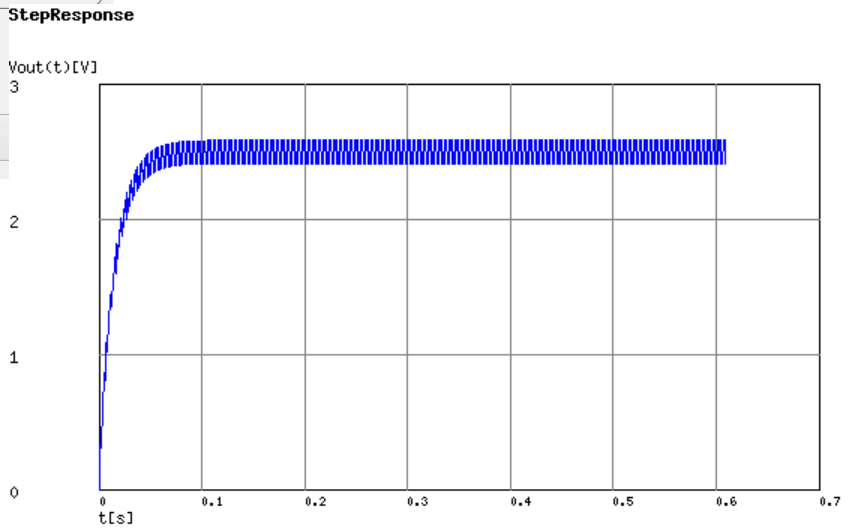
○ Cut-off frequency fc = [Hz]

◉ R and C values

R= 15000 Ω C= 1u F

p:pico, n:nano, u:micro, k:kilo, M:mega

Calculate



- Freq. di taglio circa 10 Hz
- meno veloce ad arrivare a regime
- ripple sull'uscita non trascurabile

http://sim.okawa-denshi.jp/en/PWMtool.php

https://en.wikipedia.org/wiki/Low-pass_filter

(c)okawa-denshi.jp

ARDUINO WORKSHOP

- Freq. di taglio circa 1 Hz
- lento ad arrivare a regime
- ripple trascurabile sull'uscita

OKAWA Electric Design

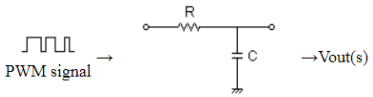
English

Top > Tools > Filters > RC Low-pass Filter Design for PWM > Result

RC Low-pass Filter Design for PWM - Result -

Calculated peak-to-peak ripple voltage and settling time at a given PWM frequency and cut-off frequency or values of R and C.

RC Filter



Transfer Function:

$$G(s)=\frac{6.6666666666667}{s+6.6666666666667}$$

Cut-off frequency

fc = 1.061032953946[Hz]

f_{PWM}= 490 Hz

Duty Step 0%→ 50 [%]

PWM signal voltage:

V_L = 0 [V] V_H = 5 [V]

R and C values of filter | Cut-off frequency

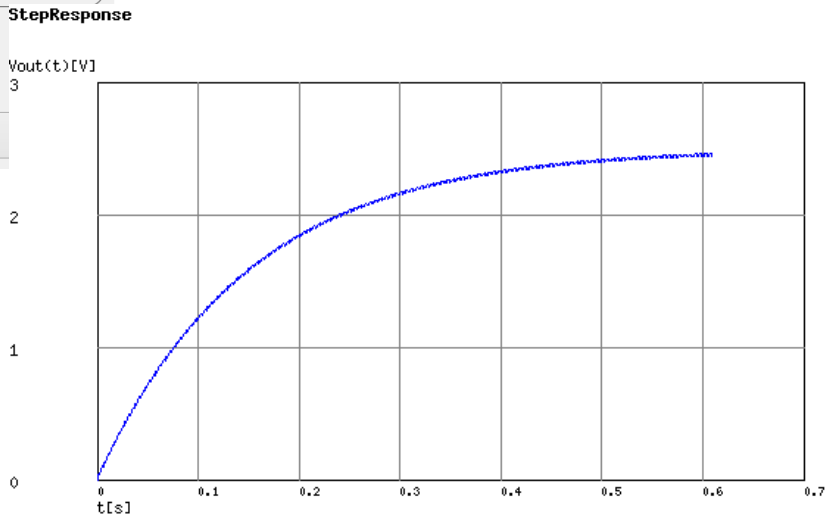
○ Cut-off frequency fc = [Hz]

◉ R and C values

R= 15000 Ω C= 10u F

p:pico, n:nano, u:micro, k:kilo, M:mega

Calculate



http://sim.okawa-denshi.jp/en/PWMtool.php

https://en.wikipedia.org/wiki/Low-pass_filter

(c)okawa-denshi.jp

ARDUINO WORKSHOP

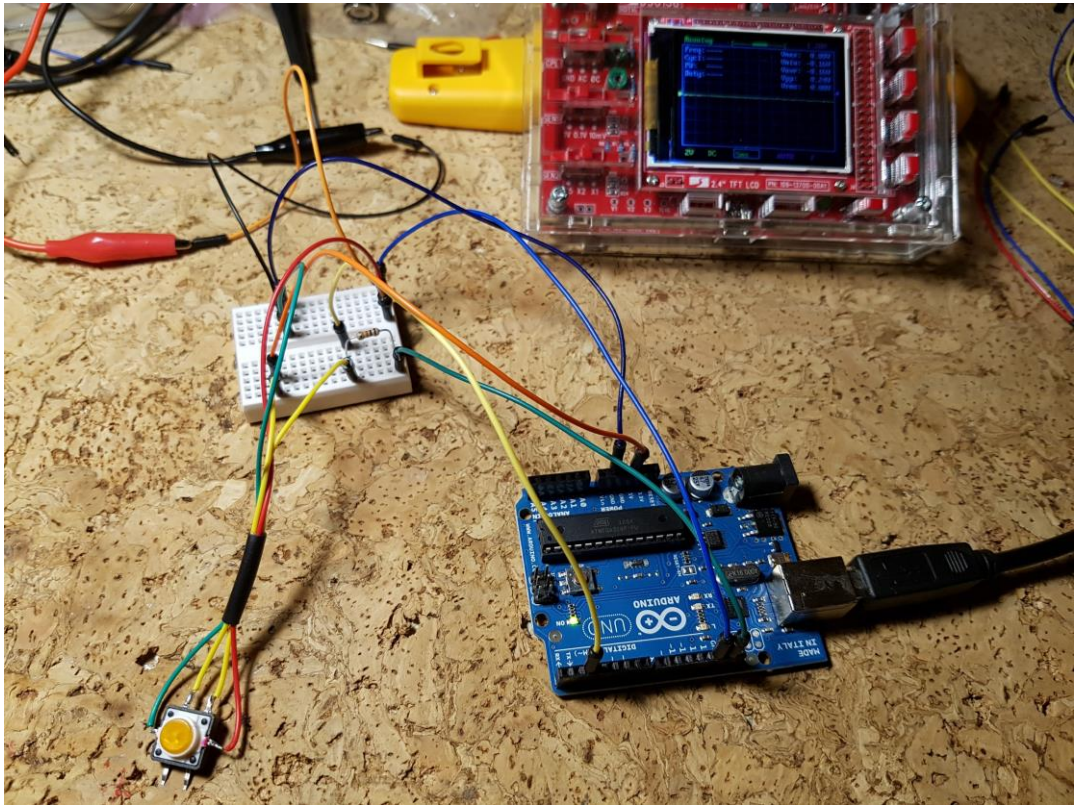
Utilizzo di un pulsante. Resistenze di pull-up e pull-down

Pulsante con LED incorporato

- Il LED è collegato al pin 13

- Il pulsante collega il pin 2 ai 5 V quando viene premuto

- **Una resistenza da 10K è collegata tra il pin 2 e massa**



PERCHE'?

Utilizzo di un pulsante. Resistenze di pull-up e pull-down

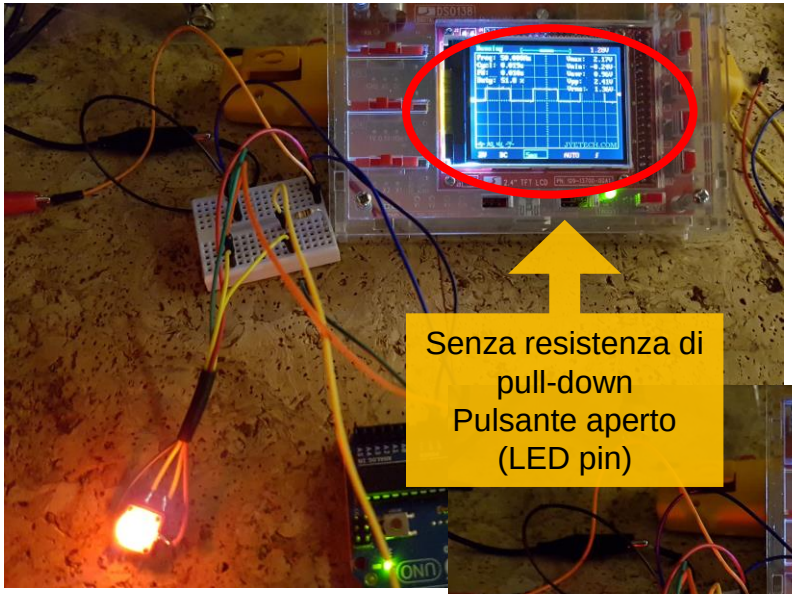
_04_Button

```
/*
  Button

 Turns on and off a light emitting diode(LED) connected to digital
 pin 13, when pressing a pushbutton attached to pin 2.


 The circuit:
 * LED attached from pin 13 to ground
 * pushbutton attached to pin 2 from +5V
 * 10K resistor attached to pin 2 from ground

 * Note: on most Arduinos there is already an LED on the board
 attached to pin 13.


 created 2005
 by DojoDave <http://www.0j0.org>
 modified 30 Aug 2011
 by Tom Igoe

 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/Button
 */

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
```

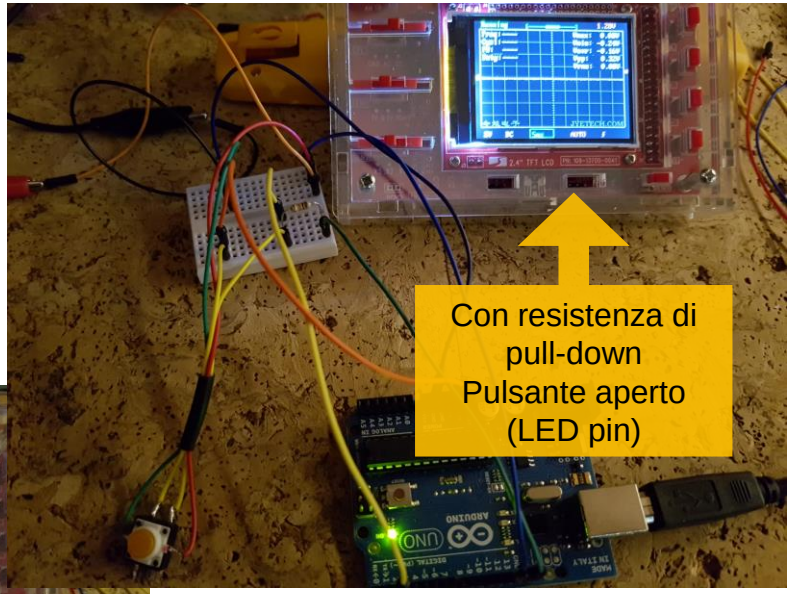
```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // 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) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

ARDUINO WORKSHOP

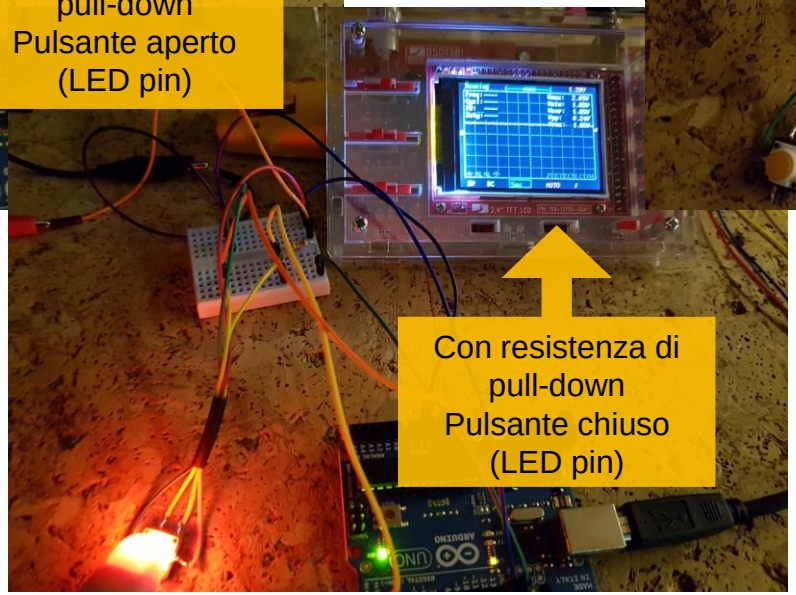
Utilizzo di un pulsante. Resistenze di pull-up e pull-down



Senza resistenza di pull-down
Pulsante aperto
(LED pin)

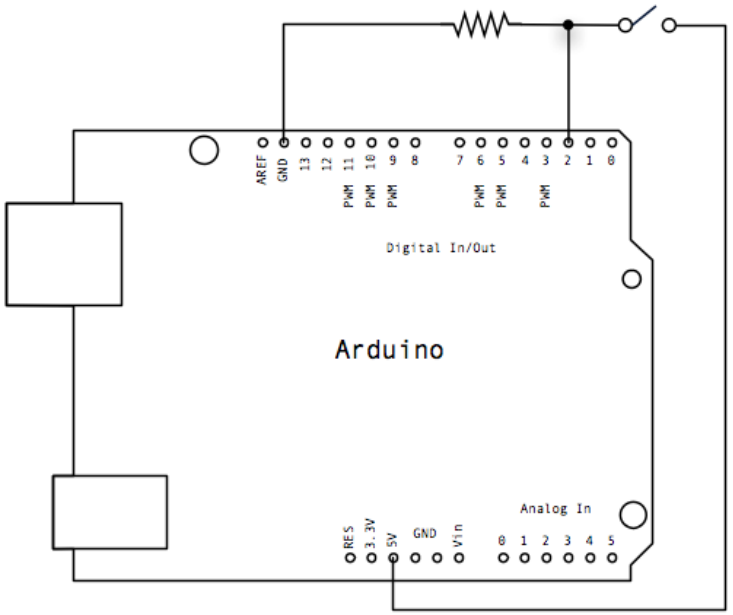


Con resistenza di pull-down
Pulsante aperto
(LED pin)



Con resistenza di pull-down
Pulsante chiuso
(LED pin)

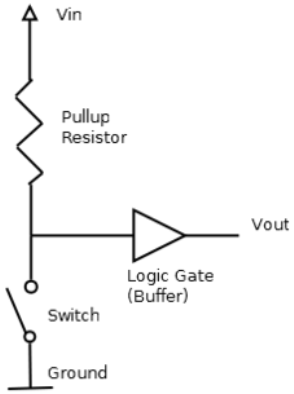
Resistenze di pull-up e pull-down



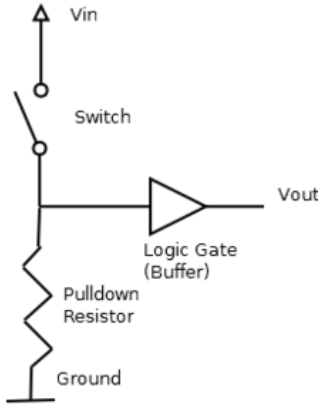
L'ingresso digitale, connesso al pulsante, richiede l'utilizzo di una resistenza aggiuntiva, collegata o a massa o ai +5V, a seconda del circuito. Questo perché, in assenza della stessa, quando il circuito è aperto (ovvero se il pulsante non è premuto) l'ingresso, ad alta impedenza, risulta in uno stato indefinito, captando disturbi (ad es. i 50 Hz di rete). Per cui con la resistenza si definisce uno **stato «di default» all'ingresso del pin**, che il pulsante porta a 0 o +5V. L'uso di una resistenza (es. 10 KOhm) e non di un collegamento diretto assicura che non ci siano cortocircuiti quando il pulsante viene premuto. Quindi, nel circuito dell'esempio (resistenza di pull-down), quando il pulsante è aperto al pin 2 ci sono 0V (massa), quando il pulsante viene premuto al pin 2 ci sono 5 V.

Resistenze di pull-up e pull-down

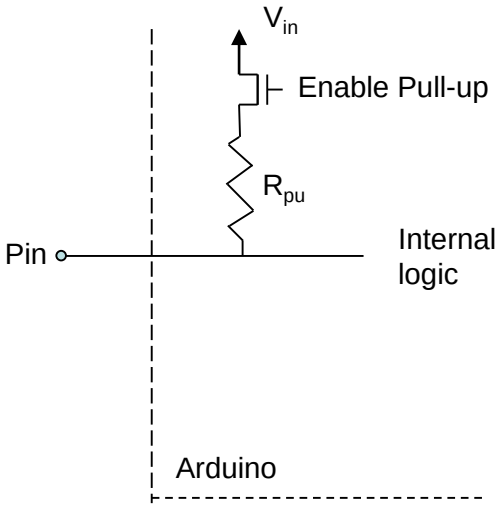
Pull-up



Pull-down



Internal Pull-up



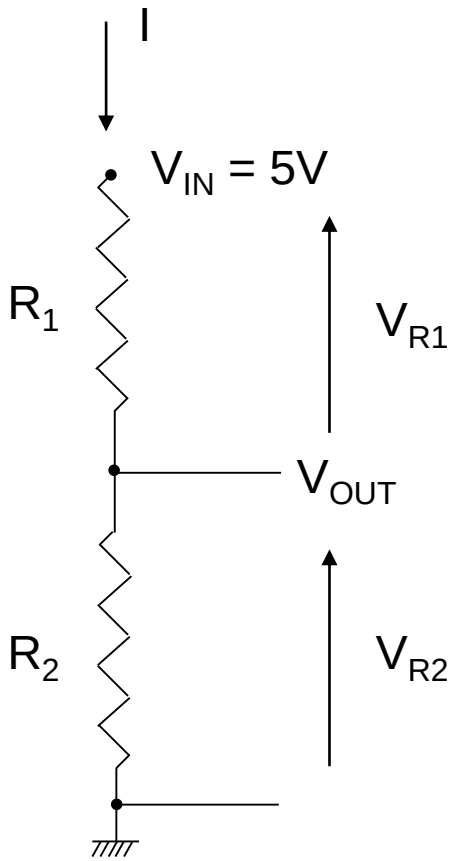
NOTA: come visto in precedenza, la funzione pinMode ha tra i suoi vari *mode* l'opzione INPUT oppure OUTPUT oppure **INPUT_PULLUP**.

Quindi è possibile impostare via software un pullup interno al microcontrollore (circa 20K). Ma questa operazione è (elettricamente) identica a impostare il livello logico alto del pin (quando in uso come output). Ovvero:

```
pinMode(pin, INPUT);
digitalWrite(pin, HIGH);
```

```
pinMode(pin, INPUT_PULLUP)
```

Il partitore di tensione



$$V_{R1} = R_1 \cdot I$$
$$V_{R2} = R_2 \cdot I$$

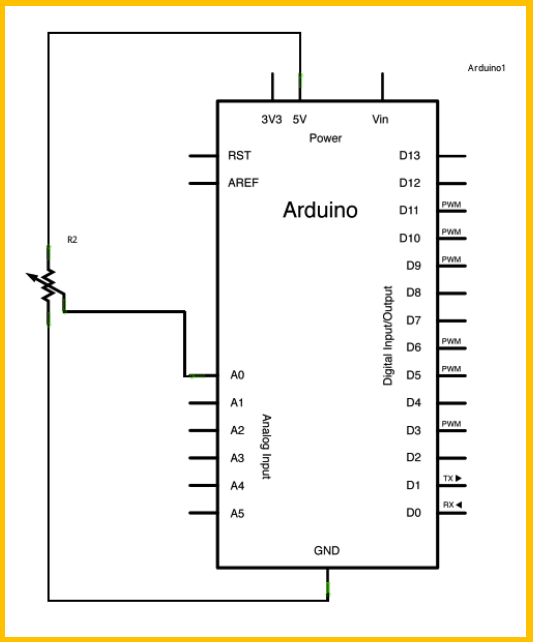
$$V_{IN} = (R_1 + R_2) \cdot I$$

$$I = \frac{V_{IN}}{R_1 + R_2}$$

$$V_{R1} = V_{IN} \cdot \frac{R_1}{R_1 + R_2}$$

$$V_{R2} = V_{IN} \cdot \frac{R_2}{R_1 + R_2} = V_{OUT}$$

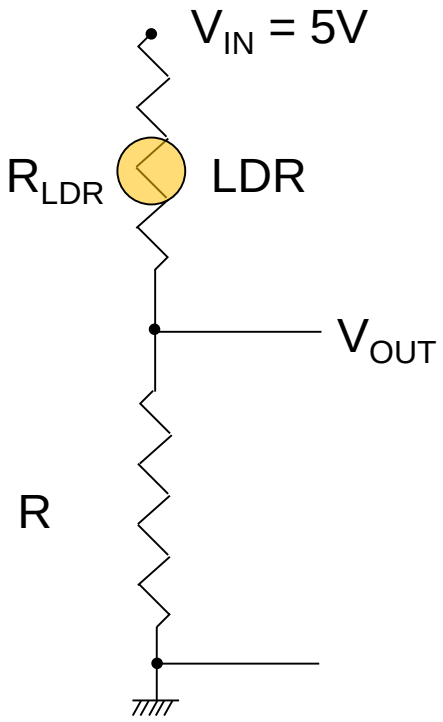
Il potenziometro è un partitore di tensione:
- R_2=0 → V_{R2}=0
- R_1=0 → V_{R2}=VIN
- R_1=R_2 → V_{R2}=V_{IN}/2

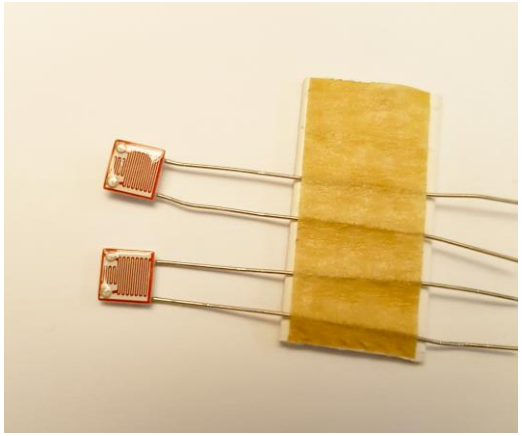


→ Resistenze pull-up e pull-down

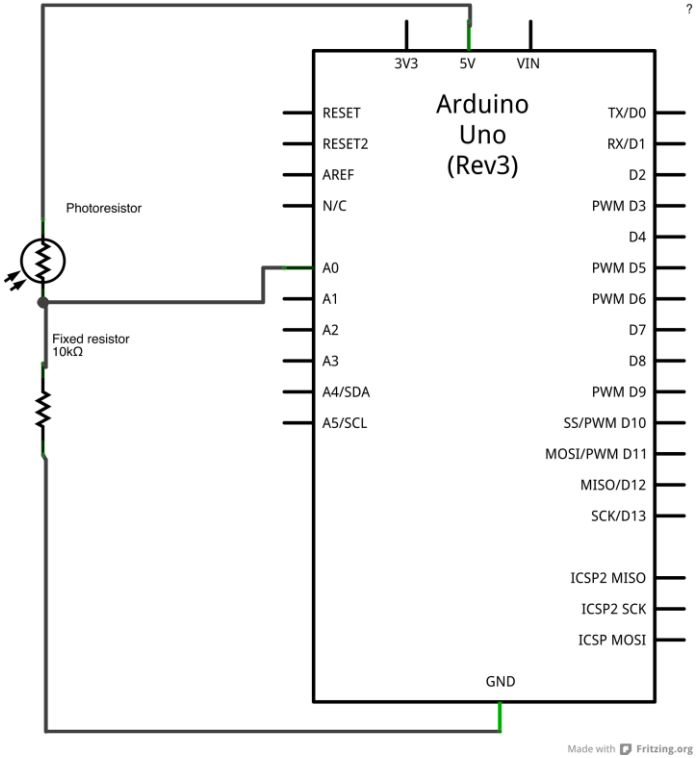
ARDUINO WORKSHOP

Altri sensori di tipo resistivo. Es. fotocellula





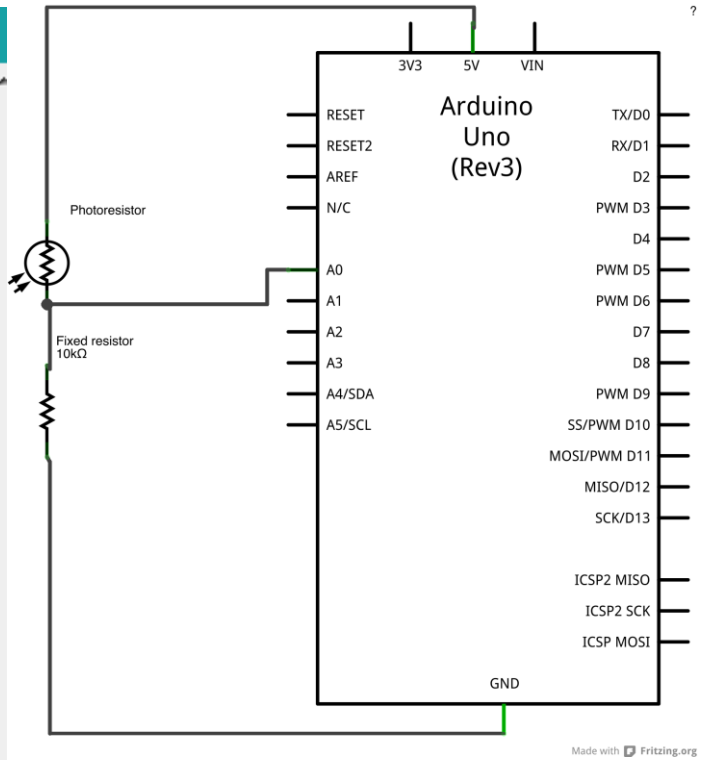
$$V_{OUT} = V_{IN} \cdot \frac{R}{R_{\mathrm{LDR}} + R}$$



Altri sensori di tipo resistivo. Es. fotocellula

_01_WS_AnalogReadSerial

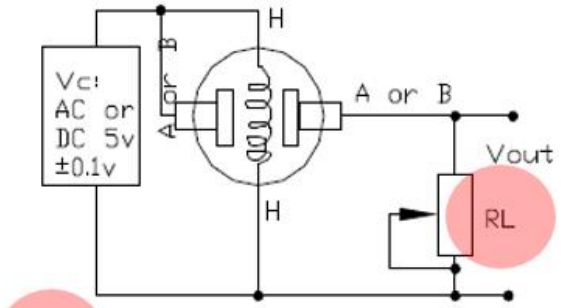
```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```



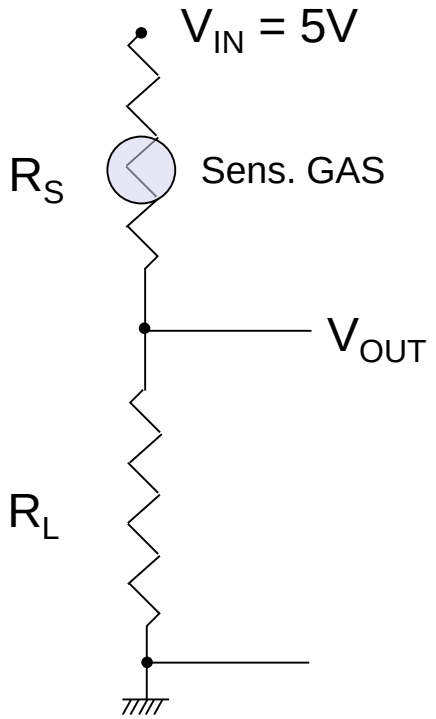
Proviamo sia con R = 10K che con R = 1K

$$V_{OUT} = V_{IN} \cdot \frac{R}{R_{\mathrm{LDR}} + R}$$

Altri sensori di tipo resistivo. Sensore di GAS

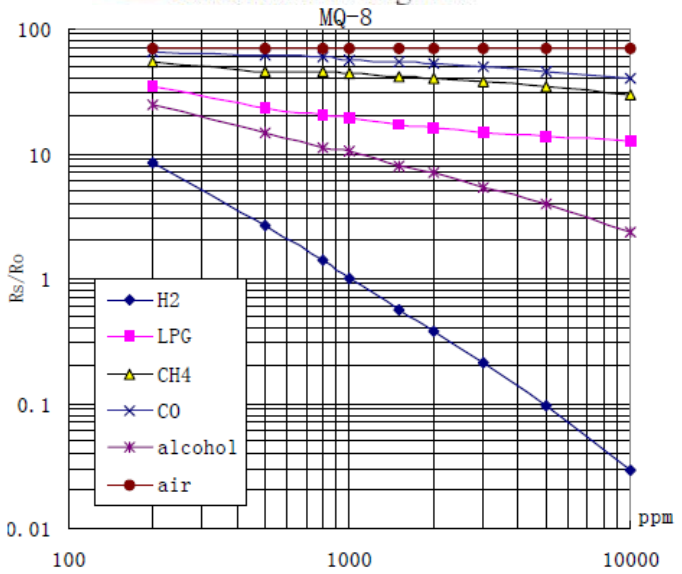


Ro: sensor resistance at 1000ppm of H_2 in the clean air.
Rs:sensor resistance at various concentrations of gases.

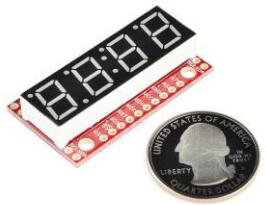


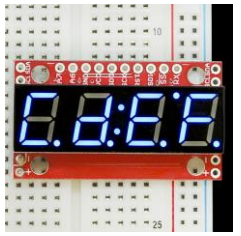
$$V_{OUT} = V_{IN} \cdot \frac{R_L}{R_S + R_L}$$

$$R_S = \frac{R_L \cdot (V_{IN} - V_{OUT})}{V_{OUT}}$$



Display a 7 segmenti (seriale)

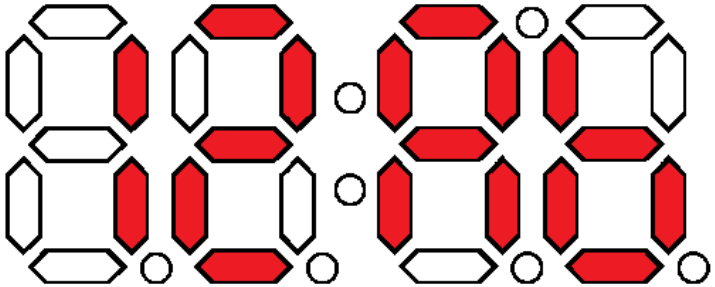




Per evitare di dover comandare ogni singolo segmento e renderne più semplice l'utilizzo, questo *shield* include un microscopico Arduino che comunica con l'Arduino principale e riceve comandi su cosa visualizzare. Può connettersi in 3 modi: seriale (TTL), seriale **SPI** o seriale **I²C**.

Arduino Sample Snippet (Serial Mode): To make the display read *12Ab*., we can't be guaranteed that the cursor is at position 1. To ensure that it is, we can use the clear display command before sending our data.

```
// ... after initializing Serial at the correct baud rate...
Serial.write(0x76);  // Clear display command, resets cursor
Serial.write(0x01);  // Hex value for 1, will display '1'
Serial.write('2');   // ASCII value for '2', will display '2'
Serial.write(0x0A);  // Hex value for 10, will display 'A'
Serial.write('B');   // ASCII value for 'B', will display 'b'
```

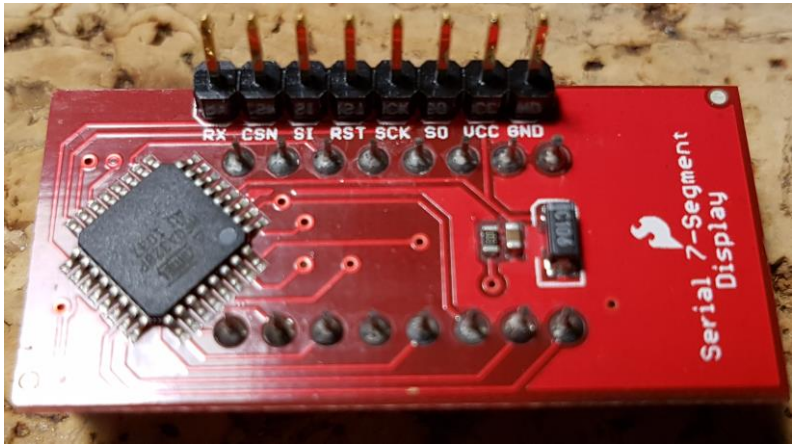


https://www.sparkfun.com/products/11442

ARDUINO WORKSHOP

Collegamento tramite SPI

Because we use the SPI library, you'll need to connect the Arduino's hardware SPI pins to the display.



Arduino Pin	Serial 7-Segment Display Pin
10 (CS)	SS (with a bar over it)
11 (MOSI)	SDI
13 (SCK)	SCK
5V	VCC
GND	GND

Connecting the displays *SDO* pin to MISO (12) on the Arduino is not required. Communication only goes one way - from Arduino (master) to display (slave).

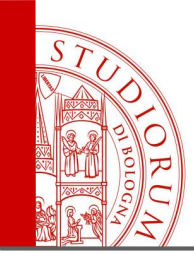
SPI: Serial Peripheral Interface

bus a 4 fili: **MOSI** (Master Out Slave In), **MISO** (Master In Slave Out), **SCK** (Clock), **SS** (Slave Select, **SS1**, **SS2**, .., **SSn**)

https://www.sparkfun.com/products/11442

```
#include <SPI.h>

int csPin = 10; //You can use any IO pin but for this example we use 10

int cycles = 0;

void setup()
{
  pinMode(csPin, OUTPUT);
  digitalWrite(csPin, HIGH); //By default, don't be selecting OpenSegment

  Serial.begin(9600); //Start serial communication at 9600 for debug statements
  Serial.println("OpenSegment Example Code");

  SPI.begin(); //Start the SPI hardware
  SPI.setClockDivider(SPI_CLOCK_DIV64); //Slow down the master a bit

  //Send the reset command to the display - this forces the cursor to
  //return to the beginning of the display
  digitalWrite(csPin, LOW); //Drive the CS pin low to select OpenSegment
  SPI.transfer('v'); //Reset command
}
```

https://www.sparkfun.com/products/11442



ARDUINO WORKSHOP


```
void loop()
{
  cycles++; //Counting cycles! Yay!
  Serial.print("Cycle: ");
  Serial.println(cycles);

  spiSendValue(cycles); //Send the four characters to the display

  delay(1); //If we remove the slow debug statements, we need a very small delay to prevent flickering
}

//Given a number, spiSendValue chops up an integer into four values and sends them out over spi
void spiSendValue(int tempCycles)
{
  digitalWrite(csPin, LOW); //Drive the CS pin low to select OpenSegment

  SPI.transfer(tempCycles / 1000); //Send the left most digit
  tempCycles %= 1000; //Now remove the left most digit from the number we want to display
  SPI.transfer(tempCycles / 100);
  tempCycles %= 100;
  SPI.transfer(tempCycles / 10);
  tempCycles %= 10;
  SPI.transfer(tempCycles); //Send the right most digit

  digitalWrite(csPin, HIGH); //Release the CS pin to de-select OpenSegment
}
```

https://www.sparkfun.com/products/11442

ARDUINO WORKSHOP

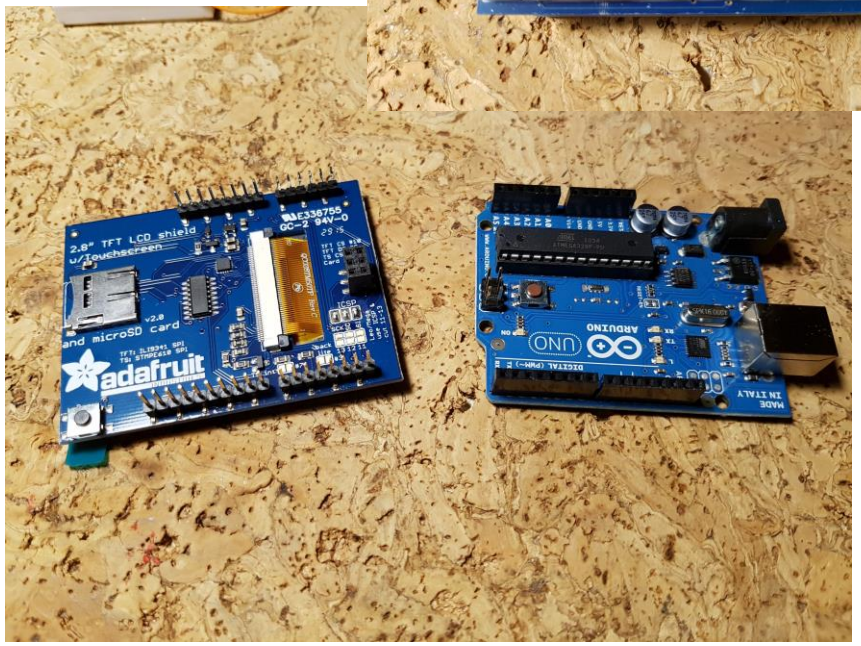
Display grafico TFT a colori (2.8”)

240 x 320 pixel, colori a 16 o 18 bit
Interfacciamento tramite SPI (della parte TFT) con UNO:

Clock dell’SPI: pin 13
MISO dell’SPI: pin 12
MOSI dell’SPI: pin 11
CS (chip select): pin 10
DS (data select): pin 9

Volendo si possono collegare anche il touch screen via I^2C e la microSD anche essa via SPI

Per il software: installare le librerie Adafruit_ILI9341 e Adafruit_GFX



https://www.adafruit.com/product/1651

ARDUINO WORKSHOP

_06_TFT28Adafruit §

```
/***************************************************
  This is an example sketch for the Adafruit 2.2" SPI display.
  This library works with the Adafruit 2.2" TFT Breakout w/SD card
  ----> http://www.adafruit.com/products/1480

  Check out the links above for our tutorials and wiring diagrams
  These displays use SPI to communicate, 4 or 5 pins are required to
  interface (RST is optional)
  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 ****************************************************/

#include "SPI.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9340.h"

// These are the pins used for the UNO
// for Due/Mega/Leonardo use the hardware SPI pins (which are different)
#define _sclk 13
#define _miso 12
#define _mosi 11
#define _cs 10
#define _dc 9
#define _rst 8
```

F(dice al compilatore di usare la stringa nella memoria flash

Pin di connessione SPI

```
void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("Adafruit 2.2\" SPI TFT Test!");

  tft.begin();

  Serial.println(F("Benchmark                Time (microseconds)"
  Serial.print(F("Screen fill              "));
  Serial.println(testFillScreen());
  delay(500);

  Serial.print(F("Text                     "));
  Serial.println(testText());
  delay(3000);

  Serial.print(F("Lines                    "));
  Serial.println(testLines(ILI9340_CYAN));
  delay(500);

  Serial.print(F("Horiz/Vert Lines         "));
  Serial.println(testFastLines(ILI9340_RED, ILI9340_BLUE));
  delay(500);

  Serial.print(F("Rectangles (outline)     "));
  Serial.println(testRects(ILI9340_GREEN));
  delay(500);

  Serial.print(F("Rectangles (filled)      "));
  Serial.println(testFilledRects(ILI9340_YELLOW, ILI9340_MAGENTA
```

https://www.sparkfun.com/products/11442

ARDUINO WORKSHOP


```
void loop(void) {
  for(uint8_t rotation=0; rotation<4; rotation++) {
    tft.setRotation(rotation);
    testText();
    delay(2000);
  }
}

unsigned long testText() {
  tft.fillScreen(ILI9340_BLACK);
  unsigned long start = micros();
  tft.setCursor(0, 0);
  tft.setTextColor(ILI9340_WHITE);  tft.setTextSize(1);
  tft.println("Hello World!");
  tft.setTextColor(ILI9340_YELLOW); tft.setTextSize(2);
  tft.println(1234.56);
  tft.setTextColor(ILI9340_RED);    tft.setTextSize(3);
  tft.println(0xDEADBEEF, HEX);
  tft.println();
  tft.setTextColor(ILI9340_GREEN);
  tft.setTextSize(5);
  tft.println("Groop");
  tft.setTextSize(2);
  tft.println("I implore thee,");
  tft.setTextSize(1);
  tft.println("my foonting turlingdromes.");
  tft.println("And hooptiously drangle me");
  tft.println("with crinkly bindlewurdles,");
  tft.println("Or I will rend thee");
  tft.println("in the gobberwarts");
  tft.println("with my blurglecruncheon,");
  tft.println("see if I don't!");
  return micros() - start;
}
```

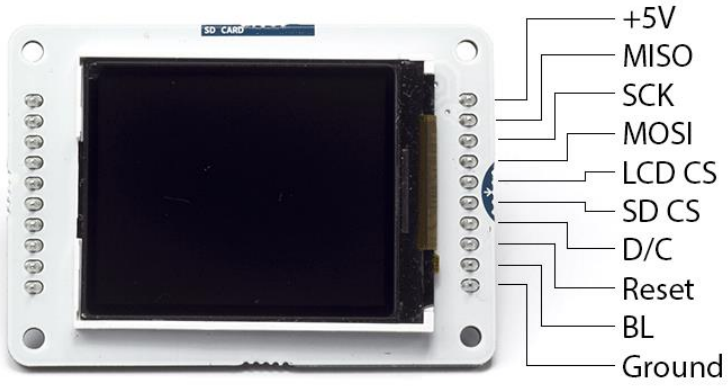
Funzioni varie per il test delle istruzioni grafiche

```
unsigned long testLines(uint16_t color) {
  unsigned long start, t;
  int           x1, y1, x2, y2,
                w = tft.width(),
                h = tft.height();

  tft.fillScreen(ILI9340_BLACK);

  x1 = y1 = 0;
  y2    = h - 1;
  start = micros();
  for(x2=0; x2<w; x2+=6) tft.drawLine(x1, y1, x2, y2, color);
  x2    = w - 1;
  for(y2=0; y2<h; y2+=6) tft.drawLine(x1, y1, x2, y2, color);
  t     = micros() - start; // fillScreen doesn't count against ti

  tft.fillScreen(ILI9340_BLACK);

  x1    = w - 1;
  y1    = 0;
  y2    = h - 1;
  start = micros();
  for(x2=0; x2<w; x2+=6) tft.drawLine(x1, y1, x2, y2, color);
  x2    = 0;
  for(y2=0; y2<h; y2+=6) tft.drawLine(x1, y1, x2, y2, color);
  t    += micros() - start;

  tft.fillScreen(ILI9340_BLACK);

  x1    = 0;
  y1    = h - 1;
  y2    = 0;
  start = micros();
  for(x2=0; x2<w; x2+=6) tft.drawLine(x1, y1, x2, y2, color);
  x2    = w - 1;
  for(y2=0; y2<h; y2+=6) tft.drawLine(x1, y1, x2, y2, color);
  t    += micros() - start;
```

https://www.sparkfun.com/products/11442

ARDUINO WORKSHOP

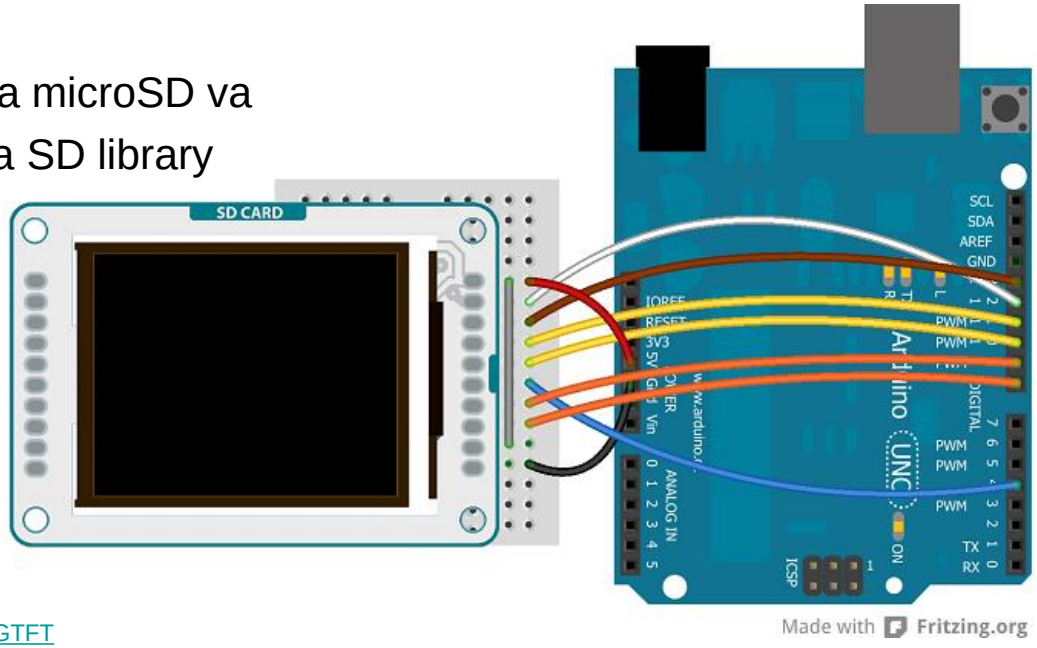
Collegamento di un altro display grafico TFT (1.8”)



160 x 128 pixel
Interfacciamento tramite SPI

Per funzionare, vanno installate le librerie Adafruit GFX e Adafruit ST7735

Per usare anche la microSD va installata la libreria SD library



+5V:	+5V
MISO:	pin 12
SCK:	pin 13
MOSI:	pin 11
LCD CS:	pin 10
SD CS:	pin 4
D/C:	pin 9
RESET:	pin 8
BL:	+5V
GND:	GND

https://www.arduino.cc/en/Main/GTFT

ARDUINO WORKSHOP

_07_TFTPong_TFT18 §

```
This example for the Arduino screen reads the values
of 2 potentiometers to move a rectangular platform
on the x and y axes. The platform can intersect
with a ball causing it to bounce.

This example code is in the public domain.

Created by Tom Igoe December 2012
Modified 15 April 2013 by Scott Fitzgerald

http://arduino.cc/en/Tutorial/TFTPong

*/

#include <TFT.h>  // Arduino LCD library
#include <SPI.h>

// pin definition for the Uno
#define cs   10
#define dc   9
#define rst  8

TFT TFTscreen = TFT(cs, dc, rst);

// variables for the position of the ball and paddle
int paddleX = 0;
int paddleY = 0;
int oldPaddleX, oldPaddleY;
int ballDirectionX = 1;
int ballDirectionY = 1;

int ballSpeed = 10; // lower numbers are faster

int ballX, ballY, oldBallX, oldBallY;
```

```
void loop() {

  // save the width and height of the screen
  int myWidth = TFTscreen.width();
  int myHeight = TFTscreen.height();

  // map the paddle's location to the position of the potentiometers
  paddleX = map(analogRead(A0), 0, 1023, 0, myWidth) - 20 / 2;
  paddleY = map(analogRead(A1), 0, 1023, 0, myHeight) - 5 / 2;

  // set the fill color to black and erase the previous
  // position of the paddle if different from present
  TFTscreen.fill(0, 0, 0);

  if (oldPaddleX != paddleX || oldPaddleY != paddleY) {
    TFTscreen.rect(oldPaddleX, oldPaddleY, 20, 5);
  }

  // draw the paddle on screen, save the current position
  // as the previous.
  TFTscreen.fill(255, 255, 255);

  TFTscreen.rect(paddleX, paddleY, 20, 5);
  oldPaddleX = paddleX;
  oldPaddleY = paddleY;

  // update the ball's position and draw it on screen
  if (millis() % ballSpeed < 2) {
    moveBall();
  }
}
```

```
// this function determines the ball's position on
void moveBall() {
  // if the ball goes offscreen, reverse the direc
  if (ballX > TFTscreen.width() || ballX < 0) {
    ballDirectionX = -ballDirectionX;
  }
```

https://www.arduino.cc/en/Main/GTFT

ARDUINO WORKSHOP

_08_TFTGraph_TFT18

```
/*

 TFT Graph

 This example for an Arduino screen reads
 the value of an analog sensor on A0, and
 graphs the values on the screen.

 This example code is in the public domain.

 Created 15 April 2013 by Scott Fitzgerald

 http://arduino.cc/en/Tutorial/TFTGraph

 */

#include <TFT.h>  // Arduino LCD library
#include <SPI.h>

// pin definition for the Uno
#define cs   10
#define dc   9
#define rst  8

// pin definition for the Leonardo
// #define cs   7
// #define dc   0
// #define rst  1

TFT TFTscreen = TFT(cs, dc, rst);

// position of the line on screen
int xPos = 0;
```

```
void setup() {
  // initialize the serial port
  Serial.begin(9600);

  // initialize the display
  TFTscreen.begin();

  // clear the screen with a pretty color
  TFTscreen.background(250, 16, 200);
}

void loop() {
  // read the sensor and map it to the screen height
  int sensor = analogRead(A0);
  int drawHeight = map(sensor, 0, 1023, 0, TFTscreen.height());

  // print out the height to the serial monitor
  Serial.println(drawHeight);

  // draw a line in a nice color
  TFTscreen.stroke(250, 180, 10);
  TFTscreen.line(xPos, TFTscreen.height() - drawHeight,
                                    xPos, TFTscreen.height());

  // if the graph has reached the screen edge
  // erase the screen and start again
  if (xPos >= 160) {
    xPos = 0;
    TFTscreen.background(250, 16, 200);
  }
  else {
    // increment the horizontal position:
    xPos++;
  }

  delay(16);
}
```

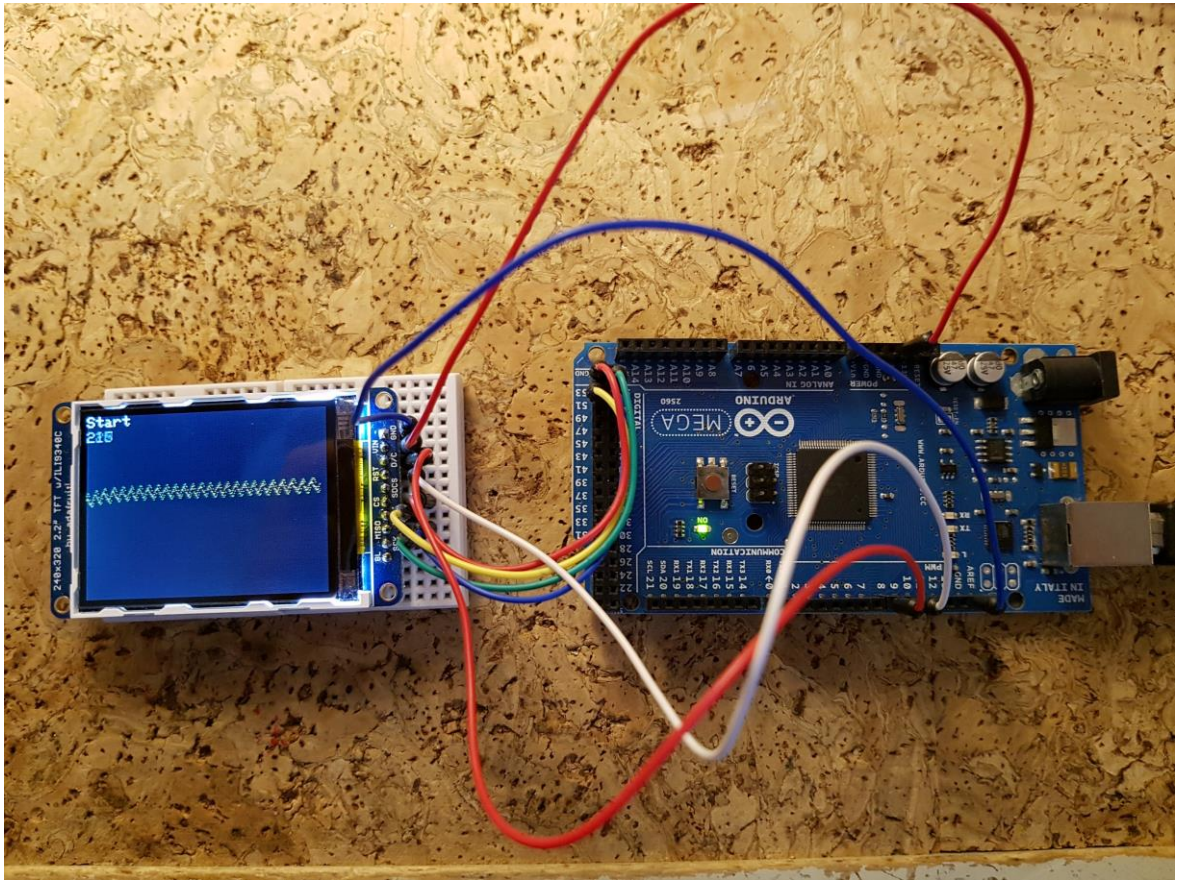
https://www.arduino.cc/en/Main/GTFT

ARDUINO WORKSHOP

Collegamento di un altro display grafico TFT (2.2”) usando Arduino MEGA2560

240 x 320 pixel, colori a 16 bit
Interfacciamento tramite SPI
ad Arduino MEGA2560:

SCK (Clock) dell’SPI: pin 52
MISO dell’SPI: pin 50
MOSI dell’SPI: pin 51
CS (chip select): pin 53
RST (reset): pin 9
DC (data/command select): pin 8



Per il software: installare le librerie
Adafruit_ILI9340 e Adafruit_GFX

https://www.adafruit.com/product/1480

ARDUINO WORKSHOP

_09_MyScopeTFT22§

```
#include "SPI.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9340.h"

#if defined(__SAM3X8E__)
    #undef __FlashStringHelper::F(string_literal)
    #define F(string_literal) string_literal
#endif

// These are the pins used for the Mega
#define _sclk 52
#define _miso 50
#define _mosi 51
#define _cs 53
#define _rst 9
#define _dc 8

#define BLACK 0x0000
#define BLUE 0x001F
#define RED 0xF800
#define GREEN 0x07E0
#define CYAN 0x07FF
#define MAGENTA 0xF81F
#define YELLOW 0xFFE0
#define WHITE 0xFFFF

Adafruit_ILI9340 tft = Adafruit_ILI9340(_cs, _dc, _rst);

// char array to print to the screen
char sensorPrintout[5];
int xPos = 1;
```

```
void setup() {
  // put your setup code here, to run once:

tft.begin();
tft.fillScreen(ILI9340_RED);
tft.fillScreen(ILI9340_BLACK);
tft.setRotation(1);
}

void loop() {
  // put your main code here, to run repeatedly:

 tft.setCursor(0, 0);
 tft.setTextColor(ILI9340_WHITE);  tft.setTextSize(2);
 tft.println("Start");

 String sensorVal1 = String(analogRead(A0));
 sensorVal1.toCharArray(sensorPrintout, 5);
 tft.setCursor(0, 20);
 tft.fillRect(0,20,50,28,ILI9340_BLACK);
 tft.println(sensorVal1);

// delay(100);

 tft.drawPixel(xPos, 50+analogRead(A0)/5,YELLOW); // draw a line across the screen
 tft.drawPixel(xPos, 50+analogRead(A1)/5,GREEN); // draw a line across the screen

  xPos = xPos + 1;
  if(xPos>=tft.width()) {
  xPos=0;
  tft.fillRect(0,50,tft.width(),tft.height(),ILI9340_BLACK);
 }
}
```

Oscilloscopio degli ingressi interni analogici A0 e A1

ARDUINO WORKSHOP

Aggiungiamo all'esempio precedente un convertitore DAC e un convertitore ADC esterni

Convertitore Digitale Analogico

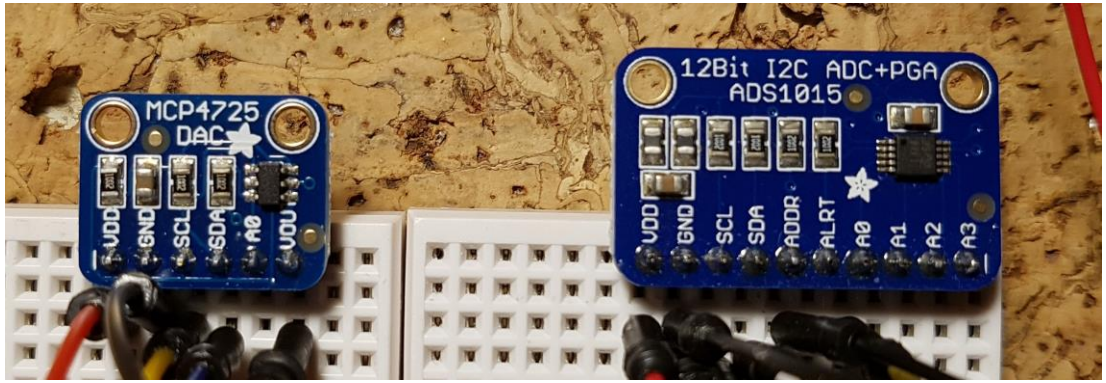
DAC **MCP4725** (12 bit)

https://learn.adafruit.com/mcp4725-12-bit-dac-tutorial

Convertitore Analogico Digitale (x4)

ADC **ADS1015** (12 bit)

https://learn.adafruit.com/adafruit-4-channel-adc-breakouts



Entrambi si collegano ad Arduino usando il protocollo **I²C**, chiaramente su indirizzi diversi.

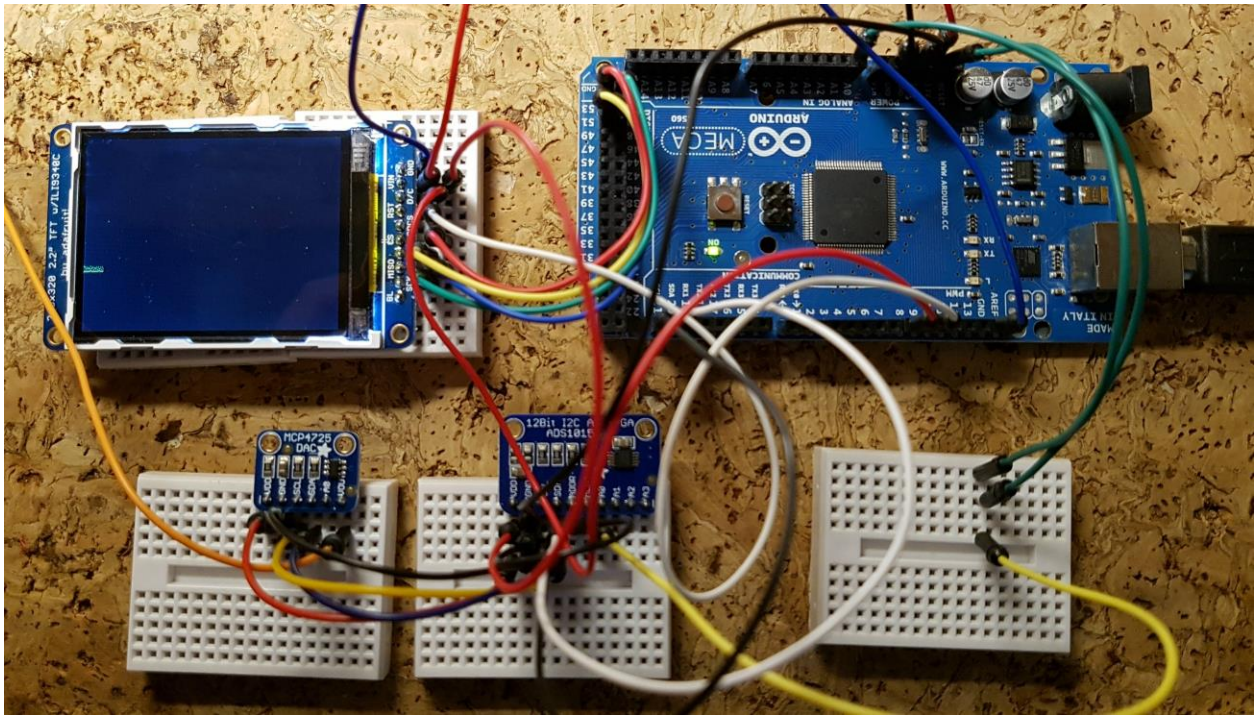
Usando Arduino MEGA i pin dedicati all'I²C sono il 20 (SDA) e il 21 (SCL), quindi entrambi i dispositivi andranno collegati in parallelo a questo bus. L'indirizzo viene fissato in modo hardware sulla rispettiva scheda:

MCP4725: A0 scollegato → indirizzo 0x62; A0 collegato a VDD → indirizzo 0x63

ADS1015: ADDR collegato a GND → indirizzo 0x48; ADDR collegato a VDD → indirizzo 0x49
ADDR collegato a SDA → indirizzo 0x4A; ADDR collegato a SCL → indirizzo 0x4B

ARDUINO WORKSHOP

Si ha ora quindi il display TFT collegato sul bus SPI e l'ADC e il DAC collegati entrambi sullo stesso bus I^2C ma con indirizzi diversi. Lo sketch caricato legge la tensione sull'ingresso analogico di Arduino sul piedino A0 e la mostra e disegna sullo schermo. Inoltre l'ADC esterno legge una tensione dal suo ingresso A0, questo valore viene disegnato sullo schermo e viene impostato il DAC esterno con la stessa tensione. Nella parte iniziale dello sketch vengono incluse tutte le librerie necessarie al funzionamento dei dispositivi usati. Nella parte di setup questi vengono inizializzati.



Il DAC **MCP4725** è impostato all'indirizzo 0x62 (il suo pin di selezione indirizzo è lasciato scollegato) mentre l'ADC **ADS1015** è impostato all'indirizzo 0x48 (il suo pin selezione indirizzo viene collegato a massa). Gli SDA di entrambi vanno al pin 20 di Arduino e gli SCL al pin 21.

ARDUINO WORKSHOP

_10_ADC_DAC_TFT22

```
#include "SPI.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9340.h"
#include <Wire.h>
#include <Adafruit_MCP4725.h>
#include <Adafruit_ADS1015.h>

Adafruit_MCP4725 dac;
Adafruit_ADS1015 ads1015(0x48);

// These are the pins used for the Mega
#define _sclk 52
#define _miso 50
#define _mosi 51
#define _cs 53
#define _rst 9
#define _dc 8

#define BLACK 0x0000
#define BLUE 0x001F
#define RED 0xF800
#define GREEN 0x07E0
#define CYAN 0x07FF
#define MAGENTA 0xF81F
#define YELLOW 0xFFE0
#define WHITE 0xFFFF

Adafruit_ILI9340 tft = Adafruit_ILI9340(_cs, _dc, _rst);

// char array to print to the screen
char sensorPrintout[20];
int xPos = 1;
```

```
void setup() {
  // put your setup code here, to run once:

tft.begin();
tft.fillScreen(ILI9340_RED);
tft.fillScreen(ILI9340_BLACK);
tft.setRotation(1);
tft.setTextSize(1);

// DAC
dac.begin(0x62);
tft.println("DAC MCP4725 Started");

//    GAIN_TWOTHIRDS (for an input range of +/- 6.144V)
//    GAIN_ONE (for an input range of +/-4.096V)
//    GAIN_TWO (for an input range of +/-2.048V)
//    GAIN_FOUR (for an input range of +/-1.024V)
//    GAIN_EIGHT (for an input range of +/-0.512V)
//    GAIN_SIXTEEN (for an input range of +/-0.256V)

ads1015.begin();  // Initialize ads1115
ads1015.setGain(GAIN_ONE);
if (! ads1015.getGain()==GAIN_ONE)
      { tft.println("Error ADC ads1115"); }
      else { tft.println("ADC ADS1115 Started"); }

delay(1000);
tft.fillScreen(ILI9340_BLACK);
tft.println("Loop start");
delay(1000);
}
```

Setup display TFT

Setup DAC

Setup ADC
Il range dipende dal GAIN

ARDUINO WORKSHOP


```
66  //-----------------------------------------------
67  //int16_t -32,768 to 32,767
68  int16_t adc0, adc1, adc2, adc3;
69  float VoltageExtADC,VoltageArdADC,VoltageExtDAC;
70  //-----------------------------------------------
71
72  void loop() {
73    // put your main code here, to run repeatedly:
74
75   tft.setCursor(0, 0);
76   tft.setTextColor(ILI9340_WHITE);
77
78   tft.print("Time ");
79   tft.println(millis());
80
81   VoltageArdADC=float(analogRead(A0))*5/1023; // 5V su 10 bit
82   String sensorV = String(VoltageArdADC,4); // Arduino ADC (10 bit)
83   sensorV.toCharArray(sensorPrintout, 20);
84   tft.print("Arduino IN  V=");
85   tft.println(sensorV);
86
87   tft.drawPixel(xPos, 240-VoltageArdADC*30,YELLOW); // plot ADC interno
88
89    xPos = xPos + 1;           // Avanzamento cursore plot
90    if(xPos>=tft.width()) {
91    xPos=0;
92    tft.fillRect(0,50,tft.width(),tft.height(),ILI9340_BLACK);
93   }
94
95  // ADC: 2047 per V=4.096 V
96   adc0 = ads1015.readADC_SingleEnded(0);
97
98  // DAC: 4095 per V=5 V;  2047 per V=2,5 V
99  // DAC: 4095 div 5 per V=1 V
100  dac.setVoltage(adc0*4.096/2.5, false); // V(DAC) = V(ADC)
101
102  VoltageExtADC=float(2*adc0)/1000; // per 1015
103  tft.drawPixel(xPos, 240-VoltageExtADC*30,WHITE); // plot ADC esterno
104
```

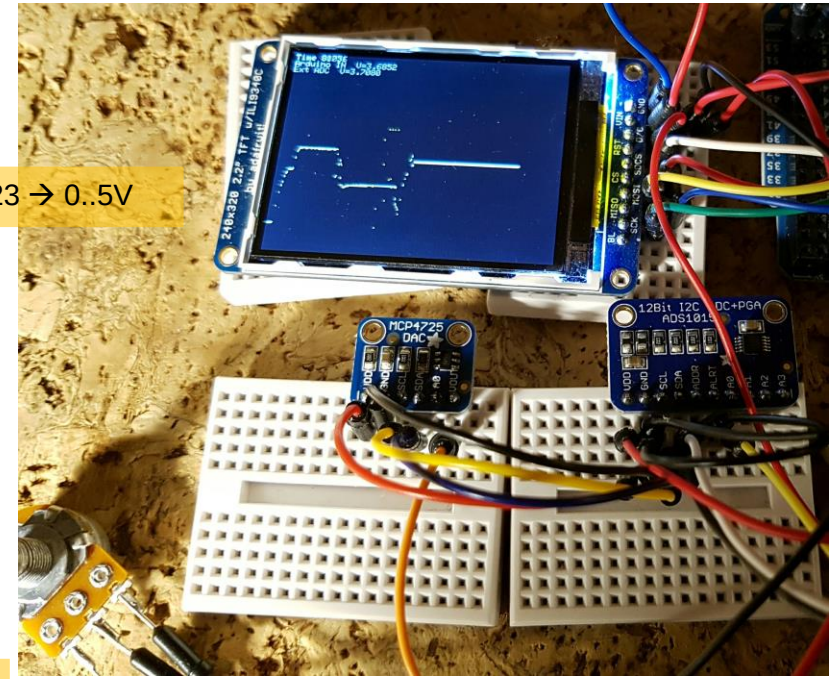
```
104
105   sensorV = String(VoltageExtADC,4); //(12 bit ADC)';
106   sensorV.toCharArray(sensorPrintout, 20);
107   tft.print("Ext ADC  V=");
108   tft.println(sensorV);
109   delay(200);
110   tft.fillRect(0,0,160,28,ILI9340_BLACK);
111 }
```

0..1023 → 0..5V

Lettura ADC

Scrittura DAC

Max 4,096 V su IN



Potenziometro (partitore) → ADC esterno
ADC esterno → DAC esterno → ADC interno
I grafici degli ADC int. e est. sono sovrapposti

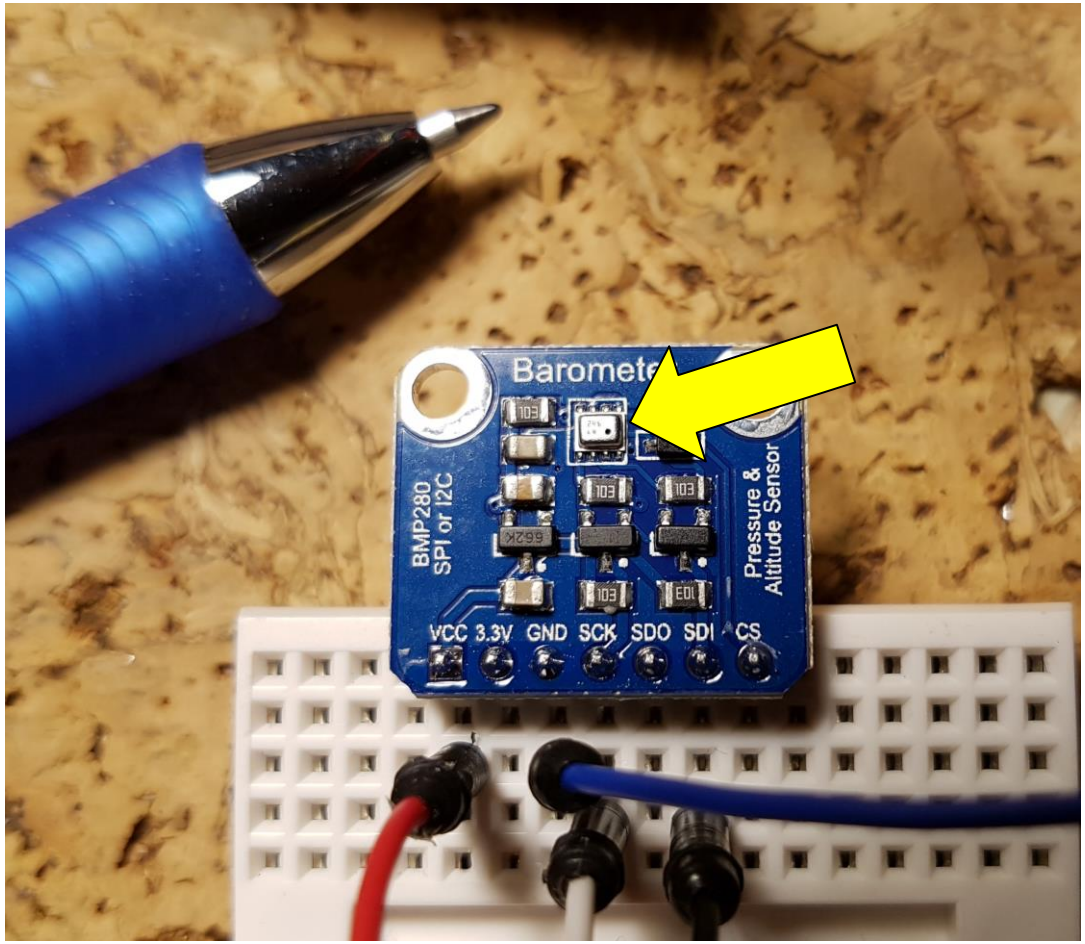
ARDUINO WORKSHOP

Sensore di temperatura e pressione

BMP280. Interfaccia I²C e SPI

Pressione:
Range: 300-1100 hPa
Risoluzione: 0.16 Pa
Rumore: 1.3 Pa

Temperatura:
Range: -40 / +85 °C
Risoluzione: 0.01 °C



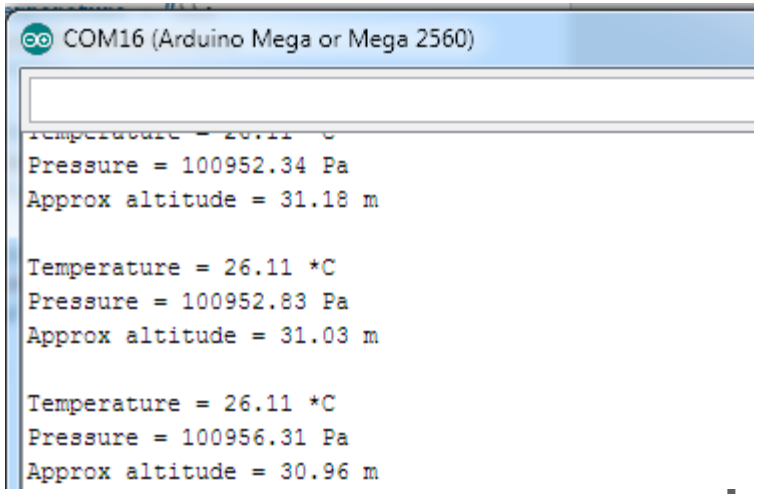
https://www.sunfounder.com/bmp280-barometric-pressure-temperature-altitude-sensor-module.html
https://www.bosch-sensortec.com/bst/products/all_products/bmp280

ARDUINO WORKSHOP

_11_BMP280_TempPressAlt

```
/***************************************************************************
  This is a library for the BMP280 humidity, temperature & pressure sensor

  Designed specifically to work with the Adafruit BMEP280 Breakout
  ----> http://www.adafruit.com/products/2651

  These sensors use I2C or SPI to communicate, 2 or 4 pins are required
  to interface.

  Adafruit invests time and resources providing this open source code,
  please support Adafruit andopen-source hardware by purchasing products
  from Adafruit!

  Written by Limor Fried & Kevin Townsend for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ***************************************************************************/

#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>

//Vin to 5V
//Gnd to Gnd
//SCK to SCL (21 on MEGA, A5 UNO)
//SDI to SDA (20 on MEGA, A4 UNO)

Adafruit_BMP280 bme; // I2C

void setup() {
  Serial.begin(9600);
  Serial.println(F("BMP280 test"));

  if (!bme.begin()) {
    Serial.println(F("Could not find a valid BMP280 sensor, check wiring!"));
    while (1);
  }
}
```

```
void loop() {
    Serial.print(F("Temperature = "));
    Serial.print(bme.readTemperature());
    Serial.println(" *C");

    Serial.print(F("Pressure = "));
    Serial.print(bme.readPressure());
    Serial.println(" Pa");

    Serial.print(F("Approx altitude = "));
    Serial.print(bme.readAltitude(1013.25));
    Serial.println(" m");

    Serial.println();
    delay(1000);
}
```



ARDUINO WORKSHOP

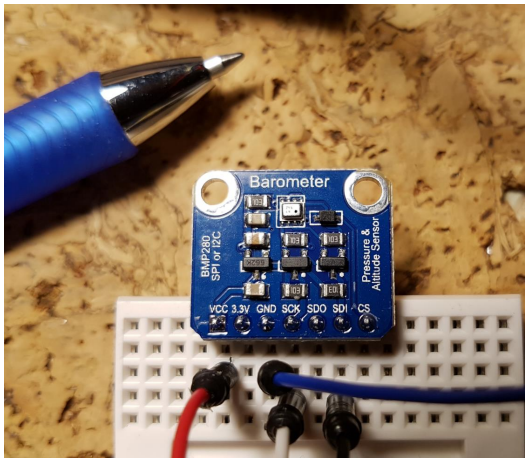
Comunicazione dati da Arduino al computer (tramite porta seriale)

Una delle tante *shield* di Arduino permette di salvare dati su microSD o altri tipi di memorie riscrivibili, ma può essere utile ricevere ed elaborare in tempo reale direttamente su computer i dati provenienti da Arduino.
E' possibile fare questo in molti modi, ad esempio tramite shield WiFi o Ethernet o Bluetooth o GSM (le possibilità sono infinite).
Un'opzione a costo zero, senza l'utilizzo di alcun shield aggiuntivo, consiste nell'utilizzare la porta monitor seriale di Arduino: è possibile scrivere e far girare su computer un software ad hoc, molto semplice, che legge i dati che Arduino invia sulla porta seriale (in forma di stringhe di testo), li converte in valori numerici e li utilizza per elaborazioni o archiviazione su computer in tempo reale. Ovviamente si può implementare anche la comunicazione da computer ad Arduino, in modo analogo.

http://www.lazarus-ide.org/

Si mostra ora un esempio effettuato utilizzando il compilatore Lazarus, ovvero free Pascal a oggetti, gratuito, open source (GPL/LGPL) e multipiattaforma (Windows, OSX, Linux).

ARDUINO WORKSHOP



Nell'esempio presentato si utilizza ancora il BMP280 e si manderà al computer attraverso la porta seriale il valore di temperatura, in modo continuo. Per facilitare l'interpretazione dei dati (che sono inviati in forma di stringa di testo e in modo non sincronizzato tra Arduino e il computer), questi saranno formattati nella forma [xx.yy] dove xx.yy è la temperatura, con due decimali.
Il programma su computer legge la seriale, estrapola la stringa formata dai 7 caratteri [xx.yy], converte xx.yy in forma xx,yy (in Italia il formato numerico prevede la virgola) e converte questa stringa in valore numerico, utilizzabile per elaborazione o archiviazione diretta sul disco rigido.
Lo sketch caricato su Arduino (MEGA) è una versione semplificata di quello visto in precedenza per il test del BMP280.

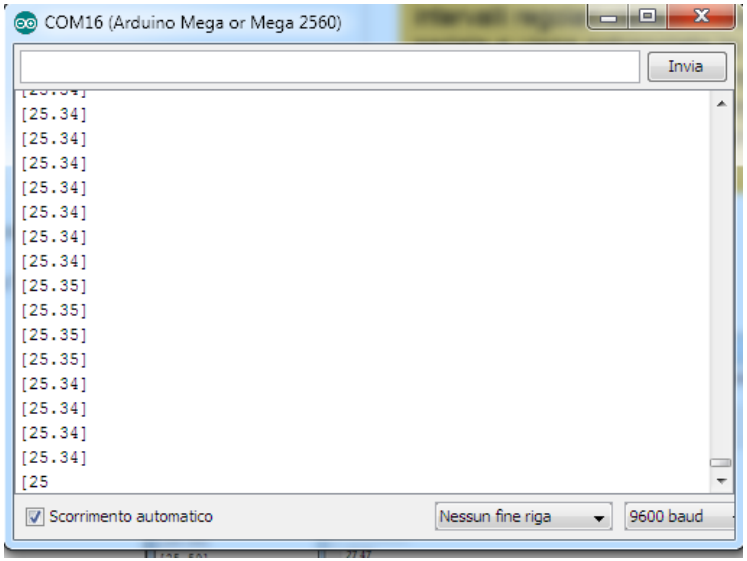
Il programma creato con Lazarus per leggere la porta seriale prevede l'installazione della libreria gratuita 5dpoSerial https://sourceforge.net/projects/sdpo-cl/files/ utile a gestire la comunicazione su porta seriale (virtuale).

I sorgenti di entrambi i software sono disponibili per il download.

ARDUINO WORKSHOP

Versione modificata dello sketch dell'esempio precedente. Su seriale viene scritta solo la temperatura, aggiungendo le parentesi quadre prima e dopo il valore numerico.

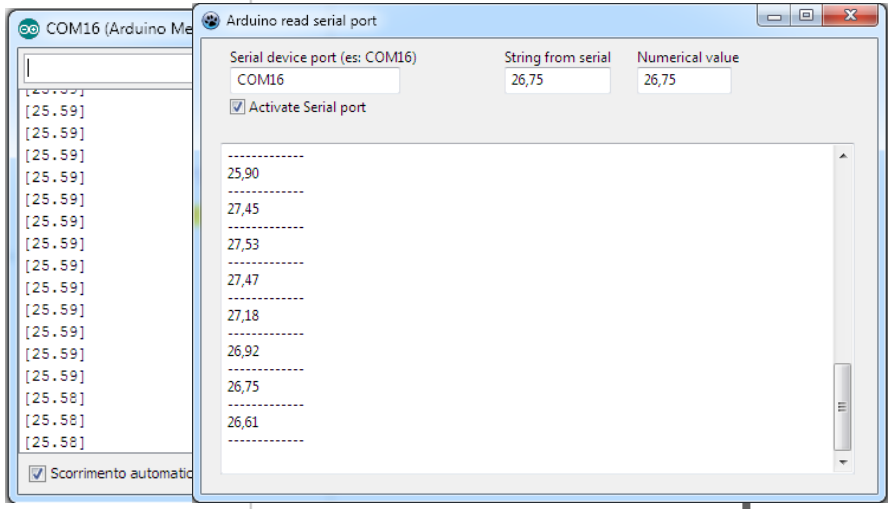
```
//Vin to 5V
//Gnd to Gnd
//SCK to SCL (21 on MEGA, A5 UNO)
//SDI to SDA (20 on MEGA, A4 UNO)

Adafruit_BMP280 bme; // I2C

void setup() {
  Serial.begin(9600);

  if (!bme.begin()) {
    Serial.println(F("Could not find a valid BMP280 sensor, check wiring!"));
    while (1);
  }
}

String SerialRow;

void loop() {

 SerialRow = String()+ "["+bme.readTemperature()+"]";
 Serial.println(SerialRow);
 delay(10);

}
```



ARDUINO WORKSHOP

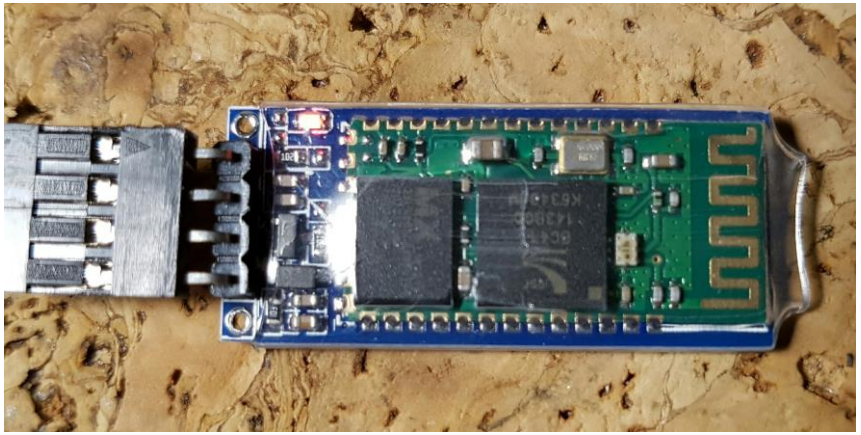

```
procedure TForm1.Timer1Timer(Sender: TObject);
var Data,First7chars,First5chars:String; NumericalValue:double; FlagError:boolean;
begin
  if not SdpoSerial1.Active then exit;

  Data:=SdpoSerial1.ReadData;

  if length(Data)>6 then
  begin
   First7chars:=copy(Data,1,7); // [24.57] expected

   if not ( (copy(First7chars,1,1)='[') or (copy(First7chars,7,1)=']') ) then exit;

   First5chars:=copy(First7chars,2,2)+','+copy(First7chars,5,2); // 24,57

   Memo1.Lines.Add(First5chars);
   Memo1.Lines.Add('-------------');
   Edit2.Text:=First5chars; // String
  end
   else exit;

  FlagError:=false;
  if length(First5chars)<>5 then exit;
  try NumericalValue:=StrToFLoat(First5chars);
    except On E:EConvertError do FlagError:=true; end;

  if not FlagError then Edit3.Text:=FloatToStr(NumericalValue)
                   else Edit3.Text:='N/A';

  sleep(150);
end;
```

Routine principale del programma
In Pascal scritto con Lazarus. A intervalli regolari viene letta la porta seriale e viene estrapolata la stringa contenente il valore della temperatura. Il punto viene convertito in virgola e la stringa è convertita in valore numerico.

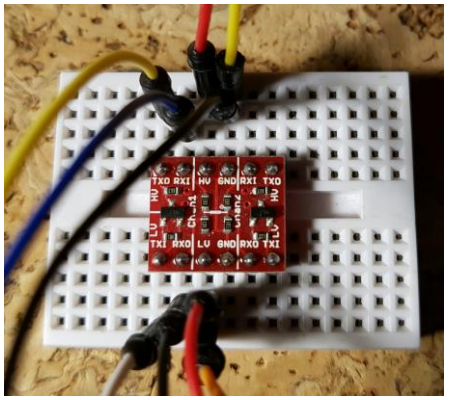


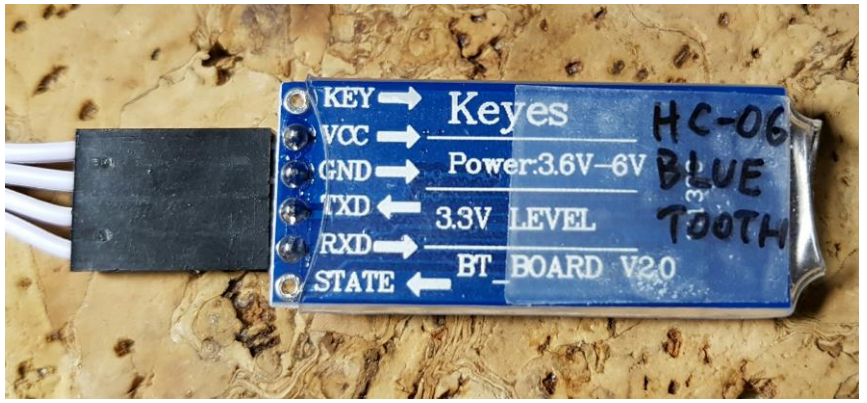
Comunicazione dati da smartphone ad Arduino tramite Bluetooth

Una shield chiamata HC-06 è utilizzata, connessa ad Arduino MEGA. Questa schedina contiene un transceiver Bluetooth e funziona a 3.3 V (anche se viene indicato 3.6-6V). Comunica con Arduino tramite una porta seriale, quindi un filo TX e un filo RX. Questi due segnali seguono lo standard CMOS a 3.3V quindi per non danneggiare la scheda occorre utilizzare uno shield *level shifter* che trasforma i segnali digitali da Arduino alla schedina da 5V a 3.3V e trasforma i segnali a 3.3V dalla schedina ai 5V richiesti da Arduino. Si usa la porta seriale 1 di Arduino Mega (ne ha 4), quindi i piedini 18 (TX1) e 19 (RX1). Il pin TXD della schedina HC-06 è quindi connesso al level shifter e quindi al pin RX1 di Arduino.



https://www.sunfounder.com/bluetooth-transceiver-module-hc-06-rs232-4-pin-serial.html





Il pin RXD della schedina HC-06 è invece connesso al level shifter e dopo la conversione al pin TX1 di Arduino

ARDUINO WORKSHOP

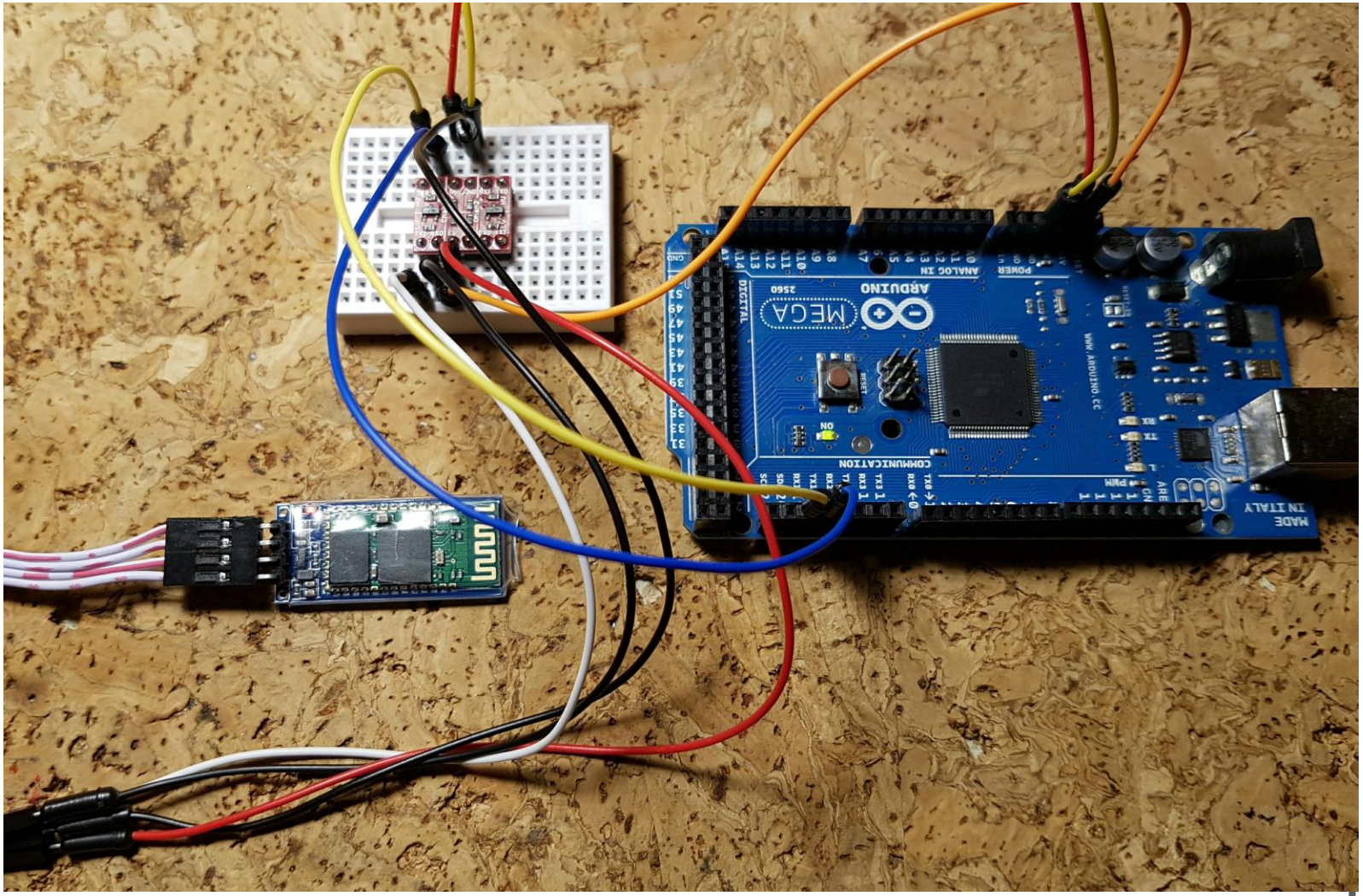
Lo sketch caricato su Arduino si occupa semplicemente di inizializzare le due porte seriali utilizzate: *Serial* è la porta seriale virtuale su computer che permette di visualizzare il monitor seriale e *Serial1* è invece una delle 4 porte seriali hardware di Arduino MEGA e in particolare la porta 1, che utilizza i pin 19 e 18. Il loop principale legge continuamente ciò che viene ricevuto dalla porta *Serial1* (HC-06) e lo ricopia sulla porta seriale virtuale per essere visualizzato sul computer. Le stringhe vengono mandate dal Bluetooth di uno smartphone mediante un'app gratuita chiamata Arduino Bluetooth Controller. In questo sketch il messaggio ricevuto via Bluetooth viene solo visualizzato ma lo stesso schema operativo può essere utilizzato per fare compiere ad Arduino delle azioni in remoto (es: irriga il prato), riconoscendo un determinato comando.

_13_HC06_Bluetooth

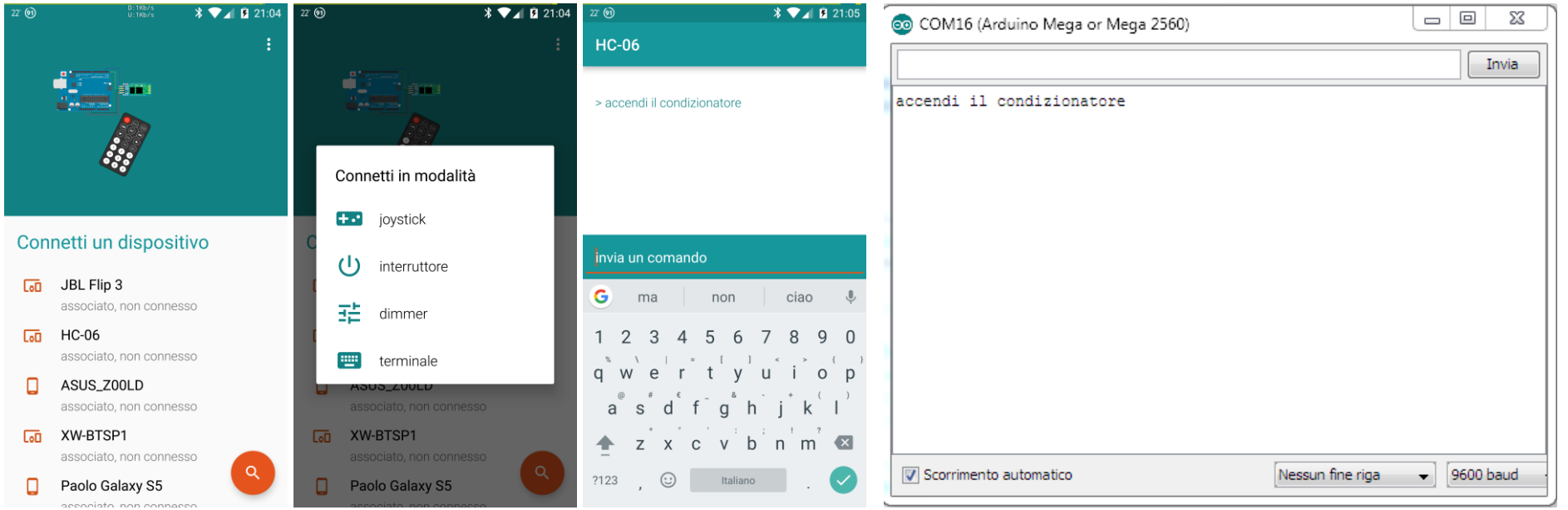
```
1  // Mega: Serial1, RX pin 19, TX pin 18
2  // su Android: "Arduino bluetooth controller", modalità Terminale
3
4  String message; //string that stores the incoming message
5
6  void setup()
7  {
8    Serial.begin(9600); //set baud rate (monitor su pc)
9    Serial1.begin(9600); //set baud rate (comunicazione con HC-06)
10 }
11
12 void loop()
13 {
14   while(Serial1.available())
15   {//while there is data available on the serial monitor
16     message+=char(Serial1.read());//store string from serial command
17   }
18   if(!Serial1.available())
19   {
20     if(message!="")
21     {//if data is available
22       Serial.println(message); //show the data
23       message=""; //clear the data
24     }
25   } else Serial.println("Serial 1 not available");
26   delay(1000); //delay
27 }
28
```

http://www.instructables.com/id/Add-bluetooth-to-your-Arduino-project-ArduinoHC-06/

ARDUINO WORKSHOP



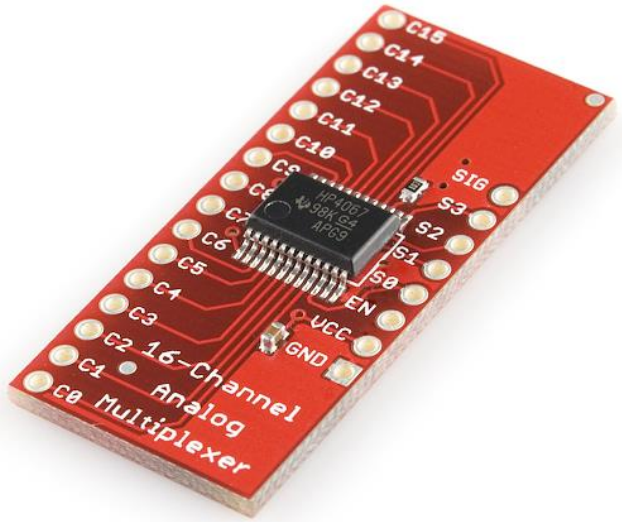
ARDUINO WORKSHOP



1. Il modulino HC-06 apparirà nella lista dei dispositivi Bluetooth
2. Con l'app Arduino Bluetooth Controller (Android) ci si connette a HC-06 in modo «Terminale»
3. Una volta connesso, il led lampeggiante dell'HC-06 resterà acceso fisso
4. Nell'IDE di Arduino sul computer aprire il monitor seriale
5. Dal terminale dell'app sul telefono si può digitare una frase e inviarla
6. Nel monitor seriale sul computer apparirà la frase ricevuta da Arduino

Utilizzo di un Multiplexer

Può essere necessario a volte l'utilizzo di un multiplexer, ovvero un dispositivo che funziona in modo simile a un selettore rotativo, quando ad esempio vogliamo connettere sequenzialmente un singolo ADC a diversi sensori analogici esterni. Lo shield usato è un semplice adattatore del circuito integrato CD74HC4067. Si può alimentare da 2V a 6V. Tramite i 4 ingressi digitali S0..S3 si può selezionare quale dei 16 pin C0..C15 viene collegato al pin SIG (in modo bidirezionale) tramite logica binaria: il numero in base 2 in ingresso a S0..S3 viene convertito nel numero Cx decimale. Il pin EN se collegato a un livello logico HIGH disabilita tutte le connessioni.



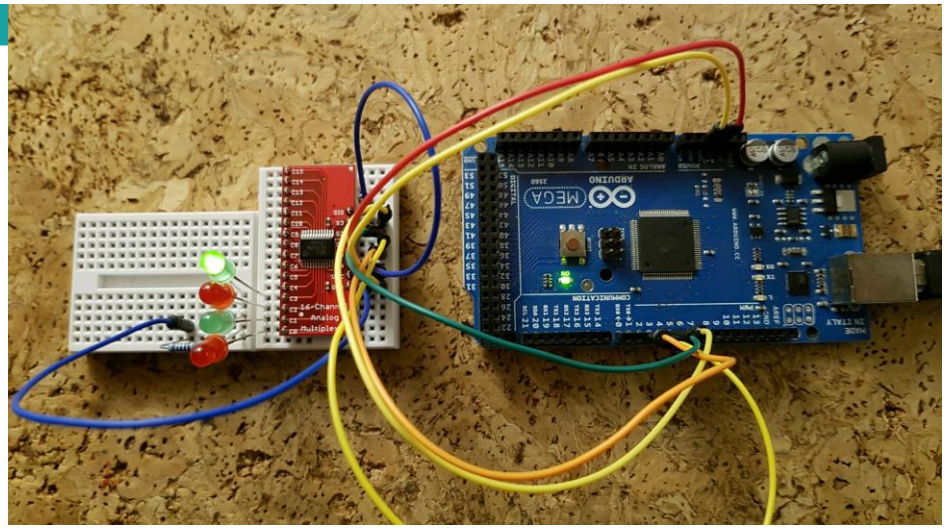
L'utilizzo è molto semplice: 4 uscite digitali di Arduino si collegano ai 4 pin di selezione S0..S3 e ragionando in logica binaria si seleziona il collegamento desiderato tra SIG e C0..15

https://www.sparkfun.com/products/9056

ARDUINO WORKSHOP

_14_Multiplexer §

```
// address multiplexer
int A_zero =  2; // pin S0 a pin 2 di MEGA
int A_one =   3; // pin S1 a pin 3 di MEGA
int A_two =   4; // pin S2 a pin 4 di MEGA
int A_three = 5; // pin S3 a pin 5 di MEGA

void setup() {
// 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
}

void loop() {
// Select address 0000  =0
  digitalWrite(A_zero, LOW); digitalWrite(A_one,  LOW);
  digitalWrite(A_two,  LOW); digitalWrite(A_three,LOW);
  delay(1000);

// Select address 0001  =1
  digitalWrite(A_zero, HIGH); digitalWrite(A_one,  LOW);
  digitalWrite(A_two,  LOW);  digitalWrite(A_three,LOW);
  delay(1000);

// Select address 0010  =2
  digitalWrite(A_zero, LOW); digitalWrite(A_one, HIGH);
  digitalWrite(A_two, LOW);  digitalWrite(A_three, LOW);
  delay(1000);

// Select address 0011  =3
  digitalWrite(A_zero, HIGH); digitalWrite(A_one, HIGH);
  digitalWrite(A_two, LOW);   digitalWrite(A_three, LOW);
  delay(1000);
}
```



In Arduino si utilizzano i pin 2, 3, 4 e 5 per comandare il selettore del multiplexer S0..S3. 4 LED sono collegati (+) alle prime 4 uscite del multiplexer. Gli altri piedini dei 4 LED sono collegati in comune a una resistenza da 1K, a sua volta collegata a massa. L'ingresso del multiplexer SIG è collegato a 5V.
Lo sketch abilita in sequenza le prime 4 uscite, tenendole accese per un secondo, quindi i 4 LED si accendono in sequenza. Si noti che nel numero binario di selezione dei pin, la cifra più a destra è S0, la penultima è S1, ecc..

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