Coppia di Termometri con LCD in comunicazione I2C

di Naclerio Pasquale

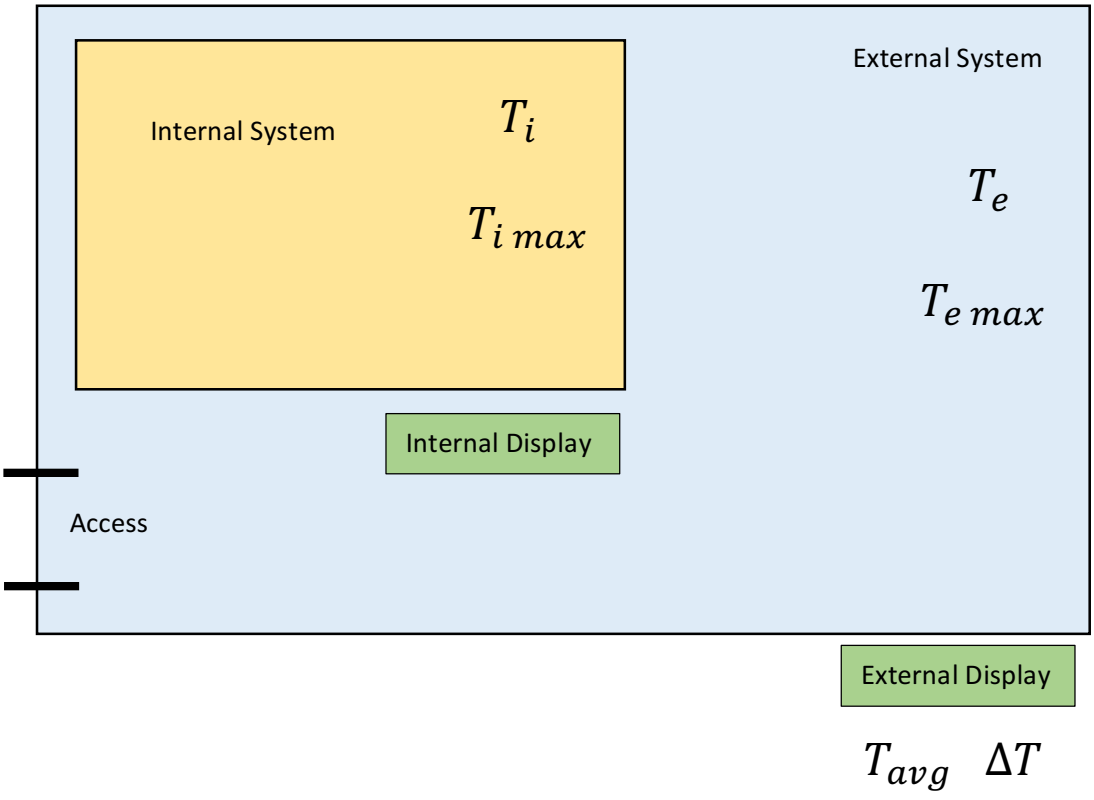
Laurea Magistrale in Ingegneria Elettronica e Informatica curriculum Elettronica

Introduzione:

Il progetto che segue va a realizzare il monitoraggio della temperatura in un sistema complessivo "external system" al cui interno vi è collocato un altro sistema "internal system". L'operatore dall'esterno può monitorare la temperatura media del sistema e la differenza di temperatura tra i due sistema. Nel momento in cui la temperatura interna o esterna dovessero superare una certa soglia, l'operatore dall'esterno può vedere che sistema è andato in surriscaldamento (overheated). Nel caso il sistema ad essere surriscaldato sia quello interno, può decidere di accedervi e vedere direttamente la temperatura del sistema da un suo display apposito. Un apposito segnale acustico accompagnato da un segnale di pericolo sul display segnalano invece che l'intero sistema ha raggiunto una temperatura di surriscaldamento.

Specifiche del progetto

Il sistema immaginato è il seguente:



Vi sono due sistemi con differenti temperature; il sistema interno ha una temperatura T_i e una temperatura massima $T_{i\,max}$ a cui l'intero sistema deve sottostare per non andare in surriscaldamento. All'esterno di questo sistema ho un altro sistema con una temperatura T_e e una sua temperatura massima $T_{e\,max}$ a cui l'intero sistema deve sottostare per non andare in surriscaldamento. Naturalmente la temperatura del sistema esterno è influenzata da quella interna, questa influenza viene tenuta da conto dai vincoli massimi di temperatura esterna. L'operatore potrà vedere dall'esterno di tutti i sistemi la temperatura media del

sistema e che differenza di temperatura c'è tra i due sistemi. Si è immaginato un sistema chiuso dove l'accesso al sistema esterno per verificare la temperatura del sistema interno è un operazione eseguita solo per particolari manutenzioni. Per questo fatto, dal Display esterno sarà possibile non solo visualizzare i dati di temperatura media e di differenza di temperatura, ma si potranno vedere i vari segnali di allarme, differenti per ogni possibile situazione di surriscaldamento.

I vari casi sono:

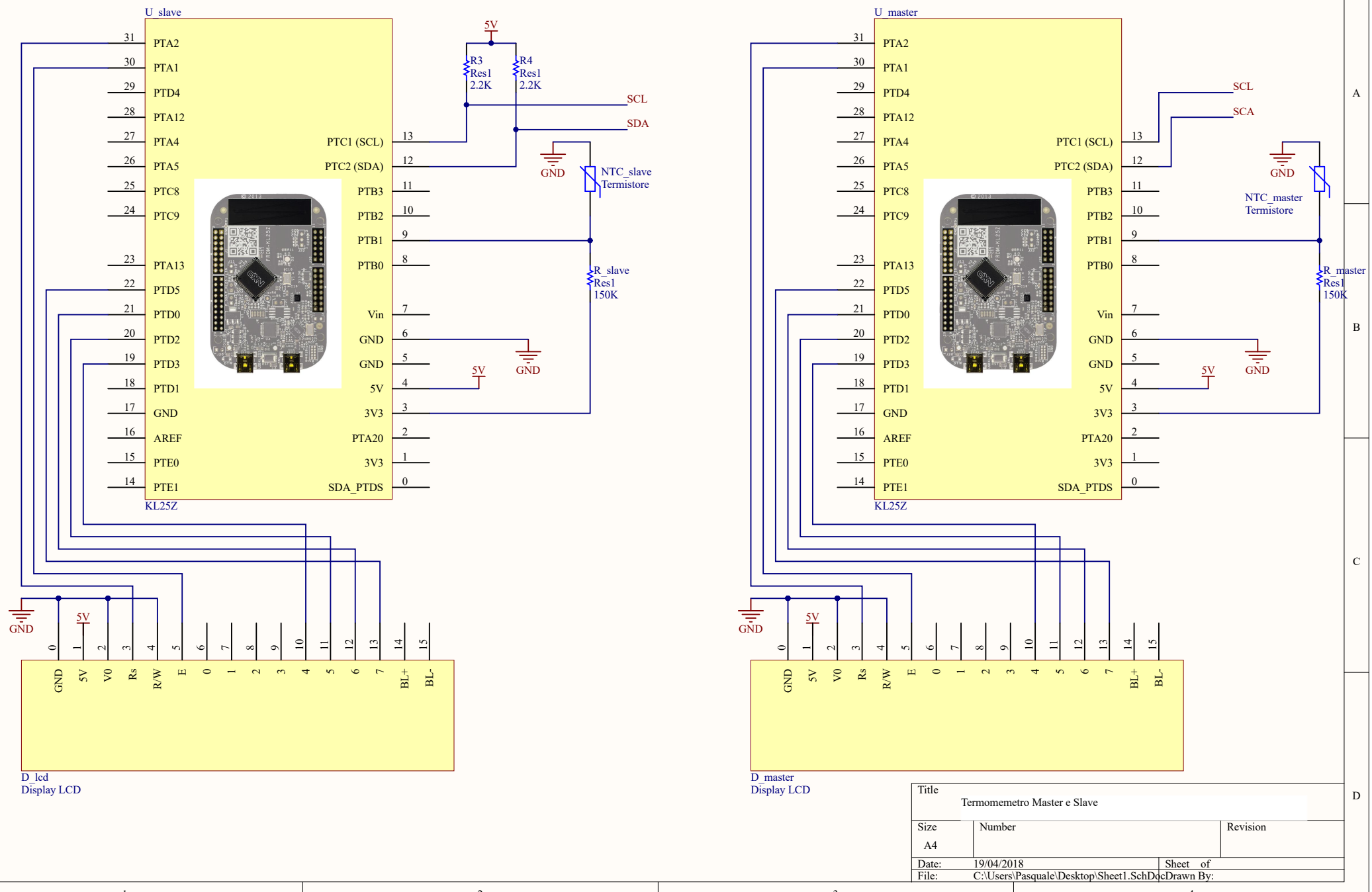
- $T_i < T_{i\,max}\ e\ T_e < T_{e\,max}$ il sistema è nella situazione di normalità e quindi ho un led verde su entrambi i sistemi e un i display mostreranno per il sistema interno la sua temperatura, mentre per il sistema esterno mostrerà la temperatura media e il la differenza di temperatura.
- $T_i \geq T_{i\,max}\ e\ T_e \geq T_{e\,max}$ il sistema è surriscaldamento e quindi i led su tutto il sistema saranno rossi, si avverte un segnale acustico e sui display viene visualizzato un messaggio di pericolo "DANGER!!! system overheated"
- $T_i \geq T_{i\,max}$ mentre $T_e < T_{e\,max}$ solo il display esterno mi darà un messaggio di pericolo per il sistema esterno in surriscaldamento "internal system overheated".
- $T_e \geq T_{e\,max}$ mentre $T_i < T_{i\,max}$ solo il display esterno mi darà un messaggio di pericolo per il sistema esterno in surriscaldamento "external system overheated".

Realizzazione Hardware

Il sistema usa i seguenti componenti:

- n.2 schede Freescale FRDM-KL25Z.
- n.2 schermi LCD 162B-D REV.A
- n.2 termistori NTC B57164K da $150k\Omega$
- n.2 resistenze da $150k\Omega$
- n.2 resistenze da $2.2k\Omega$
- n.1 buzzer 3.5-5-5V passivo.

Lo schema seguente mostra i collegamenti dei vari componenti, le due schede si interfacciano attraverso il protocollo I2C in cui una scheda fa da Master e una da Slave. Lo Slave è la scheda che farà le misure per il sistema interno mentre il Master è la scheda che farà le misure per il sistema esterno e si occuperà di mostrare i vari allarmi all'operatore.


U_slave
PTA2
PTA1
PTD4
PTA12
PTA4
PTA5
PTC8
PTC9
PTA13
PTD5
PTD0
PTD2
PTD3
PTD1
GND
AREF
PTE0
PTE1
PTC1 (SCL)
PTC2 (SDA)
PTB3
PTB2
PTB1
PTB0
Vin
GND
GND
5V
3V3
PTA20
3V3
SDA_PTDS
KL25Z
5V
R3
Res1
2.2K
R4
Res1
2.2K
SCL
SDA
GND
NTC_slave
Termistore
R_slave
Res1
150K
GND
5V
GND
D_lcd
Display LCD
GND
5V
V0
Rs
R/W
E
BL+
BL-
U_master
SCL
SCA
GND
NTC_master
Termistore
R_master
Res1
150K
5V
GND
D_master
Display LCD
Title
Termomemetro Master e Slave
Size
A4
Number
Revision
Date:
19/04/2018
Sheet of
File:
C:\Users\Pasquale\Desktop\Sheet1.SchDoc
Drawn By:

Misura di temperatura

La misura della temperatura viene eseguita in entrambe le schede nel medesimo modo. Si esegue un partitore di tensione tra una resistenza da $R = 150\Omega$ e l'NTC. L'NTC a $T_0 = 298{,}6\ K\ (25°C)$ ha una resistenza di $R_T = 150k\Omega$, siccome alimento il sistema a $V_{cc} = 3{,}3\ V$ allora per la formula del partitore di tensione cadono su R_T esattamente $V_T = 1{,}65V$, cioè la metà. Al contrario, se la temperatura è diversa la R_T sarà differente dal suo valore standard e quindi misurando la tensione di caduta sulla resistenza posso ricavarmi il valore della tensione a quella determinata temperatura. Successivamente da questo valore di resistenza, usando una formula presente sul Datasheet posso ricavare il valore di temperatura. Il mio processore quindi dovrà campionare questa misura di tensione e attraverso le varie formule ricavare la misura di temperatura.

$$R_T = \frac{V_T}{V_{cc} - V_T} R$$

$$T = \frac{B}{\ln \frac{R_T}{Re^{\left(-B/T_o\right)}}}$$

Dove B è una costante che vale $B = 4600$.

Il codice C che si occupa di questo è il seguente:

```
#include "mbed.h"

AnalogIn temperature(PTB1);

double Rt; double Vout; double Vcc = 3.3; double R = 150000; double B = 4600; double T0 = 298.15;  double array_T[5]; double T; int i;

int main() {

  while(1) {

      for(i=0; i<5; i++){

      Vout = temperature.read()*3.3;

      Rt = (Vout/(Vcc - Vout))*R;

      array_T[i] = (B / (log(Rt/(R*exp(-B/T0)))))-273.15;

      }

      T = (array_T[0]+array_T[1]+array_T[2]+array_T[3]+array_T[4])/5;

  }

}
```

Display LCD

Per visualizzare i dati su LCD invece, una volta eseguiti gli opportuni collegamenti (come da schematico), si passa ad una semplice stesura del codice che permette la visualizzazione. Il display è a 16 segmenti per 2 righe quindi devo gestire bene lo spazio a disposizione per visualizzare il numero minimo di informazioni che mi consentano di essere esauriente e chiaro nella rappresentazione.

Ho incluso al progetto la libreria TextLCD.h e con questa ho usato l'opportuno metodo:

```
TextLCD lcd(rs, e, d4, d5, d6, d7, TextLCD::LCD16x2B);
```

Per fare la stampa su LCD si usa

```
lcd.printf("Temp %-2.2f C", T);
```

La quale stamperà il valore della temperatura con due cifre significative (es. T = 26,56780 stampa Temp 25,57 C).

Per poter collocare la scritta in vari punti dello schermo uso il metodo lcd.locate(n colonna, m riga) con questo posso scrivere sulla seconda riga.

Siccome però voglio visualizzare un messaggio di errore superiore ai 16 caratteri allora visualizzo i vari messaggi in scorrimento, da destra verso sinistra.

Per fare questo invece di allocare dinamicamente il messaggio, faccio stampare in un ciclo un messaggio contenuto in un array di char così da visualizzare tutto il messaggio e dare l'illusione dello scorrimento.

```
char out[] =  "            DANGER!!! system overheated                  ";


for(int i=0; i < 50; i++){
            wait(0.2);
            lcd.cls();
            lcd.printf("%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c" ,
out[i],out[i+1],out[i+2],out[i+3],out[i+4],out[i+5],out[i+6],out[i+7],out[i+8],out[i+9],out[i+10],out[i+11],out[i+12],out[i+13],out[i
+14],out[i+15],out[i+16]);
                        }
            }
```

Con questo metodo è stato possibile stampare tutti i messaggi per i vari casi.

Quindi in definitiva il display stamperà per i seguenti messaggi, nei seguenti casi:

Casi:	Stampa su LCD Master
sempre	T system = "media T slave e T master " Diff i – e = "differenza tra T interna ed esterna"
se T master ≥ T master max e T slave ≥ T slave max	DANGER!!! system overheated
se T master < T master max e T slave ≥ T slave max	internal system overheated T = "T slave"
se T master ≥ T master max e T slave < T slave max	external system overheated T = "T master"

Casi:	Stampa su LCD Slave
sempre	T = “T slave”
se T slave ≥ T slave max	DANGER! T =”T slave”

Comunicazione in I2C

La comunicazione in I2C tra il master e lo slave è eseguita su un bus di due linee bidirezionali scl, per la tempistica di sincronizzazione, e sda, per il trasporto dati. I dati trasportati sono singoli byte, e da protocollo occorre mettere entrambe le linee in pull-up con due resistenza $R_{pull-up} = 2{,}2k\Omega$. Ogni dispositivo ha un indirizzo di 7 o 10 bit e la modalità di trasmissione scelta tra master e slave è:

1. Master spedisce indirizzo dello Slave sul Bus
2. Lo Slave spedisce i dati al Master
3. Il Master termina la trasmissione

A livello di codice mi vengono in aiuto i metodi

Per il Master:

```
I2C i2c(pin_sda, pin_sdl);
```

Per lo Slave:

```
I2CSlave slave(pin_sda, pin_scl);
```

Che si occupano di tutto il protocollo, occorrerà quindi definire in modo opportuno i parametri che occorrono. Lo slave manderà al master una serie di byte che sono la lettura della sua temperatura. Ho previsto 7 byte char per fare questo, però per il contenuto informatico di temperatura bastano solo 5 byte char; 4 byte di numeri e uno di virgola. Userò quindi i restanti byte per mandare il valore della temperatura massima dello slave. In questo modo slego il master dalla conoscenza di questo dato tipico dello slave, così facendo, nel caso il mio sistema preveda più slave, io posso ricevere questo valore dai vari slave; inoltre se volessi modificare quel dato mi basterebbe cambiarlo sullo slave senza toccare il master. Quindi l’array di char che lo slave manderà sarà fatto così: per una temperatura di 23,45 °C e una temperatura massima di 67°C avrò una stringa di char “23,4567”.

Il lato slave farà le seguenti operazioni:

1. va a definirsi il suo indirizzo 0x90 “slave.address(0x90)”
2. calcola la temperatura
3. converte la temperatura da double in char e la mette nel vettore da inviare
4. assegna ai due byte finali il valore della temperatura massima dello slave
5. aspetta di ricevere la richiesta di invio dati “ slave.receive()”
6. lo slave invia “slave.write(messaggio, strlen(temp)+1)”

Il lato Master farà le seguenti operazioni:

1. Invia la richiesta di lettura allo slave “i2c.read(addr, cmd, 7, true)”
2. Riceve e dal valore ricevuto ricava la temperatura massima dello slave e la temperatura dello slave
3. Fa il suo calcolo della temperatura
4. Esegue il codice per la scrittura sul display

Durante la progettazione del sistema, mi sono accorto che i sistemi avevano problemi di sincronizzazione, mentre il master inviava sempre la richiesta allo slave, lo slave se si trovava ad eseguire il codice di scrittura sul master rispondeva, altrimenti non inviava il dato. Per ovviare a questo problema quindi ho scelto di lasciare lo slave in attesa del master, per farlo lo slave eseguirà un ciclo con "slave.recive()" e uscirà solo dopo che avrà ricevuto la richiesta del master. Questo però comporta un rallentamento del sistema dello slave che non eseguirà una nuova lettura della temperatura fino a che non riceve la richiesta del master. In questo caso però la richiesta arriva sempre quindi si risolve il problema della sincronizzazione, ovviamente tutto è concepito per un sistema che non deve avere rapide variazioni di temperatura e quei pochi millisecondi persi non compromettono il sistema. In alternativa si sarebbe dovuto ripensare al protocollo di comunicazione, perché se avessi più slave allora dovrebbero essere loro ad inviare sempre al master e il master non farebbe altro che aspettare i vari slave. In questo caso sarebbe lui in attesa.

Conclusione

In conclusione sono state aggiunte delle piccole interfacce led per segnalare in maniera visibile il guasto e un piccolo buzzer sul master che si allerterà non appena l'intero sistema è andato in surriscaldamento. Il codice è concepito in modo tale che da essere modificato semplicemente nei suoi valori e quindi personalizzato a seconda delle esigenze.

Le problematiche riscontrate durante la mia esperienza di laboratorio sono state varie, sia di natura hardware e software.

Per quelle hardware:

- Taratura del sistema di misura per la misura di temperatura, usando una resistenza campione di valore analogo a quella del sensore a 25°C così da vedere la lettura del dimezzamento della tensione misurata.
- Connessione di tutti i componenti attraverso un uso razionale della breadboard.
- Connessione dell'LCD sulla scheda.
- Realizzazione del bus per I2C usando le resistenze pull-up.

Per quelle software:

- Reperire il materiale necessario per programmare la scheda dal sito mbed.
- Scrivere un programma di C.
- Permettere alla scheda di leggere e scrivere sui suoi pin.
- Gestire la connessione I2C con le sue tempistiche e dover compiere delle scelte su chi opera da master e da slave in modo coerente senza perdere la sincronizzazione.
- Gestire LCD in modo da poter risolvere il problema della scrittura dei vari messaggi.

Il sistema da me realizzato è in grado quindi di misurare due temperature diverse e di comunicarle al master in modo da diretto, gestisce varie situazioni critiche e permette ad un operatore di sapere con esattezza quale sistema ha dei problemi di surriscaldamento.

Codice Slave

```
#include "mbed.h"

#include "string.h"

#include "TextLCD.h"


I2CSlave slave(PTC2, PTC1);

AnalogIn temperature(PTB1);

TextLCD lcd(PTA1, PTA2, PTD3, PTD2, PTD0, PTD5, TextLCD::LCD16x2B); PwmOut r(LED_RED);

PwmOut g(LED_GREEN);


const int Tmax = 30;


double Rt;

double Vout; //lettura della tensione

double Vcc =3.3;

double R= 150000;

double B= 4600;

double T0= 298.15;  //temp. in kelvin vale 25°c

double array_T[5];

double T;

int i;

int j;

char out[] = "             Sensor in short circuit!!!                    ";


char temp[8];

char temp_max[2];


int main() {

  slave.address(0x90);

  while (1) {


      for(j = 0; j < 5; j++){

      Vout = temperature.read()*3.3;

       Rt = (Vout/(Vcc - Vout))*R;

       array_T[j] = (B / (log(Rt/(R*exp(-B/T0)))))-273.15;

       }

       T=(array_T[0]+array_T[1]+array_T[2]+array_T[3]+array_T[4])/5;
```

```
 snprintf(temp,8,"%f",T);
 snprintf(temp_max,3,"%d",Tmax);
temp[4] = temp_max[0];
temp[5] = temp_max[1];
int s = slave.receive();
while (s != 1){
   s = slave.receive(); }
r = slave.write(temp, strlen(temp)+1);
if(T > Tmax){
        r = 0;
        g = 1;
        lcd.cls();
        lcd.printf("DANGER! T = %-2.0f C", T); }
 else{
    r = 1;
    g = 0;
    lcd.cls();
    lcd.locate(3,0);
    lcd.printf("T = %-2.1f C", T);}}}
```

Codice del Master

```
#include "mbed.h"

#include<string>

#include "TextLCD.h"


I2C i2c(PTC2, PTC1); //PTC2 = SDA PTC1= SCL (clock)

AnalogIn temperature(PTB1);

PwmOut r(LED_RED);

PwmOut g(LED_GREEN);

TextLCD lcd(PTA1, PTA2, PTD3, PTD2, PTD0, PTD5, TextLCD::LCD16x2B); // rs =PTA1 , e=PTA2, d4-d7 =  PTD2, PTD0, PTD5,PTA13 rs, e, d4, d5, d6, d7


AnalogOut buzzer(PTE30);


const int addr = 0x90;

const int Tmax_internal = 28; // temperatura massima del master

int Tmax_external; //temperatura massima dello slave

double Rt;

double Vout;

double Vcc = 3.3;

double R = 150000;

double B = 4600;

double T0 = 298.15; // 25°C

double array_T[5];

double T;

int j;


int com;

double Tslave; // temperatura dello slave

double Tmedia; // media

double Var;


char temp[8];

char temp_var[8];

char temp_max_ext[2];

char out[] =  "            DANGER!!! system overheated                  ";

char outi[] =  "           external system overheated T =    C            ";

char outs[] =  "           internal system overheated T =    C            ";

char out_slave[] = "T ext =    C";
```

```
char cmd[7];

int main() {

   while (1) {
      com = i2c.read(addr, cmd, 7, true);
      temp_max_ext[0] = cmd[4];
      temp_max_ext[1] = cmd[5];
      Tmax_external = atoi(temp_max_ext);

      cmd[4] = cmd[6];
      cmd[5] = cmd[6];
      for(j=0; j<5; j++){
         Vout = temperature.read()*3.3;
         Rt = (Vout/(Vcc - Vout))*R;
         array_T[j] = (B / (log(Rt/(R*exp(-B/T0)))))-273.15;
         }
      T = (array_T[0]+array_T[1]+array_T[2]+array_T[3]+array_T[4])/5;
      Tslave = atof(cmd);
      cmd[4] = temp_max_ext[0];
      cmd[5] = temp_max_ext[1];
      if(com == 1){
         Tmedia = (Tslave + T)/2;
         Var = (Tslave - T);
         snprintf(temp_var,8,"%f",Var);
         outs[45]=cmd[0]; //2
         outs[46]=cmd[1]; //5
         outs[47]=cmd[2]; //.
         outs[48]=cmd[3]; //1
            lcd.cls();
            lcd.printf("T system = %-2.1fC", Tmedia);


            lcd.locate(24,1);
            lcd.printf("D");
            lcd.locate(25,1);
            lcd.printf("i");
            lcd.locate(26,1);
            lcd.printf("f");
```

```
        lcd.locate(27,1);
        lcd.printf("f");
        lcd.locate(29,1);
        lcd.printf("i");
        lcd.locate(30,1);
        lcd.printf("-");
        lcd.locate(31,1);
        lcd.printf("e");
        lcd.locate(33,1);
        lcd.printf("=");
        lcd.locate(35,1);
        lcd.printf("%c", temp_var[0]);
        lcd.locate(36,1);
        lcd.printf("%c", temp_var[1]);
        lcd.locate(37,1);
        lcd.printf("%c", temp_var[2]);
        lcd.locate(38,1);
        lcd.printf("%c", temp_var[3]);
        lcd.locate(39,1);
        lcd.printf("C");
        wait(1);

        if(T < Tmax_internal && Tslave < Tmax_external){
           r = 1;
           g = 0;  // accende il led verde
           buzzer.write(0);   }

        if(T >= Tmax_internal && Tslave >= Tmax_external){
        for(int i=0; i < 50; i++){
        r = 0;
        g = 1;
        wait(0.2);
        lcd.cls();
        lcd.printf("%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c" ,
out[i],out[i+1],out[i+2],out[i+3],out[i+4],out[i+5],out[i+6],out[i+7],out[i+8],out[i+9],out[i+10],out[i+11],out[i+12],out[i+13],out[i+14],out[i+15],out[i+
16]);

        buzzer.write(0.5);}}
```

```
        else{
            if(Tslave >= Tmax_external){
              for(int s=0; s < 42; s++){
              r = 0;
              g = 1;
              wait(0.2);
              lcd.cls();
              lcd.printf("%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c" ,
outs[s],outs[s+1],outs[s+2],outs[s+3],outs[s+4],outs[s+5],outs[s+6],outs[s+7],outs[s+8],outs[s+9],outs[s+10],outs[s+11],outs[s+12],outs[s+13],outs[s
+14],outs[s+15],outs[s+16]);
                                  } }

            if(T >= Tmax_internal){
              snprintf(temp,8,"%f",T);
              outi[45]=temp[0];
              outi[46]=temp[1];
              outi[47]=temp[2];
              outi[48]=temp[3];
              for(int t=0; t < 42; t++){
              r = 0;
              g = 1;
              wait(0.2);
              lcd.cls();
              lcd.printf("%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c" ,
outi[t],outi[t+1],outi[t+2],outi[t+3],outi[t+4],outi[t+5],outi[t+6],outi[t+7],outi[t+8],outi[t+9],outi[t+10],outi[t+11],outi[t+12],outi[t+13],outi[t+14],out
i[t+15],outi[t+16]);
                                   }}}}}}
```