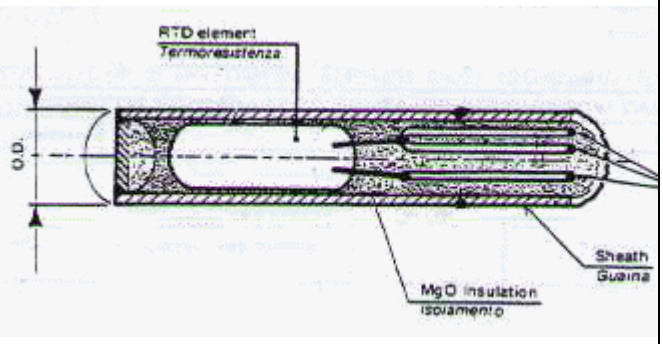
Trasduttori di temperatura

Termoresistenza **RTD** (dall'inglese **R**esistance **T**emperature **D**etector).

Il funzionamento si basa sul cambiamento della resistenza di un conduttore metallico in funzione della temperatura. Molto diffuse sono le cosiddette Pt100 e Pt1000, ovvero termoresistenze in platino (Pt), in cui la resistenza alla temperatura di 0 °C è pari rispettivamente a 100 Ω e 1000 Ω.
Il **TCR** (**T**emperature **C**oefficient of **R**esistance) di una termoresistenza indica la variazione media per grado centigrado del valore della resistenza fra gli 0 °C e i 100 °C. Può essere espresso con la formula: $$TCR = \frac{R_{100} - R_0}{R_0 \cdot 100}$$
La norma IEC 751 prescrive per le Pt100 un TCR di 0,00385Ω/Ω/°C.



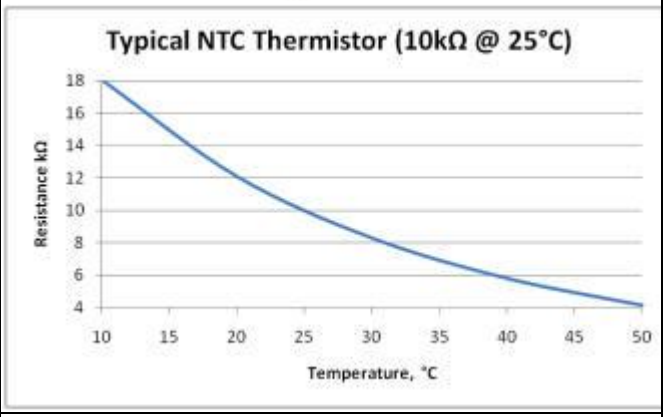
Termistori
Si basano sullo stesso principio delle termoresistenze, l'unica differenza tra i due sensori risiede nel materiale con cui sono realizzati:

- le termoresistenze sono composte da materiali conduttori metallici (per esempio platino)
- i termistori sono composti da materiali semiconduttori

I termistori si possono classificare in:

- NTC (Negative Temperature Coefficient) (resistenza che decresce con l'aumentare della temperatura);
- PTC (Positive Temperature Coefficient) (resistenza che cresce con l'aumentare della temperatura

$$R = R_0 \, e^{B(1/T - 1/T_0)}$$

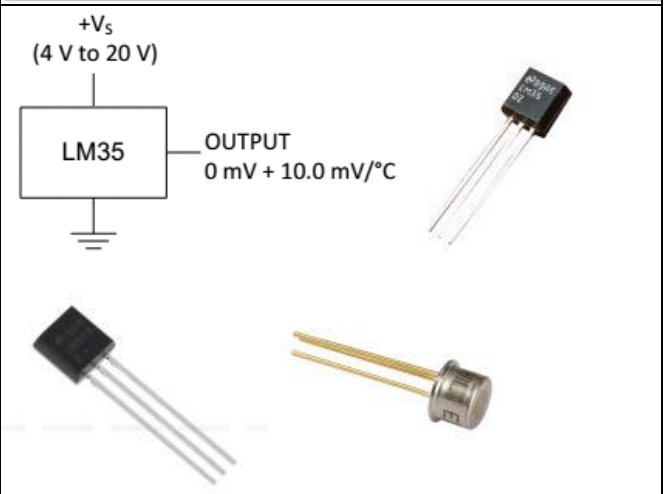


Sensori a circuito integrato analogici

LM35 (10mV/°C)

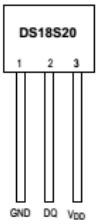
LM335 (10mV/°K)

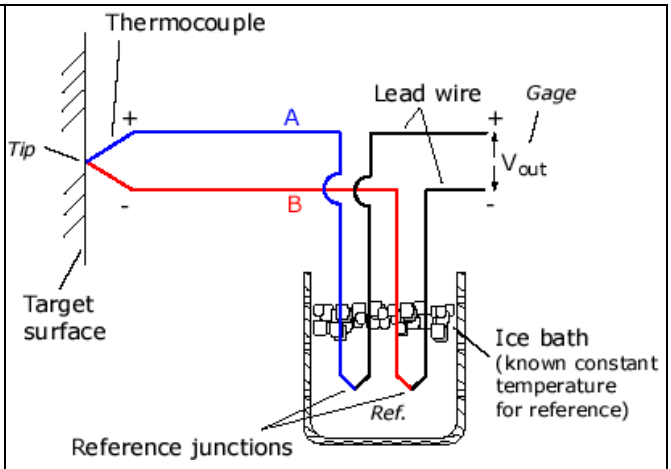
AD590 (10uA/°K)



Sensori a circuito integrato digitali

DS18S20



TERMOCOPPIE	
PIROMETRI	
TERMOCAMERE	

NTC

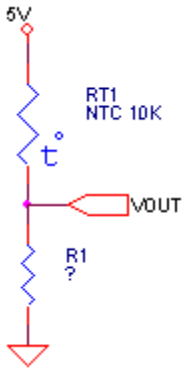
Registrare il cambiamento della resistenza di un NTC in funzione della temperatura.
Occorre:

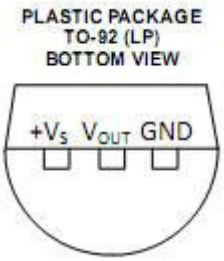
- Una sonda di temperatura di precisione per misurare la temperatura: usiamo un LM35
- Un riscaldatore su cui appoggeremo sia l'NTC che l' LM35: usiamo una resistenza da 33 Ohm da 5W con corpo ceramico.

La misura della resistenza dell'NTC verrà effettuata per via indiretta misurando la VOUT di un partitore composto dall'NTC e R1 (vedi figura), mentre la temperatura si rileva leggendo la Vout della sonda LM35 (10mV °C)

Progetto di R1

Dati:
Valim=5V
NTC=10K a 25°C
Range di temperatura da misurare: da **25°C** (tmin) a **80°C** (tmax)
R_NTC_tmin= 10000 ohm
R_NTC_tmax= 2250 ohm
VOUT_max=1V



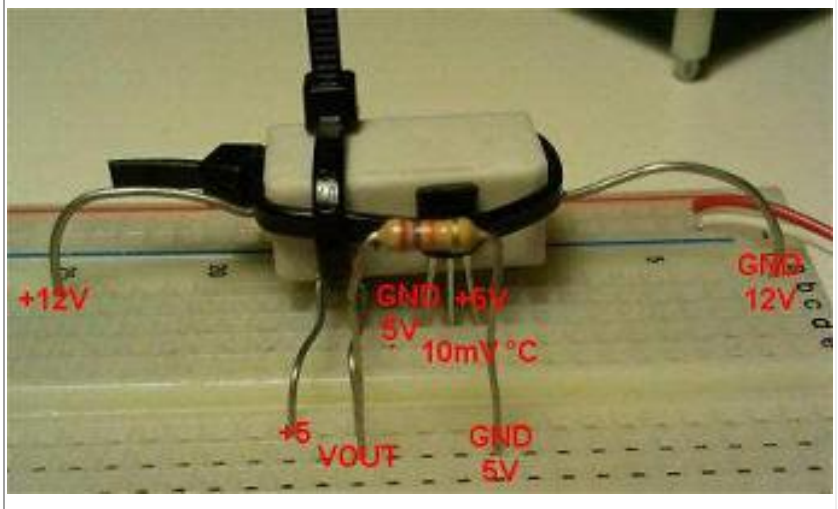


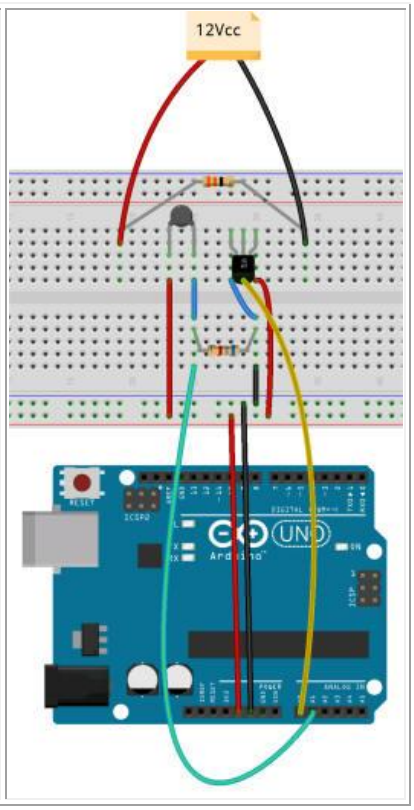
Datasheets
lm35.pdf
ntc_murata.pdf

Siccome la scheda Arduino è stata impostata da programma per leggere al massimo una tensione di 1.1V progettiamo R1 per ottenere la massima tensione su VOUT in corrispondenza di tmax.

R1=VOUT_max/Valim*(R1+R_NTC_tmax)
R1*(1-VOUT_max/Valim)=VOUT_max/Valim*R_NTC_tmax
R1=(VOUT_max/Valim*R_NTC_tmax)/(1-VOUT_max/Valim)=(1/5*2250)/(1-1/5)=562,5 (**Valore commerciale più vicino= 560 Ohm**)

Montaggio:





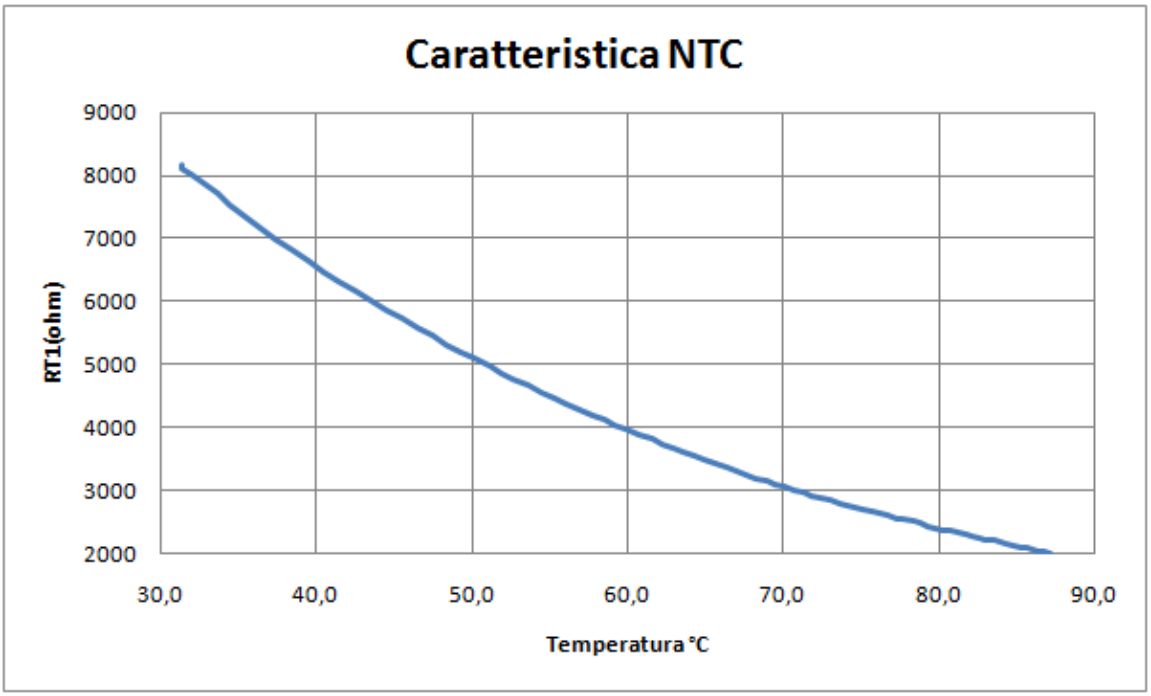
Procedimento

1. Montare il circuito come da figura.
2. Collegare il cavo USB da Arduino a PC
3. Lanciare il programma **ardu_acq115200,** scegliere la porta assegnata al collegamento USB
4. Predisporre le impostazioni per registrare le tensioni su A0 e A1, testare con spunta auto per vedere se vengono lette correttamente.
5. Impostare Acq. T a 2 secondi e N.Acqs a 100
6. Alimentare la resistenza riscaldatrice a 12V se subito cliccare su Registra.
7. Alla fine del ciclo di acquisizione togliere alimentazione alla resistenza riscaldatrice.
8. Salvare la tabella su file.
9. Caricare il file su foglio elettronico
10. Convertire i dati registrati nella grandezze corrispondenti: Temperatura=A0*1,1/1023/100, RT1=.....

N.B. per ottenere una lettura più stabile della temperatura sull' LM35 occorre un condensatore da 1uF in parallelo tra uscita e massa.

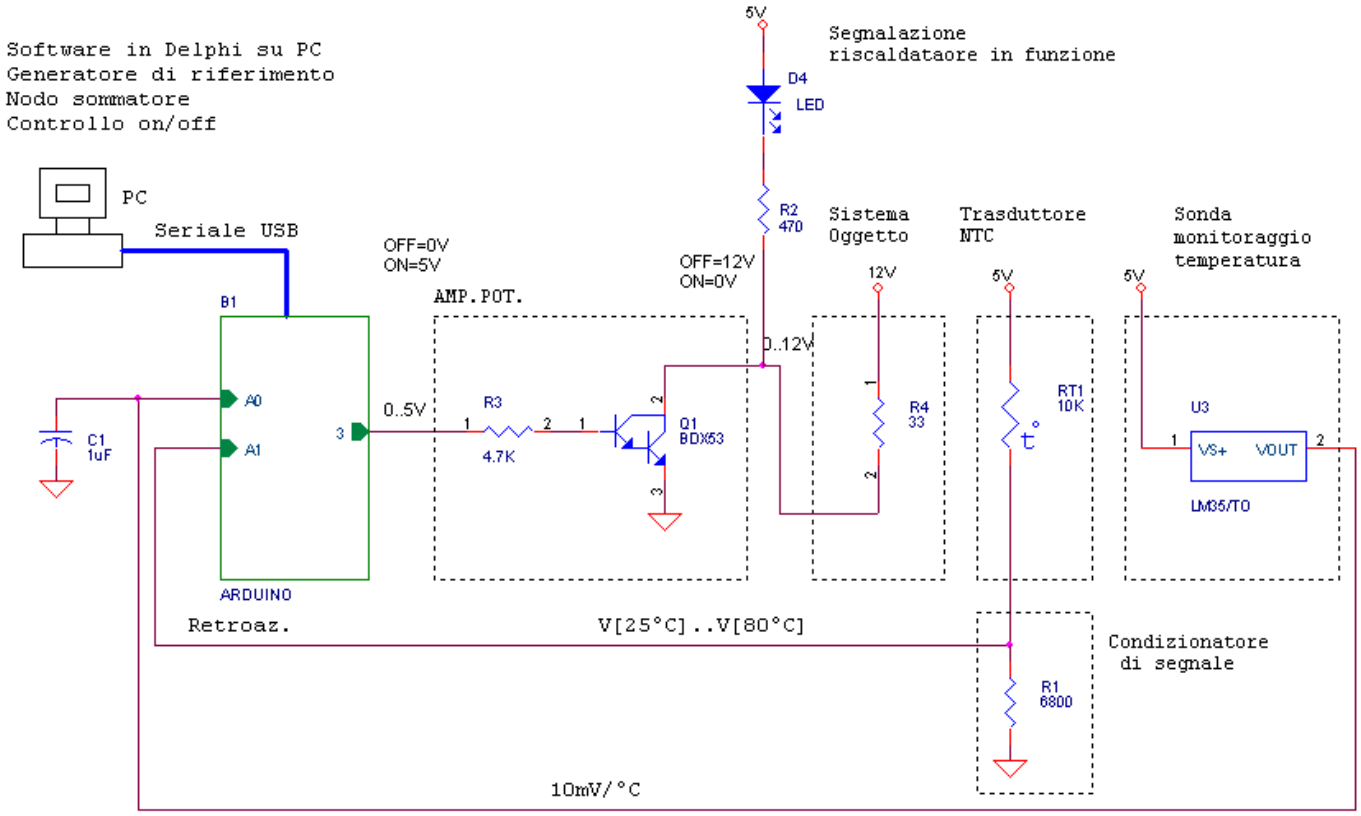
Risultati della prova

T(°C)	RT1(ohm)
31,4	8178
31,4	8120
32,0	8006
32,8	7867
33,7	7707
34,4	7527
35,4	7355
36,5	7167
37,4	6988
38,4	6817
39,5	6653
40,4	6478
41,5	6311
42,6	6169
43,5	6016
44,5	5870
45,5	5730
46,5	5582
47,5	5454
48,3	5318
49,4	5201
50,2	5089



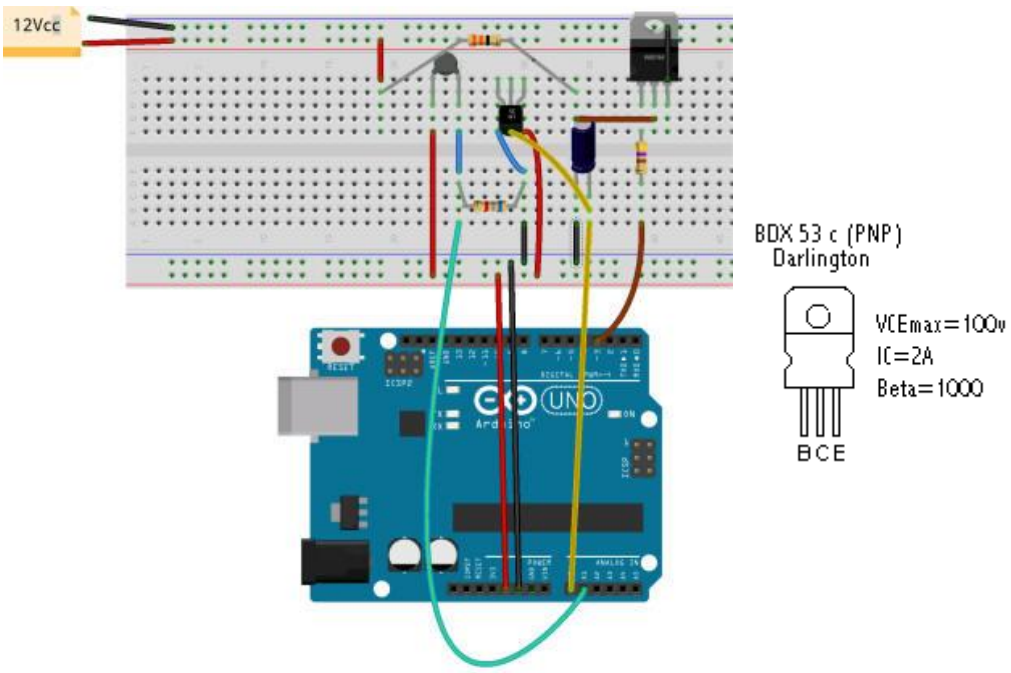
Controllo ON/OFF di temperatura software con Delphi e Arduino.

Schema elettrico

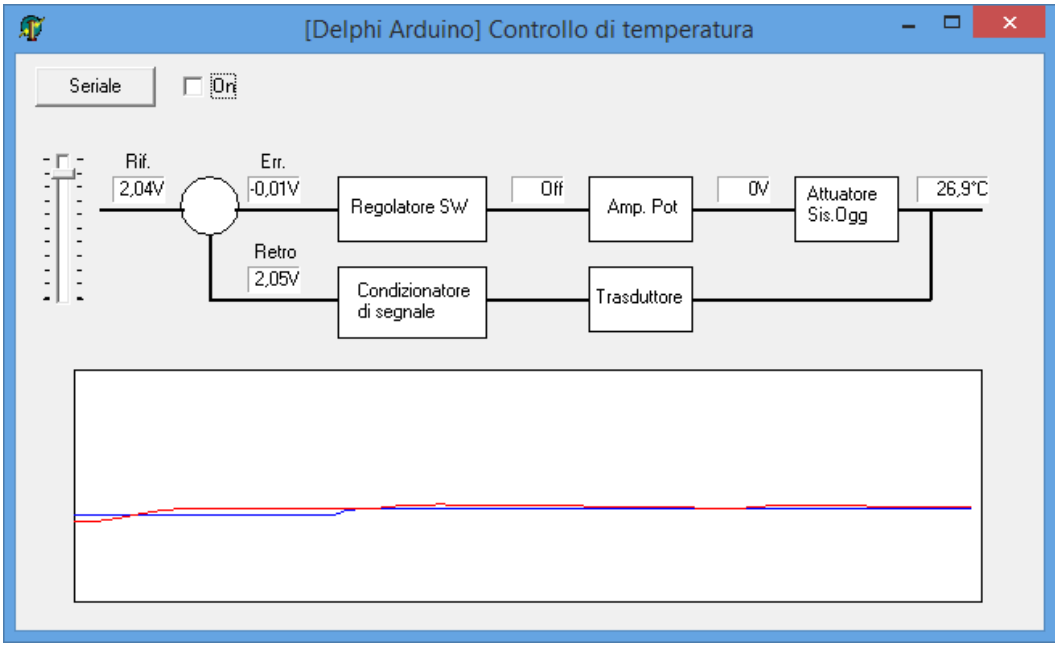


R_NTC[25°C]=11037
R_NTC[80°C]=3677
V[25°C]=5*R1/(R_NTC[25°C]+R1)=1.906V
V[80°C]=5*R1/(R_NTC[80°C]+R1)=3.245V

Schema di montaggio



Form



Sorgente software

unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ComCtrls, ExtCtrls;

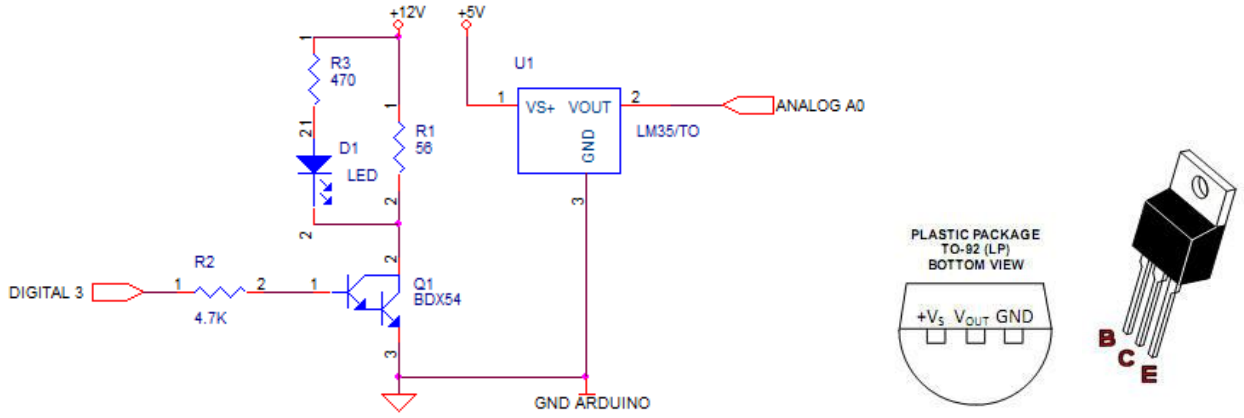
type
  TForm1 = **class**(TForm)
    Button1: TButton;
    Shape1: TShape;
    Label1: TLabel;
    Shape2: TShape;
    Shape3: TShape;
    Shape4: TShape;
    Label2: TLabel;
    Shape5: TShape;
    Shape6: TShape;
    Shape7: TShape;
    Shape8: TShape;
    Label3: TLabel;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TrackBar1: TTrackBar;
    StaticText4: TStaticText;
    Shape9: TShape;
    Shape10: TShape;
    StaticText5: TStaticText;
    Label4: TLabel;
    Shape11: TShape;
    Shape12: TShape;
    Label5: TLabel;
    Shape13: TShape;
    Shape14: TShape;
    Shape15: TShape;
    StaticText6: TStaticText;
    Timer1: TTimer;
    CheckBox1: TCheckBox;
    Shape16: TShape;
    Image1: TImage;
    Label6: TLabel;
    Label7: TLabel;

```
    Label8: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Grafica;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses UArduSerCom;

{$R *.DFM}
const
  KV=5/1023;
  KT=5/1023*100;
  RNTC_25=11037;
  RNTC_80=3677;
  R1=6800;
  MaxRef=5*(R1/(R1+RNTC_80));
  MinRef=5*(R1/(R1+RNTC_25));

var
  Rif:real;
  Retro:real;
  Err:real;
  Attuat:real;
  Uscita:real;

  ORif:real;
  ORetro:real;
  CurGr:integer;


procedure TForm1.Button1Click(Sender: TObject);
begin
  FormArduSerCom.Show;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  TrackBar1.Max:=round(MaxRef/KV);
  TrackBar1.Min:=round(MinRef/KV);
  TrackBar1.Frequency:=(TrackBar1.Max-TrackBar1.Min) div 10;

end;

procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  Rif:=TrackBar1.Position*KV;
  StaticText3.Caption:=Format('%5.2fV',[Rif]);
end;

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked then
  begin
    FormArduSerCom.SetMode(3,1);
    FormArduSerCom.WrDig(3,False);
    Timer1.Enabled:=True;
    CurGr:=0;
  end
```

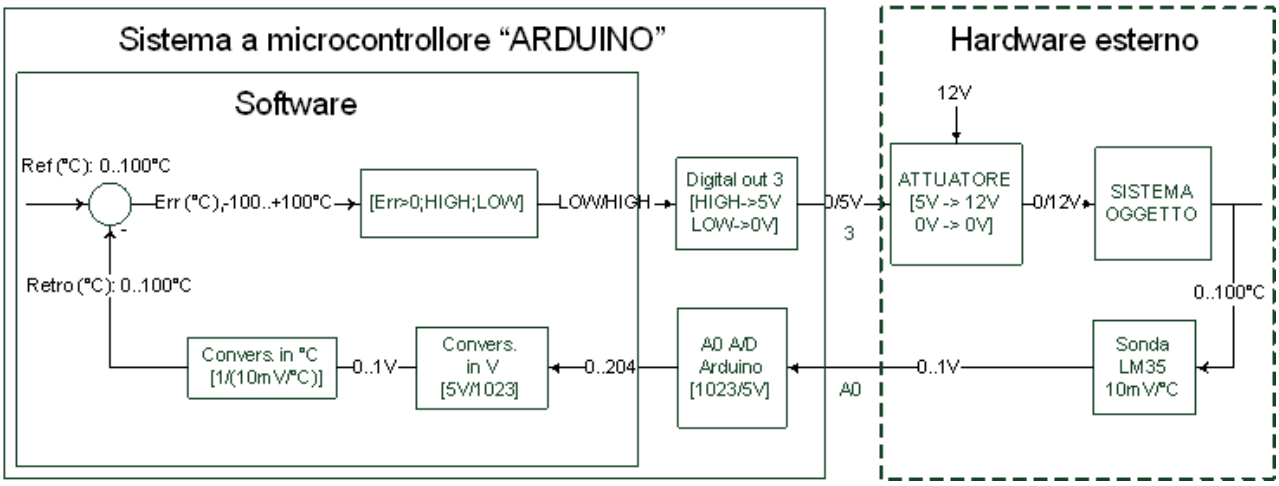
```
  else
  begin
    FormArduSerCom.WrDig(3,False);
    Timer1.Enabled:=False;
  end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Uscita:=FormArduSerCom.RdAna(0)*KT;
  StaticText6.Caption:=Format('%5.1f°C',[Uscita]);
  Retro:=FormArduSerCom.RdAna(1)*KV;
  StaticText2.Caption:=Format('%5.2fV',[Retro]);
  Err:=Rif-Retro;
  StaticText1.Caption:=Format('%5.2fV',[Err]);
  if Err>0 then
  begin
    StaticText4.Caption:='On';
    StaticText5.Caption:='24V';
    FormArduSerCom.WrDig(3,True);
  end
  else
  if Err<0 then
  begin
    StaticText4.Caption:='Off';
    StaticText5.Caption:='0V';
    FormArduSerCom.WrDig(3,False);
  end;
  Grafica;
end;

procedure TForm1.Grafica;
begin
  if CurGr>0 then
  begin
    Image1.Canvas.Pen.Color:=ClBlue;
    Image1.Canvas.MoveTo(CurGr-1,Image1.Height-round(ORif*Image1.Height/5));
    Image1.Canvas.LineTo(CurGr,Image1.Height-round(Rif*Image1.Height/5));
    Image1.Canvas.Pen.Color:=ClRed;
    Image1.Canvas.MoveTo(CurGr-1,Image1.Height-round(ORetro*Image1.Height/5));
    Image1.Canvas.LineTo(CurGr,Image1.Height-round(Retro*Image1.Height/5))
  end
  else
  begin
    Image1.Canvas.Brush.Color:=clWhite;
    Image1.Canvas.Pen.Color:=clBlack;
    Image1.Canvas.rectangle(0,0,Image1.Width,Image1.Height);
    CurGr:=0;
  end;
  inc(CurGr);
  if CurGr>Image1.Width then
  begin
    CurGr:=0;
  end;
  ORif:=Rif;
  ORetro:=Retro;
end;

end.
```

Controllo di temperatura con scheda Arduino

Schema elettrico



Schema a blocchi controllo

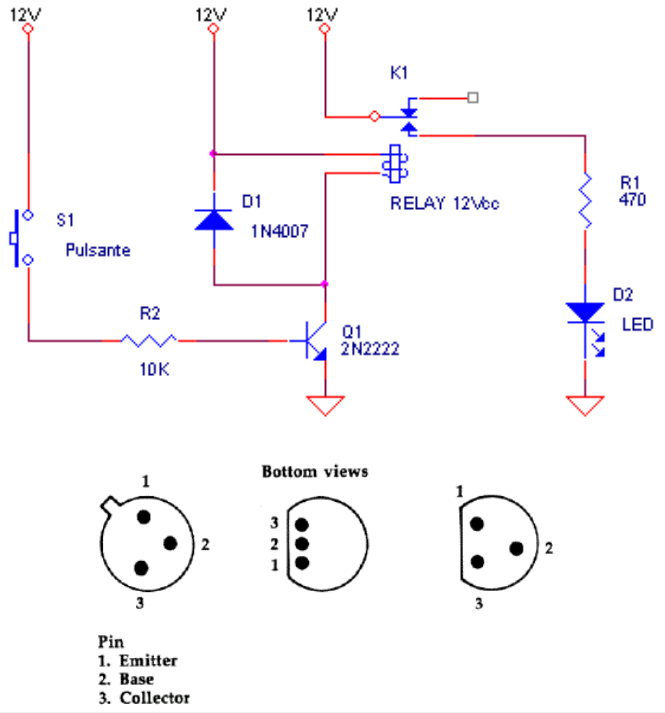
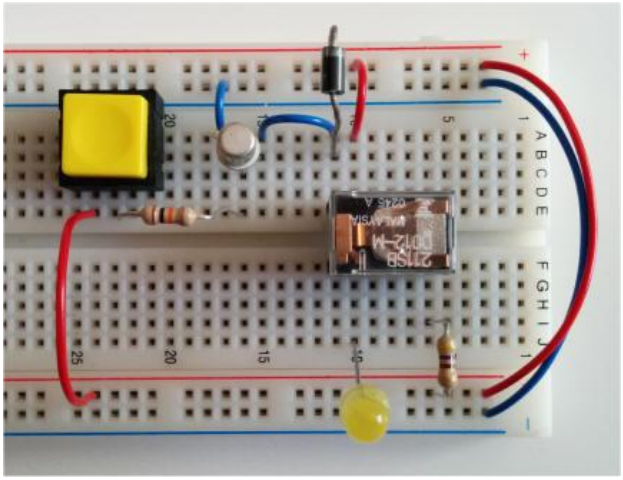


Software

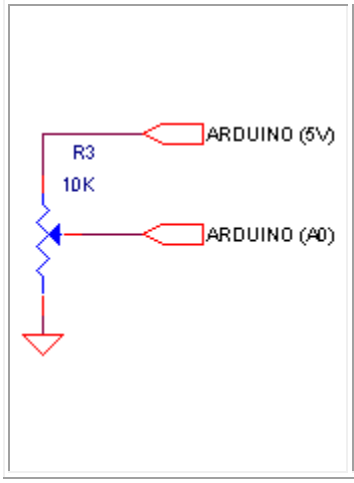
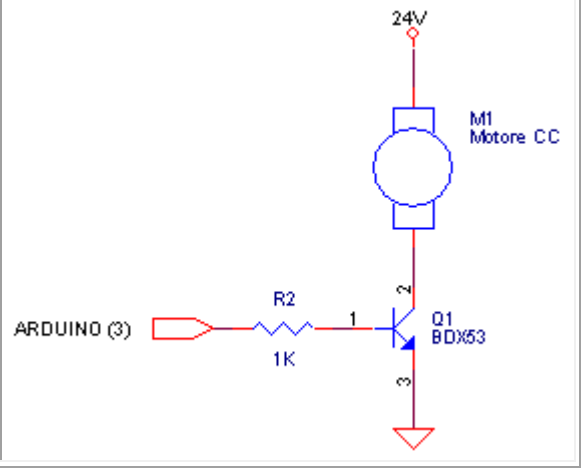
```
const float Rif=40.0;  //Riferimento di temperatura
const float KT=1/0.01; //K di conversione da tensione a temperatura
const float KV=5/1023; //K di conversione da valore ADC a tensione;
int   ValADC;         //Valore a ADC corrispondente a tensione applicata all'ingresso
int   V;              //Valore si tensione
float Retro;          //Temperatura misurata
float Err;            //Errore


void setup() {
  // put your setup code here, to run once:
  pinMode(3,OUTPUT);    //Inizializza pin 3 come uscita diglitale
  digitalWrite(3,LOW);  //Mette a 0 l'uscita (riscaldamento spento)
  Serial.begin(9600);   //Inizializza la porta seriale
}

void loop() {
  ValADC=analogRead(A0); //Legge ADC
  V=ValADC*KV;           //Trasforma in tensione
  Retro=V*KT;           //Trasforma in temeperatura
  Err=Rif-Retro;        //Elabora errore
  if (Err>0)            //Prende la decisione se accendere o spegnere il riscaldatore.
    digitalWrite(3,HIGH);
  else
    digitalWrite(3,LOW);

  //-----Monitoraggio sistema----------------------------
```

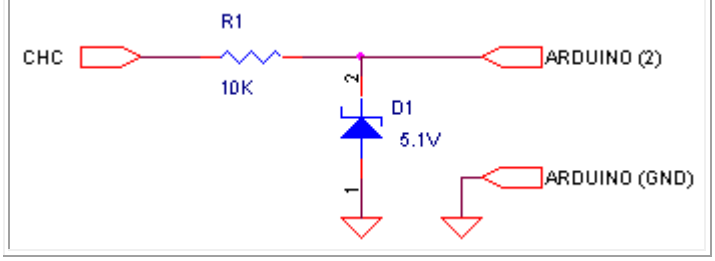
```
Serial.print("Rif:");
Serial.print(Rif);
Serial.print(" Retro:");
Serial.print(Retro);
Serial.print(" Err:");
Serial.print(Err);
Serial.print(" Out:");
Serial.println(digitalRead(3));
```

Pilotaggio relè con transistor in funzionamento ON/OFF

Schema elettrico	**Montaggio**
	
Parametri transistor 2N2222 Vce_sat.=0,4V Vbe =0,7V HFE =70 **Parametri Relè** RBobina = 320 Ohm **Parametri LED** VLed = 2.1 V ILed = 20 mA	**Progetto** Ic = (12-Vce_sat.)/RBobina=(12-0,4)/320=0,0356 A Ib = Ic/HFE = 0,0356/70 = 508,5uA R2 = (12-Vbe)/Ib = (12-0,7)/508,5e-6 = 22222,2 Ohm *Per assicurarsi la saturazione:* R2=R2/2= 22222,2/2 = 11111,1 Ohm *valore comm.* **10000** Ohm R1=(12-VLed)/ILed = (12-2,1)/20e-3 = 495 Ohm *valore comm.* **470** Ohm

Controllo di velocità PWM per motore cc con retroazione ad encoder con scheda Arduino

Riferimento	Attuatore
	

Retroazione


Lettura velocità tramite Encoder usando l'interrupt

```
// Tmin a   3500 RPM   =  60000/3500 =17mS
// Tmax a    500 RPM   =  60000/500 =120mS

#define  enc 2
#define  K 60000.0   // 60/10e-3
unsigned long ot=0;
unsigned long t;

void setup()
{
  pinMode(enc, INPUT);
  attachInterrupt(0, int_srv, RISING);
  Serial.begin(9600);
}

void loop()
{
  Serial.print(t);
  Serial.print(" mS  ");
  Serial.print(K/t);
  Serial.print(" RPM");
}

void int_srv()
{
  unsigned long tmp;
```

```
  tmp=millis();
  t=tmp-ot;
  ot=tmp;
}
```

Realizzare il software per controllo della velocità dell'albero motore:
Arduino(A0) a 0V, RPM=0
Arduino(A0) a 5V, RPM=3300

```
#define  enc 2
#define  K 60000000.0   // costante per l'elaborazione della velocità RPM
#define  A 1/8          // costante che determina la velocità di reazione (un aumento di A corrisponde ad un aumento della velocità di reazione)
unsigned long ot=0;
unsigned long t;
int u;

void setup()
{
  pinMode(enc, INPUT);
  attachInterrupt(0, int_srv, RISING);
  Serial.begin(9600);
}

void loop()
{
  long int ref,retro,err;

  ref=analogRead(A0);   // ref  =0..1023
  retro=K/t/3.3;        // K/t=0..3300 RPM, retro=0..1000
  err=(ref-retro);      // elaborazione errore
  u+=err*A;             // elaborazione uscita
  if (u>255)            // limitazione uscita
    u=255;
  else
  if (u<0)u=0;
  analogWrite(3,u);         // output uscita
  if (micros()>(ot+300000)) // forzatura di t in caso di albero fermo
    t=300000;
  delay(200);               // ritardo per ottenere la misura di velocità fedele

  Serial.print("v:");
  Serial.print(K/t);
  Serial.print(" RPM ");
  Serial.print(" rif:");
  Serial.print(ref);
  Serial.print(" retro:");
  Serial.print(retro);
  Serial.print(" err:");
  Serial.print(err);
  Serial.print(" out:");
  Serial.println(u);

}

void int_srv()
{
  unsigned long tmp;
  tmp=micros();
  t=tmp-ot;           // t=tempo in uS tra due impulsi su canale C
  ot=tmp;
}
```